Journal of Data Acquisition and Processing

ISSN: 1004-9037
https://sjcjycl.cn/
DOI: 10.5281/zenodo.7766110

IMPLEMENTATION OF DYNAMIC LOAD BALANCING FOG COMPUTING BASED FRAMEWORK FOR HEALTHCARE

SejalBhavsar[1*], KiritModi[2] and Eshani Patel[3]
[1]Research Scholar, Ganpat University, India; seju812@gmail.com
[2]Professor & Head, Sankalchand Patel University, India; kiritmodi@gmail.com
[3]Research Scholar, University Of Cumberlands, USA; Eshani.26@gmail.com
* Correspondence: seju812@gmail.com

Abstract: By overcoming the several critical obstacles in IoT, Big Data, and Cloud, fog computing has emerged as one of the top technology. Because fog processes information more quickly than cloud, computing paradigms are trending that way. The abundance of idle devices close to users aids in overcoming the cloud's latency problem. A key component of effective data processing is resource management through load balancing. A dynamic resource load balancing environment is used to create a method for monitoring vital signs for Covid and emergency patients based on the pandemic condition. In addition to this, we have previously encountered a variety of illnesses like the plague, the flu, and others that were pandemics. Aside from them, there are other serious illnesses that require constant observation, such as cancer, hypertension, a heart attack, lung and liver disease, kidney failure. As the hospital's patient population is rapidly growing, it is not possible to treat every patient there. When using fog computing to treat patients, infrastructure is required to solve resource problems quickly. DynaReLoad suggested strategy would offer quick access to medical care and stop early deaths from critical diseases. Any anomaly will result in an urgent alarm being sent to the doctors. The DynaReLoad has gained improvement in results with minimum latency 23.20ms, makespan 101.94ms, scheduling time 23.76ms and response time 21.61ms, maximizing load balancing level 70.94ms and resource utilization 87.19ms as compared to other Load balancing algorithms using iFogSim.

Keywords: Load balancing; Fog computing; IoT, cloud computing; Healthcare; vital-signs monitoring sensors

1. Introduction

Fog Computing is one of the most widely used technology today, having successfully overcome numerous important Internet of Things, Cloud, and Big Data difficulties. It reduces latency in daily operations, enhances real-time applications because to its delay sensitivity, and helps to improve the quality of many facilities because it is data sensitive. By properly allocating workload with a shared pool of computing resources in a dynamic manner, fog computing ensures that no resource is overused or overcrowded. Reduce response times, low latency, shorten scheduling time, cut communication costs, and balance resource utilization are all goals of load balancing. Breakdowns in the situation, performance deviations and resource variation are managed through load balancing.

As it links actuators, sensors, and smart devices such as cars, mobile phones, traffic controls to the internet, IoT plays a significant role in handling daily tasks. They get smarter because it connects everything. By 2021, over 51 billion gadgets will be linked together globally, according to a Cisco projection. These gadgets are expected to produce close to 2.3 zettabytes of data annually. This enormous amount of structured and unstructured data cannot be processed by conventional databases. Because of this, there is a greater demand for technology that can analyse the gathered data and extract useful insights for important decision-making. One of the best options for supporting processing, storage, communication, calculation, and dissemination is cloud computing, due to a lack of storage, smart devices only support a small storage. Rangras (2011) suggested that PaaS, IaaS and XaaS common services are offered by cloud [15]. Processing occurs more slowly in the cloud than is necessary for IoT applications [13]. Dealing with the demands of IoT presents difficulties due to the inherent limitations of cloud, such as uncontrollable traffic congestion, bandwidth restrictions, fluctuating latency, and lack of mobility. The main cause of these problems is the high distance between the data centres of cloud service providers such as AWS, Facebook and Google.

Cloud computing relies on central component, whereas fog computing has emerged as an alternate and adheres to a decentralised computing concept [1][2]. The many idle devices close to users are beneficial to get over the cloud's latency problem, however the more difficult processing is handled by the cloud in place of fog.

Fog computing is overcoming some research hurdles like heterogeneous organisation and global networking. Resource management is the main difficulty presented by fog computing. Examining these service requirements and resource management has become crucial. Various reviewers have conducted a lot of reviews on fog computing. The researchers [3] suggested the principles of fog computing, its definition, architecture, framework, applications, and obstacles. The researcher [5] represented various perspectives on platform-level fog computing as well as various applications, architectural styles, and technical difficulties. Any form of delay or loss during the processing and retrieval of sensor data results in serious issues. How to balance resources is a crucial topic in fog computing at platform level? An effective framework needs to be designed to balance and process the data.

The following is the summarization of key objectives:

• Suggest a method to handle platform-level concerns, such as resource scheduling and management in fog computing with dynamic load balancing;

• Design a dynamic load balancing framework to monitor vital sign for emergency situation during pandemic;

• To compare simulations for different load balancing algorithms and other configuration based results in order to increase quality of service parameters.

