ORIGINAL ARTICLE e-ISSN: 2349-0659 p-ISSN; 2350-0964

Development of Fog-based Dynamic Load Balancing Framework for Healthcare using Fog Computing

Sejal Bhavsar[1], Kirit Modi[2]

Abstract

Fog computing has become one of the leading technologies by conquering the many significant challenges in IoT, Big Data, and Cloud. Computing models are inclining toward Fog than Cloud due to faster processing. The numerous idle devices near the users help overcome the issue of latency found in the Cloud. Resource management through load balancing plays an essential role in efficient data processing. Based on the current pandemic situation, Emergency patient's vital sign monitoring system for COVID and other variants is implemented with support of dynamic resource load balancing environment. Apart from this, previously, we have faced many such diseases such as plague and flu which were pandemic and have become normal diseases now. Apart from them, there are many critical conditions and diseases such as hypertension, kidney failure, heart attack, cancer, lung, and liver disease that need continuous monitoring. It is not feasible to treat all patients at the hospital as the count is increasing very speedily. There is a need for infrastructure to handle resource issues without any delay in the treatment of patients using the fog computing. The proposed approach DynaReLoad would provide prompt health services and prevent early deaths due to critical conditions. An immediate alert to the doctors will be generated when detecting any abnormality. The effectiveness of DynaReLoad has been analyzed with other load balancing algorithms to achieve a low latency with minimum MakeSpan, better scheduling time, and response time, maximizing load balancing level and resource utilization using iFogSim.

Keywords: Cloud computing, E-healthcare, Fog computing, IoT, Vital-signs monitoring

Asian Pac. J. Health Sci., (2022); DOI: 10.21276/apjhs.2022.9.4S1.02

Introduction

Many significant challenges in IoT, Cloud, and Big Data have been trounced by Fog computing, making it one of the leading technologies in use. It helps enhance the quality of many services as it is data sensitive, causes reduced latency in day-to-day activities, and improves real-time applications with being delay-sensitive. Fog computing also ensures that no resource is overloaded and under loaded by proper distribution of workload dynamically with shared pool of computing resources. Load balancing directs to reduce response time, improve latency, reduce scheduling time, communication cost, and balance distribution of resources. Malfunctions in the environment, fluctuation in resources, and performance deviations are managed through resource allocation and scheduling techniques by means of load balancing.

IoT plays a prominent role in managing everyday jobs as it connects actuators, sensors, and smart devices (which include energy controllers, vehicles, mobile phones, traffic controllers, and computers) to the internet. It links everything and makes them smarter. Cisco made a prediction that nearly 51 billion devices with each other worldwide will be associated by 2021. It is estimated that nearly 2.3 zetta bytes will be generated by these devices every year. The conventional databases are incapable of processing this vast amount of structured and unstructured data. This increases the need for a technology that can analyze the collected data to extricate functional insights to make essential decisions. Cloud computing is one of the most effective choices to support processing, storage, communication, computation, and distribution, as smart devices only allow limited storage. According to Rangras (2011), Cloud[1] provides IaaS, PaaS, and XaaS common services. The speed at which processing takes place in the cloud is less than that required by IoT applications.[2] The inherent hindrances of cloud, that is, unmanageable traffic congestion, bandwidth constraints, variable latency, and lack of mobility, and pose challenges when dealing with IoT's demands. The primary reason for these issues is the large distance between the data centers of cloud service providers like that of Facebook, Amazon Web Services (AWS), and Google.

Fog computing has surfaced as an alternative and it follows a decentralized computing concept, unlike Cloud computing which exclusively relies on any central component.[3,4] The numerous idle devices near the users are helpful to overcome the issue of latency found in the cloud, but the more complex processing is accomplished by the cloud instead of fog. Primary devices such as routers, base stations, and smartphones can pose as Fog devices as they are equipped with storage space and processing capacity, sometimes with multiple cores.

Some research challenges such as heterogeneous organization and global connectivity are being conquered by Fog computing. The prime challenge posed by Fog computing is

[1]Department of Computer Science and Information Technology, Ganpat University, Gandhinagar Institute of Technology, Moti Bhoyan, Gujarat, India.

[2]Department of Computer Engineering and Information Technology, Sankalchand Patel University, Mehsana District, Gujarat, India.

Corresponding Author: Sejal Bhavsar, Research Scholar, Department of Computer Science and Information Technology, Ganpat University, Gandhinagar Institute of Technology, Moti Bhoyan, Gujarat, India. E-mail: seju812@gmail.com

How to cite this article: Bhavsar S, Modi K. Development of Fog-based Dynamic Load Balancing Framework for Healthcare using Fog Computing. Asian Pac. J. Health Sci., 2022;9(4S1):3-11.

Source of support: Nil

Conflicts of interest: None.

Received: 13/4/2022 **Revised:** 15/5/2022 **Accepted:** 11/6/2022

resource management. It has become essential to inspect these service requirements and resource management. Many reviews on Fog computing have been conducted by various reviewers. The researchers[5] described the concepts of Fog computing, description, framework, various applications, architecture, applications, and challenges of it. Hierarchical architecture that contains various layers related to resource management, computing, security, storage, and communication. The researcher[6] represented different views of platform level Fog computing and several proportions of applications, architecture, and challenges. Serious problems lead due to any kind of loss or delay throughout processing and retrieving of precise sensor data. The significant issue is how to balance resource management at platform level in Fog computing? An efficient scheme needs to be designed to process and balance the data.

The key objectives are summarized as follows:

- For better scheduling of resources, need to propose dynamic load balancing method that is useful in managing platform level issues, that is, scheduling and management of resource in Fog computing.
- As per the current pandemic situation, need to propose dynamic load balancing approach for the application scenario (Emergency patient's vital sign monitoring system for COVID-19 patients and critical care patients.
- Need to simulate scenarios for three different configurations: Cloud-based implementation, Fog-based implementation, and dynamic load balancing using fog computing, each containing a numerous sensors for each Fog device to calculate the various quality of service parameters.
- Need to compare proposed approach with the other load balancing algorithms to achieve better quality of service parameters.

The paper is arranged in the following way. Section 2 covers necessary Fog computing background information, its current research trends, comparison between Fog and other computing technologies, including cloud computing. Section 3 presents recent research work related to the existing Fog computing paradigm, framework, and approaches for issues related platform level design. Section 4 presents the proposed framework and algorithm DynaReLoad for platform level design concerns in Fog computing. The proposed algorithm and framework DynaReLoad for the application scenario are explained in Section 5. Section 6 covers the experimental simulation setups. Comparative result analysis is shown in Section 7. To encapsulate this research, Section 8 presents a conclusion.

Background

Fog computing paradigm is a platform where multiple heterogeneous devices can implement processing and storage tasks while communicating with each other. They do not function under any central device and stand independent to each other. It allows the services delivered by the Cloud to be at edge of the network (closer to users). It also enables users to be mobile and spread across great geographical regions. This improves latency as well as the quality-of-service parameters.

Fog is designed to support applications that require low latency[7] as it employs numerous nodes distributed across large areas. A device that supports networking, computation, and can perform storage functionalities can be used as a Fog device which includes routers, proxy servers, and any other device capable of computation. It is one of the basic elements in the Fog environment, along with Fog servers and gateways. This developing computing paradigm has encountered some new challenges in the past few years.

Multiple architectures for Fog computing exist, many of which are in the form of clusters of heterogeneous devices. In contrast, data centers are the primary physical component of the Cloud and this increases the energy, resource consumption, and the operational costs of cloud computing. Conversely, Fog consumes comparatively low energy and also reduces operational costs. The distance between a user and a Fog device might be equal to just a few hops as the devices are closely placed to the users, as mentioned.[8] The major variance among the two is that Fog is based on a geographically distributed approach, when the cloud follows a centralized approach.[9] One of the major issues of the cloud is high latency which does not allow real-time interaction. This issue can be settled by Fog computing. However, as compared to cloud, there are high chances of failure in as it follows decentralized management, wireless connectivity, and is prone to power failure.[10,11] Overall, it should not be supposed that Fog is a better alternative to cloud as both works differently by satisfying different requirements and perspectives.

There has been an increased interest in data processing near the users in the preceding few years. Hype cycle[12] concludes that Fog computing can improve the smart home technology if both are integrated properly. To showcase the latest emerging technologies, A Hype Cycle is created. As presented by the Hype Cycle, a few significant technologies comprise of autonomous vehicles and edge computing.

In addition, numerous articles and papers related to Fog have been studied and analyzed. Figure 1 showcases analogous technologies to Fog designed over the recent years.

However, in the past 1 year, the search tendencies have reduced by beyond 3 times for edge computing. After the influx in research for edge computing, the two top searched computing technologies are Mobile Edge and Mobile Cloud computing. Fog Dew and Dew computing experienced the lowest trends. Searches regarding Fog computing have been escalating with passing years, making it one of the rapidly growing research topic.

The platform performs the role of management and maintenance. It oversees scheduling and allocation of resources

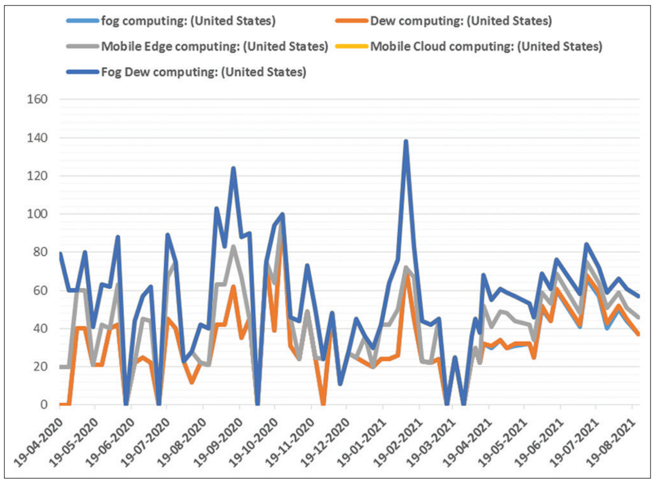

Figure 1: In Google scholar, search occurrence of computing technologies similar to Fog computing

along with other issues in Fog computing. Since Fog works with heterogeneous devices, the development of proper resource allocation and scheduling is a major challenge. Availability and efficiency are the two vital constraints for resource management using load balancing. Resources need to at hand as they are not dedicated to specific processing tasks. If an effective system for allocation and scheduling is not implemented then, processing can face undesirable delays. Next section shows related work of Fog computing.

Related Work

The following section explains the work related to IoT, Cloud, and Fog computing. It also describes the work concerned with scheduling and allocation of Fog computing resources.

The phrase "Internet of Things" was used for the initial phase by Kevin Ashton in the year 1999 on supply chain management presentation. The researchers[2] introduced that storage capacity and processing power are fairly limited, giving rise to problems such as privacy, security, performance, and reliability. Cloud is capable of processing the data accumulated by IoT devices using its storage and processing resources in batch format. Joseph Carl Robnett created Cloud computing in the 1960s for the purpose of accessing data from anyplace and at any given moment.

Fog computing is either the same as Edge computing. Services are processed to the Edge instead of cloud to ensure better reliability, shorter response time, and save bandwidth. The author examined the usage of Fog computing as a substitute for IoT.[13] The concerns and challenges for the integrations of IoT and Fog need to be systematically reviewed and synthesized. According to author,[9] Cloud resources and Edge of the network can be integrated easily with the use of Fog computing.

The papers mentioned above conclude that Fog computing might be capable of overcoming the shortcomings of cloud as it is based on real-time interactions which drastically reduces bulk processing. However, Fog computing is equipped with challenges itself including resource allocation and scheduling.

Alsaar *et al*.[14] designed an algorithm for resource allocation which uses linear decision tree rules for balancing workload. Researcher[15] thought of integrating old records of cloud customers through a Fog environment which would help in estimation of resources required. The author[16] created a Module Mapping Algorithm. The primary function of the algorithm is efficient distribution of IoT devices in in Fog-Cloud infrastructure. An algorithm found optimal methods to reduce the consumption of energy and distribute resources effectively.[17,18] The research portrays that Fog computing can improve the healthcare sector to a great degree by allowing faster data collection and proper resource allocation, saving multiple lives which are lost because of delay in treatment. The author proposed a method which allowed scheduling of health-care tasks based on priority through load balancing and efficient resource scheduling.[19] An Alert and Emergency Management system proposed[20] to send immediate notifications in case of emergency by multiple calculations based on load balancing. A health-care system based on CDS utilized classification based on decision tree.[21] Its major limitation was that high latency since processing was carried out in Cloud layer. A remote health monitoring system is designed for collecting data through wearable sensors.[22] Next section focuses on the proposed architecture and approach.