The order of paper is as follows.

Background information of fog computing is represented in Section 2. Section 3 related work and literature review of the recent papers. Proposed formulation method and proposed algorithm are represented in Section 4. The proposed architecture and application scenario are included in Section 5. Section 6 includes simulation part. Section 7 analyses comparative section using quality of service parameters. Section 8 presents a conclusion.

2. Background

Multiple heterogeneous devices can carry out processing and storage operations on a platform called fog computing while interacting with one another. They stand independently of one another and are not controlled by a single entity. It enables cloud services to be offered at the network's edge (closer to users). Additionally, it makes users mobile and dispersed across vast geographic areas possible. Both the latency and the quality-of-service metrics are enhanced by this.

Due to the use of several nodes dispersed across large distances, fog is intended to assist applications that demand low latency [10]. Routers, proxy servers, and other compute-capable devices can all be employed as Fog Devices because they support networking, computing, and storage functions. Along with Fog servers and gateways, it is one of the fundamental components of the Fog ecosystem. Over the past few years, this evolving computer paradigm has faced some fresh difficulties.

There are numerous architectures of fog computing, many of which take the shape of clusters of heterogeneous equipment. However, because data centres make up the majority of the cloud's physical structure, they use more resources, energy, and money to operate. On the other hand, Fog uses relatively little energy and lowers operating expenses. Given that the devices are situated close to the users, as was noted before [6], It may take a few hops distance between a user and a fog device. The main distinction is that fog follows geographically distributed approach, whereas cloud is based on centralized approach [12].

High latency, which prevents real-time contact, is one of the cloud's main problems. Fog computing can solve this problem. But compared to the cloud, there are more risks for failure because it uses wireless communication, decentralised management, and is more vulnerable to power outages. [7][8]. Overall, it shouldn't be assumed that fog is a superior option than cloud because they both function differently by meeting distinct needs and viewpoints.

In the recent years, User interest increases towards data processing. According to google scholar survey, Fog computing is the trending technology as compared to dew computing, mobile edge computing and mobile cloud computing.

The primary focus of taxonomy is on infrastructural, platform-level, and application-based requirements. Infrastructural requirements in the fog are highly dependent on the nodes, network establish between these nodes, and their requests.

The Fog infrastructure also includes computation-related network devices, Fog devices and gateways that are present in the Fog environment. The platform is responsible for management and upkeep. It manages scheduling, resource distribution, and other aspects of fog computing. Fog operates with diverse devices, therefore creating an efficient resource allocation and scheduling system is quite difficult. The two key limitations for resource management utilising load balancing are efficiency and availability. Resources must be available because they are not set aside for particular processing tasks. Processing could experience unfavourable delays if an efficient allocation and scheduling system is not put in place. The following section displays related work in fog computing.

3. Related Work

This section shows the Internet of things, Cloud and Fog related work. The first literature related work is done on (Internet of Things)IoT. In the initial literature of IoT, it suffers from processing and storage issues.
The future focuses on Cloud computing, due to its on demand processing and storage capabilities. After Cloud computing literature, the focus is on integration of IoT and Cloud. The fog computing is capable of handling issues of cloud computing after integration. There are various challenges in fog computing like resource scheduling and allocation issues. Next literature is prepared on the fog computing with load balancing. After that, the next literature review related work is on healthcare sector as per the current need and problems. The research suggests that load balancing in fog computing with proper resource utilization and data collection saves multiple lives. The below table 1 shows the summarization of the literature review load balancing and healthcare. The framework is designed as per the literature survey to achieve efficient resource sharing, low latency, and minimum scheduling time, less response time, maximum makespan and load balancing level.

Table 1. Literature review summary on Load balancing and healthcare

Synthesis of research	Research Contribution	Research Gap
Utility-aware resource allocation for edge computing[19]	Improve latency with Resource provisioning and latency-aware scheduling algorithm	Improved latency by dynamic approach of Load balancing
Load balancing using Energy aware [17]	load balancing model using energy aware	Execution time, processing time
Workload balancing in fog and IoT [25]	Workload based load balancing model	Resource utilization. Load balancing level
Model for latency in healthcare [26]	Latency aware model	Reliable communication and dynamic load balancing
Health monitoring Model[23]	latency-aware model for healthcare	Resource utilization and processing time
Priority based load balancing model for healthcare [22]	Load balancing model based on priority for healthcare	Resource utilization and processing
Fog based health monitoring system for Remote in-home [18]	Latency, response time and energy consumption	Load balancing, scheduling time, resource utilization, Makespan
Remote pain monitoring system for e-healthcare [21]	Latency and network usage	Load balancing, scheduling time, response time, resource utilization, Makespan

After literature review, we found a few research gaps which prompted me to move towards dynamic load balancing techniques in healthcare. Again summary of dynamic load balancing says that it improves the efficiency of resources which requires less computing nodes, better resource utilization and better load balancing level. Next section focuses on the problem formulation model.