Problem Formulation Method and Proposed Algorithm

The main objective of the proposed approach is to achieve better quality of service parameters using following problem formulation model. The main aim is to minimize overall, scheduling time, better load balancing level, good resource utilization, good response time, better latency, and better MakeSpan using proposed approach.

To manage fog requests, suppose Fog layer has N amount of fog nodes

Fog nodes= {ʃg$_1$, ʃg$_2$,…., ʃg$_N$}

There are various physical machine located in the scenario. Consider all the physical Machine with the Þᴍ$_s$ = {Þᴍ$_1$, Þᴍ$_2$, Þᴍ$_3$,…, Þᴍ$_N$}. There are total N number of physical machines. Multiple virtual machines are associated with the Þᴍ. Here, we are considering the Fog nodes as a virtual Machine.

ὐᴍ = {ὐᴍ$_1$, ὐᴍ$_2$, ὐᴍ$_3$,…, ὐᴍ$_N$}

Every ὐᴍ has approximately resources such as RAM, Network Bandwidth, CPU, nodes information, and storage. To manage consumer requests, the Fog layer contains multiple Fog nodes.

A Fog node has maximum requests from patients. Equation (1) describes the maximum number of requests.

$$R'_{\text{Total}} = \sum_{p=1}^{n} \left(R'_p\right) \tag{1}$$

Load balancer balances ὐᴍ load equally among all the ὐᴍs by distributing requests. The x tasks are executed by the ὐᴍ in parallel manner. Each ὐᴍ processes the tasks independently and runs its own resources.

a) MakeSpan: Completion time (CʼƮ) is the total execution time in which task get completely executed or scheduled. Completion time is defined time at which process completes its execution. Low MakeSpan should be required. Equation (2) defines the turnaround time (ƮãƮ). Turnaround time is the time difference between arrival time and completion time. As per the Equation (3), waiting time (WƮ) is the time difference between burst time and turnaround time.

ƮãƮ = CʼƮ−ɮʈ (2)

WƮ = ƮãƮ−ɮʈ (3)

Where, AT is the time at which a process arrives at the ready queue to initiate the execution. It defines time measurement in milliseconds. The burst time is denoted with ɮʈ. ɮʈ means time to process for its execution. It measures time in milliseconds. The arrival time is denoted with AT. MakeSpan is the maximum completion time required for a task. The objective is to alleviate the response time and MakeSpan of load balancing requests. Equation (4) explains the MakeSpan of r tasks on ὐᴍi.

MakeSpan$_r$ =Max (CʼƮ$_{r,i)}$ (4)

Where s ∈ ὐᴍ, ὐᴍ = {1,2,3,….,s,….y}, r ∈ Ʈ$_{task,}$ Ʈ$_{task}$ = { 1,2,3,…, r,..., x) and mapping of Ʈ$_{task}$ to ὐᴍ affects performance parameters. Now, the total tasks assigned to each ὐᴍ are dependent on the load balancing algorithm and end user requests.

The processing time and response time of the tasks are expressed using linear programming.

b) Response time: It, simply put, is the time taken by a Fog resource to complete a given task. For this time, the given node will be occupied and will not perform any other task. Equation (5) defines response time.

$$R\tau_{x,y} = \frac{\sum y \in \upsilon'\mu s(CTx)}{MakeSpan \times Number\ of\ \upsilon'\mu s} \quad (5)$$

Where, $RT_{x,y}$ is used to represent the total RT of the ύμs in the system and $C^{\tau}T_x$ defines the task completion time.

c) Latency: It is defined as the whole time taken by the system to respond to the receiver's data. It is a combination of the processing time and propagation time.

Latency (ρ) is computed as below Equation (6),

$$\rho = \rho_t + q \quad (6)$$

Here, ρ_t denotes the propagation time taken by the data to reach the Fog layer, Cloud layer, and Fog using load balancing and q indicates execution time, that is, processing time.

d) Scheduling time: Scheduling time effectively manage their computing resources and schedule the incoming request. Scheduling is the process of assembling incoming requests in a certain manner so that the available resources will be properly utilized. Equation (7) presents that scheduler needs to consider a number of constraints, including the nature of the task, the size of the task, the task execution time, the availability of resources, the task queue, and the load on the resources at the time of scheduling. The cost of each task, when it is to be assigned on any Fog server, is denoted by, that is, and the task execution time T_i^e.

$$S_\tau = T + T_i^e + R \quad (7)$$

e) Resource utilization: It is the complete utilization of each resource. Resource utilization ratio should be required high for the better performance. Equation (8) defines resource utilization.

$$\text{Resourcen utilization} = \frac{ßș + \dot{O}l}{fșs} \times 100\% \quad (8)$$

f) Load balancing level: High load balancing level should be required for better performance. Equation (9) defines load balancing level.

$$\text{Load Balancing level} = \frac{ßș}{fșs} \times 100\% \quad (9)$$

Where, the balanced number of FNs is denoted with ßș, the number of overloaded FNs is denoted with $\dot{O}l$, and the number of all available $fșs$ is denoted with FNs.

Proposed Algorithm: DynaReLoad

```
1.  DynamicResourceAllocation: closestDC ←null
2.  currEstimatedResponseTime ← null
3.  currEstimatedResponseDC ← null
4.  closestDC ← findClosestDC(dcArray)
5.  for each dc Є dcArray do
6.  calculateEstimatedResponseTime ← currEstimated ResponseTime(dc) < currestimated Response time ? calculateEstimatedResponseTime(dc): null
7.      currEstimatedResponseDC ← dc
8.      end if
9.  end for
10. if closestDC == currEstimatedResponseDC
11.     selectedDC ← closestDC
12. else
13. selectedDC ←currEstimatedResponseDC
14. end if
15. if selectedDC.vm[0].status == available
16.     allocate(selectedDC.vm[0])
17. else
18.     for each vm Є selectedDC.vms do
19.         if vm.status == available
20.             allocate(selectedDC.vm)
21. else
22. 22.vm.allocations < MAX (selectedDC.vms.allocations)
23.                 allocate(selectedDC.vm)
24.             end if
25.     end for
26. end if
```

The main goal of the DynaReLoad is find the closest data center from all the available data centers and find the good quality of service parameters, that is, fastest response time, scheduling time, better resource utilization, better load balancing level, and low latency. A data center is randomly chosen from proximity list when there is more than one closest data center. Initially, it first search or detect the closest data center when response time of closest data center is degrading. It will tagged the better response time data center as a quickset data center. After the quickest data center selection, the closest data center is selected as the destination data center. If the quickest data center and closest data center are not similar then it will randomly select the data center.

Initially current estimation response time and current estimation response time data center are assigned with the null values. First goal is to find the closest data center from all the available data centers. Now, calculate the fastest response time using a ternary operator. Next, it will check the current estimation response time and whether the closest DC is the same or not. After that, it is assigned to selected DC and find the data center. Let suppose it find data center 1 as closest. Then, the next task is to select Vm. Next section shows the proposed architecture and application scenario based on the current requirement.

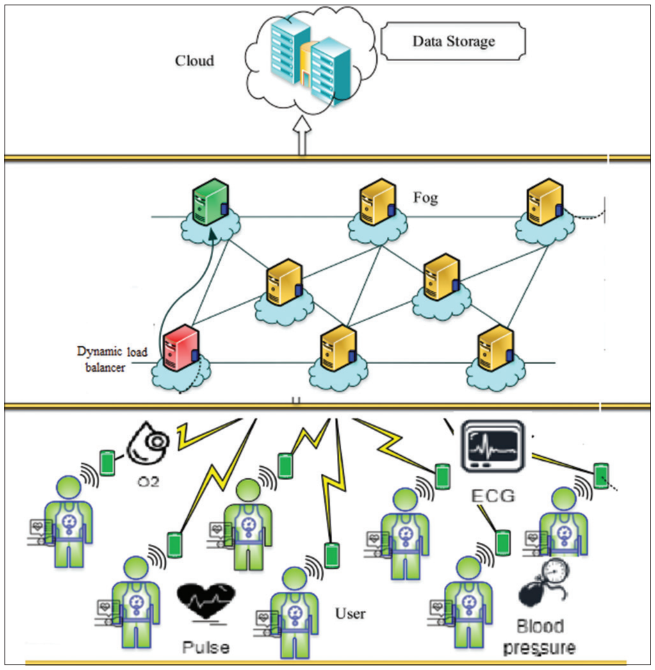

Figure 2: DynaReLoad architecture for emergency patient's vital sign monitoring system for COVID-19 pandemic situation outbursts

Proposed Architecture and Approach

The proposed Fog computing architecture and approach have been illustrated in Figure 2. The model, along with its platform level issues like resource scheduling and management using load balancing in Fog computing would be suitable for the proposed environment as it would help overcome the issues of latency and ensure that the processing is as efficient and quick as possible.

The Fog architecture is represented best through layered representation. Many analysers have analyzed different Fog architectures for process implementation, that is, three,[19] four,[22] and five.[23]

Components of Fog Computing Architecture

For the proposed framework DynaReLoad, a three-layer Fog architecture has been implemented.

a. End- user layer: The bottom layer consists of all devices which could function as the basic data source for Fog computing. These include smart devices, ECG sensors, O2 sensors, pulse monitoring sensors, and blood pressure monitoring sensors and would collect all physical data necessary. Virtual sensors are also a part of the physical layer as they can be used to estimate the conditions of a process using the readings of the physical sensors and making immediate decisions where necessary. In the proposed system, the sensors will collect data from the patients about their vital signs.
b. Fog layer: The Fog layer consists of Fog servers, Fog devices, gateways, and a dynamic load balancer. A group of virtual and physical sensors would be associated with the Fog devices and Fog server. All Fog devices associated to the same server will have the ability to communicate with each other when required. This layer also handles the various applications computation requirements. The dynamic load balancer would ensure that any computing load is distributed equally among all the nodes and none of them is overburdened.
c. Cloud layer: At the cloud layer, the data that need to be stored for a long time are sent by the Fog nodes to the cloud for processing and storage. It is only after processing in the Fog layer that the data flow component decides if it should be stored locally or for long-term storage in cloud. The main challenge is to minimize data volume by processing at the edge. Not all data are useful, and therefore, data trimming saves a vast amount of storage space. Next section layouts proposed algorithm and framework according to application scenario.

Proposed Algorithm and Framework (DynaReLoad) According to Application Scenario

Health-care systems surface massive dares as a significance of a cumulative number of patients and chronic diseases. The values of biometric parameters are measured manually in utmost hospitals. Much of the time is wasted in the manual process. We can reduce the cost and time with the help of automation process. The efficiency and quality achieved with the integration of Fog computing with dynamic load balancing in health-care systems. For example, consider Cloud computing used in the scenario of healthcare. If the storage and computation requests of all the patients are managed on the centralized single Cloud server, then it will outcome into multiple problems, that is, traffic congestion problem, enormous end to end delay, and huge network usage. Fog nodes are being used in the geographically distributed manner to resolve these issues. Congestion problem occurs if we perform numerous tasks on a single Fog node. In the multispecialty hospital, we have checked scenario like one machine or one server is not able to handle all the patients admit details and discharge details. One node is not capable to handle to all the requests.

The proposed algorithm and framework (DynaReLoad) are used to create a system for the monitoring of an emergency patient's vital signs caused due to the COVID-19 pandemic and critical diseases. With the quick increase in the number of affected patients, it is near impossible for hospitals to accommodate all patients. Consequently, many of the patients are not treated in time, leading to a higher mortality rate. To avoid any delay in treatment, the proposed framework will generate an immediate alert to the hospital and doctors in case of any abnormalities.

For a multi-storied multi-specialty hospital, there are enormous challenges in terms of data gathering and processing. For example, there are 1000 patients with 4-5 sensors attached to each of them, so for one patient the number of requests in a 1-min can be calculated as:

$$1 \text{ min} = 60 \text{ s},$$

Each second, there will be data from each sensor, so total data gathered in a minute = $60 \times 1000 \times 4 = 240000$ requests.

To handle this effectively, we can attach one Fog node to each unit of 200 patients. Hence, by limiting the number of requests hitting on a cloud, requests get distributed to Fog nodes. In that case, to avoid flooding of requests to one node, and for faster, fault-tolerant responses, the mechanism of load balancing can be taken into consideration.

To ensure that an immediate response can be generated, many quality-of-service parameters need to be maintained, which include higher latency, better execution time, faster response time, rapid request service time by the data center, and cost-efficiency.

Figure 3 shows that a patient suffering from COVID-19 or due to any critical disease issue is quarantined in critical room, which is equipped with sensors for monitoring the vital signs of a patient. Most of the affected people exhibit minor external symptoms such as sneezing, cough, shortness of breath, throat pain, tiredness, headache and fever, and severe internal symptoms (i.e., reduction in oxygen level, extreme variations in blood pressure and pulse, and ECG-related issues). The sensors will be responsible for monitoring these internal vitals through oximeter, blood pressure and pulse monitor, and ECG.

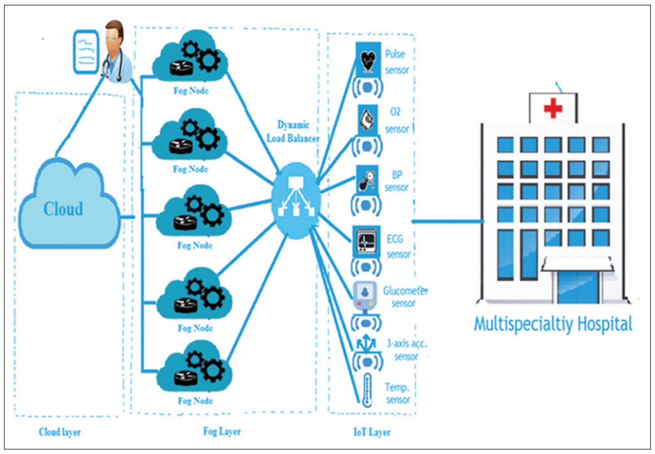

Figure 3: Proposed system Framework for multispecialty hospital

It is important to monitor these signs as these factors may affect the risk to any patient. Many patients might feel healthy and display no external symptoms, but they may have poor oxygen levels, which could cause severe lung issues. If the oxygen level of any patient drops below the normal oxygen levels (95% according to the WHO guidelines), then the doctors will be notified of the same. Similarly, an adult is expected to have a pulse rate of 60–100 beats/min and 70–100 beats/min for children and adolescents. (The maximum heart rate can be calculated using the formula, beats/min = 220 – one's age).

If the pulse rate is recorded to be less than that expected, an alert will be sent to the concerned personnel. The ECG sensor would work in a similar fashion and detect any abnormality found in the troponin level, which is the main determining factor for cardiovascular abnormalities.

The sensors will detect the current status of the patient and will send a status report to the doctor outside the ICU. This would reduce the frequent physical diagnosis required, allowing the doctors to check on the patient in a safe environment. The health history of any patient will also be registered in the system before being quarantined to the ICU, which would prove helpful when analyzing the health status.

The collected sensors data will then be transmitted to dynamic load balancer, which will then process the data on different Fog nodes and send updates for remote monitoring. According to the framework, the hospital is divided into small cells, each consisting of multiple Fog nodes. A dynamic load balancer would be responsible for randomly selecting the Fog nodes. It is able to overcome the problems posed by the separate cluster scenario. In such a scenario, the hospital would be divided into separate clusters, with each cluster head handling the requests sequentially. This Could cause a number of issues, that is, wastage of time, resource unavailability, reduced transmission time, and service migration. If the load balancer is placed on top of all sensors, these issues can be overcome.

In the given scenario, Fog nodes are assigned to the hospital with multiple Fog nodes on every floor. Initially, the accumulated data by the vital sign sensors are transmitted to the attached custom load balancer. The balancer is attached to every dedicated floor-wise Fog node. The nodes are connected to the Cloud, where further processing will be conducted for the Fog computing system. If any abnormality is detected, an immediate alert will be sent to the doctors. This would allow the patients to receive timely health services,[24] preventing early deaths due to the virus. Next section exhibits the simulation setup and results using iFogsim simulator.

Simulation Setup

In proposed application simulations, different scenarios are analyzed and evaluated, where the proposed algorithm and framework are implemented in three different configurations: Cloud-based, Fog-based, and dynamic load balancing Fog-based configuration. Four sensors are used to detect blood pressure, ECG, oxygen level, and pulse. The captured signals and data are transmitted to the Fog nodes. Fog nodes then further process the received data to detect the status. iFogSim has been used to simulate and evaluate our application scenarios in terms of response time, latency, and resource utilization. Table 1 represents the experimental setting parameters along with Fog node configuration.

Figure 4 represents the topology created by iFogsim to evaluate dynamic load balancing using Fog-based architecture as per the defined in Algorithm 1 and the proposed framework. The metrics under observation are latency, network utilization, response time, etc. Fog nodes are created, each equipped with sensors to measure the vital signs of a patient. Data generated by the sensors are processed at virtual machines placed at the Fog nodes. A dynamic load balancing algorithm is used to process data in a fast way.

Table 1: Experimental setting parameters

Total no. of users	*60/min*
No. of sensors/each user	4 for normal COVID patients. It can be vary according to the requirement.
Storage cost	0.1
No. of fog nodes	200 patients/fog node
Configuration of fog node	1 GB Storage
Resource Costs	1.0
Bandwidth	1000 MBs
Memory Cost	0.05

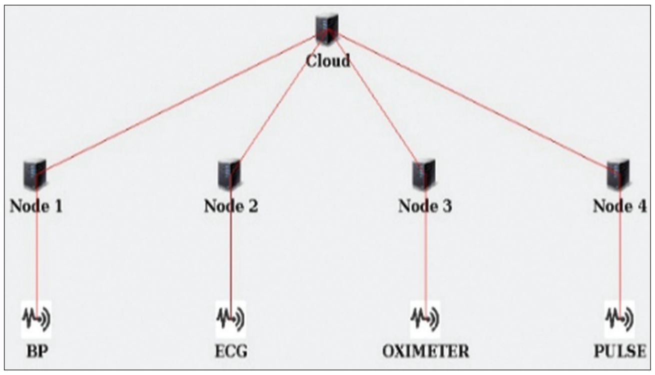

Figure 4: Topology of emergency patient's vital sign monitor

For evaluating Fog-based dynamic load balancing implementation in iFogSim, the parameters include the central processing unit computing capacity in MIPS, bandwidth of uplink, architecture level, RAM, processing in terms of rate per million instructions, busy power, downlink bandwidth, and idle power.

To run the final simulations, import the topology in iFogsim simulator, which is already created. The authors will compare cloud-based execution, Fog with the closest data center, and Fog with dynamic load balancing. The topology comprises the Fog nodes connected through a network and are attached to a load balancer. This load balancer distributes the traffic onto the underlying Fog nodes. Here, initially, only two buttons were available in the foganalyst header, that is, canvas and execution. To better understand the comparison experimental part, the authors have created two more drop down buttons in the foganalyst. Hence, the authors are able to compare results using Cloud computing and Fog computing with closest data center and dynamic load balancing using Fog computing.

Comparative Result Analysis Using Quality of Service Model

Latency

As number of sensors increases, the Cloud-based architecture increases latency significantly. The Fog nodes only process the sensed data of the sensors attached to them in Fog-based configuration. Defiantly, latency is increases in the cloud because cloud server requires to process all the sensor requests. Dynamic load balancing based Fog scenario provides less latency as compare to other two approach as per Figure 5.

Here, latency in Cloud-based approach increases tremendously with increasing number of sensors because cloud server will have to process all the sensors requests. In contrast, Fog with dynamic load balancing based approach, the Fog nodes only process the data sensed by the sensors attached to them, resulting in reduced latency.

Dynamic load balancing-based Fog scenario provides the least latency when compared with the other two approaches. Similarly, Figure 6 shows the comparison of load balancing algorithms. As per the results, DynaReload approach provides reduced latency as compared to the other load balancing algorithms, that is, round robin, active monitoring, and throttled.

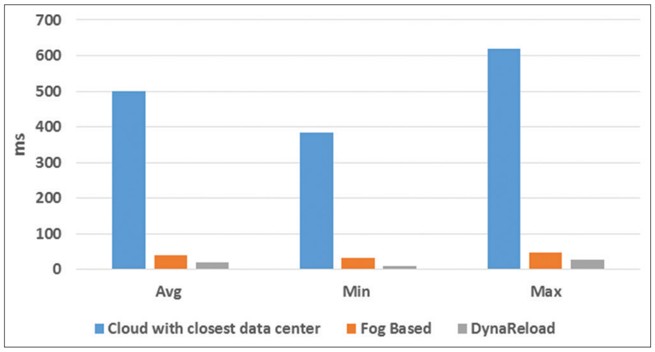

Figure 5: Comparison of latency for different configuration

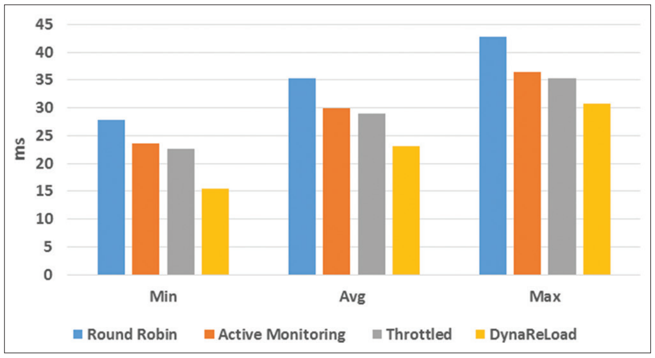

Figure 6: Comparison of load balancing algorithms

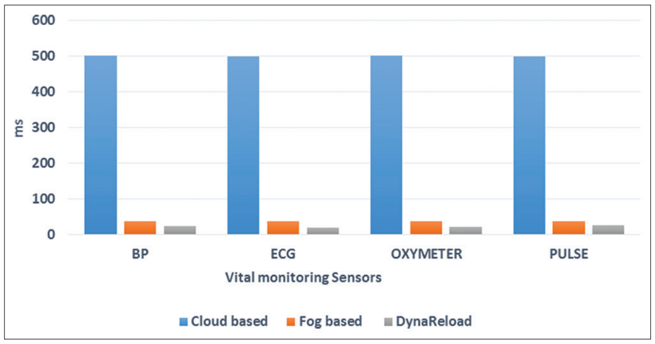

Figure 7: Response time by region

Response Time

Response time, simply put, is the time taken by a Fog resource to complete a given task. For this time, the given node will be occupied and will not perform any other task. Figure 7 presents the response time of a Fog node for processing the data sensed by different vital signs monitoring sensors in a Cloud-based, Fog-based, and dynamic load balancing-based architecture.

Figure 8 represents the response time of a Fog node for processing the data sensed by different vital signs monitoring sensors in a round robin, active monitoring, throttled, and dynamic load balancing-based architecture. Here, DynaReload approach provides better response time as compared to other load balancing algorithms.

Scheduling Time

Figure 9 represents the average, minimum, and maximum scheduling time taken by the three approaches (i.e., Cloud-based, Fog-based, and dynamic load balancing Fog-based.) dynamic load balancing approach covers less scheduling time as compare to Cloud-based configuration and Fog with closest data center configuration.

Figure 10 shows the overall scheduling time of different load balancing algorithms. Here, the DynaReload algorithm processes data in less scheduling time as compared to round robin, throttled, and active monitoring algorithms.

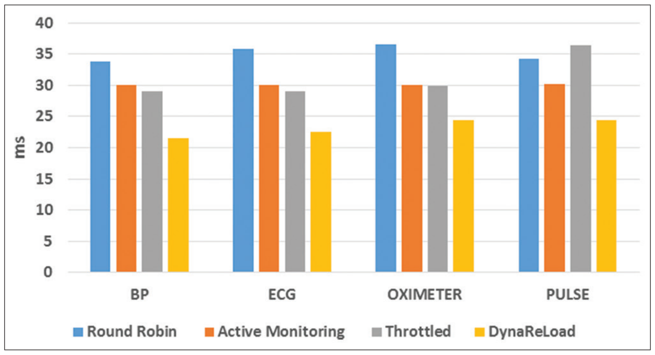

Figure 8: Comparison of load balancing algorithms

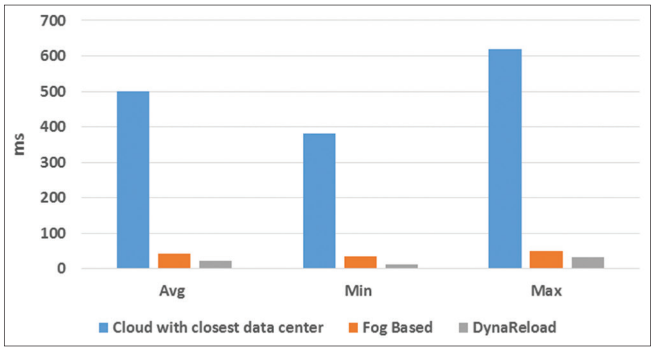

Figure 9: Overall scheduling time

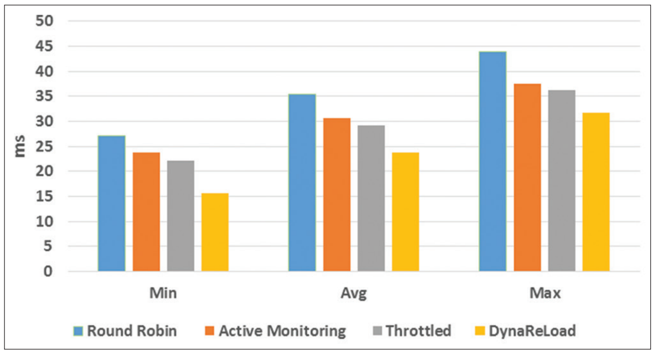

Figure 10: Comparison of load balancing algorithm in view of overall scheduling time

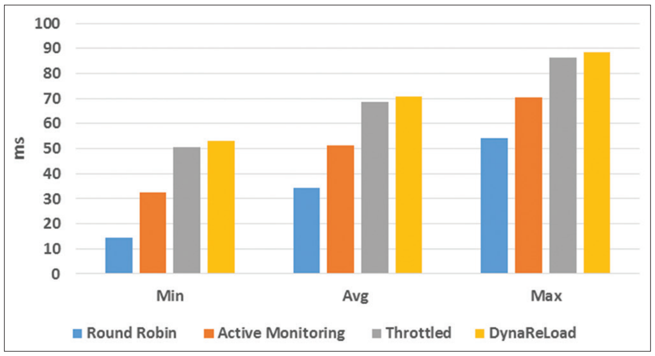

Figure 13: Comparison of DynaReload with other load balancing algorithm using load balancing level

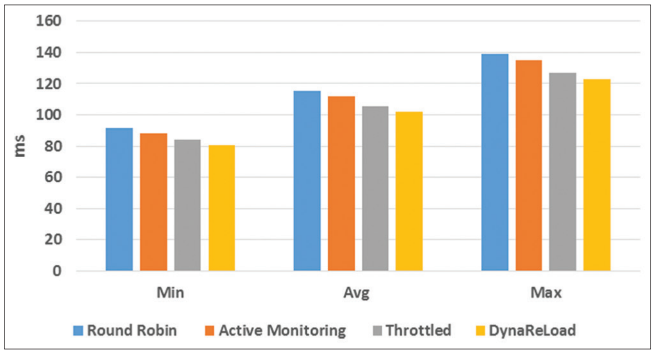

Figure 11: Comparison of DynaReload with other load balancing algorithm using MakeSpan

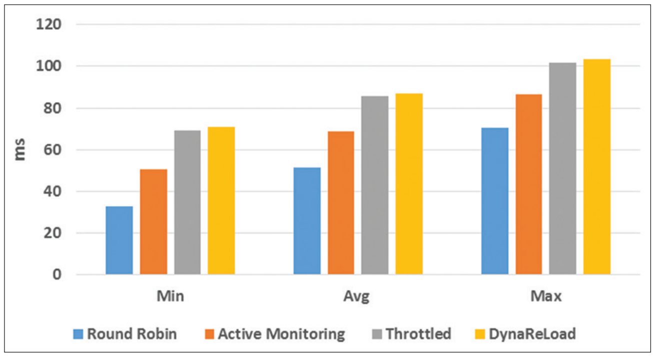

Figure 12: Comparison of DynaReload with other load balancing algorithm using resource utilization

Here, DynaReload approach provides lower MakeSpan as compared to other load balancing algorithms. DynaReload also provides better response time as compared to other load balancing algorithms as per Figure 11.

Above Figures 12 and 13 explained that DynaReload algorithm gives better result as compared to round robin, active monitoring, and throttled algorithms. Here, DynaReload algorithm provides higher load balancing level and higher resource utilization.

All the given figures present a comparison of latency, scheduling time, and response time MakeSpan, resource utilization and load balancing level for all four approached used. The findings enable us to understand that most improvement can be achieved if a Fog-based dynamic load balancing approach DynaReload is employed for the given scenario, which would allow fast service care provided to the patients. The simulation results implemented on altered scales confirm the efficacy of the proposed architecture for the implementation of emergency patient's vitals sign monitoring system. Next section concludes the research and defines scopes of future work.

Conclusion

With the increasing health conditions and pandemics in society, a need for remote health monitoring system has arisen to ensure medical services for every patient. A few remote health-care applications are available in the market based on Cloud computing, but they pose a major challenge of high latency, less privacy, more energy consumption, and less data security. Fog computing is now being used as an alternative as it serves as a middle-ware by bringing application services and computing resources closer to the edge where the data are being generated, overcoming the challenges presented by Cloud computing. The aim of the research is the resource utilization in efficient way with the least delay by assigning Fog servers between the end-users and cloud. All servers carry an equally distributed load which would prevent the breakdown of overburdened servers. This is achieved by shifting the load (using load balancing) from the overburdened server to an idle one nearby by even distribution of load at the fog layer.

The DynaReload allows dynamic load balancing for each type of computing node in the Fog and Cloud. DynaReload approach provides better results in terms of overall scheduling time, response time, and latency as compared to Cloud-based approach and Fog-based normal approach. Again, it gives good quality results in terms of latency, scheduling time, response time, MakeSpan, and load balancing level as compared to other load balancing algorithms. Furthermore, the proposed system is designed for multispecialty hospital. In the future, the authors are excited to design a system that would manage to prescribe medicines after monitoring the sensors values to enhance the overall health of a patient.

Nomenclature

Ŕ	Response Time
dc	Data Center
cdc	Closest Data Center
ert	Estimated Response Time

ὐɥ = Virtual Machine
ρ = Latency
ρ_t = Propagation time
ɋ = Execution time
$T_{(i)}^{e}$ = task execution time
S_τ = Scheduling Time
ß = Bandwidth
ƇƮ = Completion time
ƮãƮ = Turnaround time
$ƺ_{ƫ}$ = Burst time
Fș = no of Balanced FNs
Ȯƚ = no of Overloaded FNs
FNs = no of all available *ƒșs*.

References

1. Rangras J, Bhavsar S. Design of framework for disaster recovery in cloud computing. In: Data Science and Intelligent Applications. Singapore: Springer; 2021. p. 439-49.
2. Shah Y, Thakkar E, Bhavsar S. Fault tolerance in cloud and fog computing-a holistic view. In: Data Science and Intelligent Applications. Singapore: Springer; 2021. p. 415-22.
3. Mahmud R, Buyya R. Fog Computing: A Taxonomy, Survey and Future Directions. United States: Internet of Everything, Springer; 2017.
4. Gao L, Luan TH, Yu S, Zhou W, Liu B. Fogroute: Dtn-based data dissemination model in fog computing. IEEE Internet Things J 2017;4:225-35.
5. Hu P, Dhelim S, Ning H, Qiu T. Survey on fog computing: Architecture, key technologies, applications and open issues. J Network Comput Appl 2017;98:27-42.
6. Perera C, Qin Y, Estrella JC, Reiff-Marganiec S, Vasilakos AV. Fog computing for sustainable smart cities: A survey. ACM Comput Surv 2017;50:32.
7. Li J, Zhang T, Jin J, Yang Y, Yuan D, Gao L. Latency estimation for fog-based internet of things. In: Telecommunication Networks and Applications Conference (ITNAC), 2017 27th International. United States: IEEE; 2017. p. 1-6.
8. Bhavsar S, Pandit B, Modi K. Social internet of things. In: Integrating the Internet of Things in to Software Engineering Practices. United States: IGI Global; 2019. p. 199-218.
9. Mahmud R, Koch FL, Buyya R. Cloud-fog interoperability in iot-enabled healthcare solutions. In: Proceedings of the 19th International Conference on Distributed. New York, United States: Association for Computing Machinery; 2018.
10. Kai K, Cong W, Tao L. Fog computing for vehicular Ad-hoc networks: Paradigms, scenarios, and issues. J China Univ Posts Telecommun 2016;23:56-96.
11. Peng M, Yan S, Zhang K, Wang C. Fog-computing-based radio access networks: Issues and challenges. IEEE Netw 2016;30:46-53.
12. Top Trends in the Gartner Hype Cycle for Emerging Technologies; 2017. Available from: http://www.gartner.com/smarterwithgartner/top-trends-in-thegartner-hype-cycle-for-emergingtechnologies-2017 [Last accessed on 2022 Mar 12].
13. Dasgupta A, Gill AQ. Fog Computing Challenges: A Systematic Review. South Africa: Australasian Conference on Information Systems; 2017.
14. Alsaffar AA, Pham HP, Hong CS, Huh EN, Aazam M. An architecture of iot service delegation and resource allocation based on collaboration between fog and cloud computing. Mobile Inform Syst 2016;2016:6123234.
15. Aazam M, St-Hilaire M, Lung CH, Lambadaris I, Huh EN. Iot resource estimation challenges and modeling in fog. In: Fog Computing in the Internet of Things. United States: Springer; 2018. p. 17-31.
16. Taneja M, Davy A. Resource aware placement of iot application modules in fog-cloud computing paradigm. In: Proceeding of the IFIP/IEEE Symposium on Integrated Network and Service Management (IM). United States: IEEE; 2017. p. 1222-8.
17. Pooranian Z, Shojafar M, Naranjo PG, Chiaraviglio L, Conti M. A novel distributed fogbased networked architecture to preserve energy in fog data centers. In: Proceedings of the 2017 14th International Conference on Mobile Ad Hoc and Sensor Systems, Orlando, FL, USA: IEEE; 2017. p. 22-5.
18. Al-Khafajiy M, Baker T, Chalmers C, Asim M, Kolivand H, Fahim M, *et al*. Remote health monitoring of elderly through wearable sensors. Multimedia Tools Appl 2019;78:24681-706.
19. Aladwani T. Scheduling IoT healthcare tasks in fog computing based on their importance. Proc Comput Sci 2019;163:560-9.
20. Velásquez W, Munoz-Arcentales A, Salvachúa J. Fast-data architecture proposal to alert people in emergency. In: 2018 IEEE 8th Annual Computing and Communication Workshop and Conference (CCWC). United States: IEEE; 2018. p. 165-168.
21. Naghshvarianjahromi M, Kumar S, Deen M. Brain-inspired intelligence for real-time health situation understanding in smart E-health home applications. IEEE Access 2019;7:180106-26.
22. Mit HR, Diyanat A, Pourkhalili A. Mist: Fog-based data analytics scheme with cost-efficient resource provisioning for iot crowdsensing applications. J Network Comput Appl 2017;82:152-65.
23. Varshney P, Simmhan Y. Demystifying fog computing: Characterizing architectures, applications and abstractions, in Fog and Edge Computing (ICFEC). United States: 2017 IEEE 1st International Conference on IEEE; 2017. p. 115-24.
24. Bhavsar S, Modi K. Design and Development of Framework for Platform Level Issues in Fog Computing. In Research Anthology on Architectures, Frameworks, and Integration Strategies for Distributed and Cloud Computing. Belgium: IGI Global; 2021. p. 429-51.