4. Problem formulation Method using QoS and proposed algorithm

The major goal of the proposed framework is to increase the QoS parameters by applying below problem formulation method. The major goal of the suggested technique is to use as little time as possible for overall scheduling, response time, maximum load balancing, maximum resource usage, low latency, and lowest MakeSpan.

Table 2. List of symbols and notations

Symbol	Description
ʃg̱	Fog Node
ŔT	Response Time
ύӍ	Virtual Machine
ρ_t	Time of Propagation
ɋ	Execution time
P	Latency
ϼT	Time of Processing
T_{ι^e}	Time of Task execution
CͭꞰ	Completion time
$\mathcal{S}\tau$	Scheduling Time
ßș	Number of Balanced Fog nodes
Ȯļ	Number of Overloaded Fog nodes
Ƒșs	Number of all available Fog nodes
Dc	Data Center

The list of symbols and notations used for problem formulation is presented in Table 2. Suppose the fog layer has N fog nodes to handle requests for fog.

Fog Nodes= {ʃg̱1, ʃg̱2,….,ʃg̱N }

The scenario has a variety of physical machines. Consider all the physical Machine with the ÞӍs = {ÞӍ1, ÞӍ2, ÞӍ3,...,ÞӍN}. There are N total physical machines in existence scenario. The ÞӍ is connected to a no. of Virtual machines. We are treating the fog nodes in this scenario like a virtual machine.

ύӍ = {ύӍ1, ύӍ2, ύӍ3,...,ύӍN}

Each Virtual Machine has resources such as Central processing unit, RAM, Network Bandwidth and storage. The Fog layer contains various fog nodes to manage consumer requests. A fog node has maximum requests from patients and it is declared as per Equation 1.

$$\acute{R}\text{ Total} = \sum_{p=1}^{n}\left(\acute{R}_p\right) \quad (1)$$

By distributing requests, a load balancer evenly distributes the load among all of them. The x tasks are carried out concurrently by the fog node. Each carries out the work on its own and manages its own resources.

4.1. MakeSpan

The duration of time needed to accomplish a task as intended or in its entirety is known as the (CƮ) Completion time. The time at which a process ends its execution is known as the completion time. Low Makespan should to be necessary. MakeSpan is the amount of time a task can take to complete. The goal is to decrease makeSpan of requests for load balancing. The makeSpan of r tasks on ʋϻi is explained in Equation 2.
$\text{MakeSpan}_r = \text{Max}\,(CƮ_{r,i})$ (2)

Mapping of $Ʈ_{task}$to ʋϻ affects performance parameters. Where ʋϻ = {1,2,3,….,s,….y}, s Є ʋϻ, $Ʈ_{task}$ = { 1,2,3,. . . , r, . . . , x), r Є$Ʈ_{task}$,. Now, the load balancing algorithm and end user requests determine how many jobs are allotted to each ʋϻ.

Utilizing linear programming, the tasks' processing and reaction times are expressed. The processing time T of the assigned tasks from r to s is r,s.

4.2. Processing Time

The task's status is $\omega_{r,s.}$ and the processing timeꝩTof assigned tasks from $Ʈ_r$to s isꝩ$T_{r,s.}$ The ꝩ$T_{x,y}$ tasks from x taks are assigned to y ʋϻs and status $\omega_{r,s.}$

$$\omega = \begin{cases} Iftaskxisassigned, 1\,, \\ otherwise, 0. \end{cases}$$

The goal, as stated in Equation 3, is to reduce

Processing Time,

$$ꝩT = \sum_{i=1}^{x}\sum_{j=1}^{y} ꝩT_{x,y} \times \omega_{x,y} \quad (3)$$

Where, $ꝩT_{x,y} = \frac{ƮL(x)}{Ps(y)}$

4.3. Reponse time

It is the time duration a fog resource needs to finish a specific operation. The specified node will be occupied during this time and will not carry out any further tasks. It is computed with the Equation 4.

$$R\mathrm{Ʈ}_{x,y} = \frac{\sum y \in \mathrm{ύµs}\ (CTx)}{MakeSpan \times Number\ of\ \mathrm{ύµs}} \quad (4)$$

Where, $R\mathrm{Ʈ}_{x,y}$ serves as a representation of the system's total RƮ of the ύµs. $C\mathrm{Ʈ}_x$ indicates the job completion time.

4.4 Latency

It is described as the total amount of time the system took to reply to the data from the receiver. It combines the propagation time with the processing time. Here, ρ_t stands for the amount of time it took for the data to get to the cloud or fog layer and ɋ denotes execution time.

Latency (ρ) is calculated as below Equation 5,

$$\rho = \rho_t + \mathrm{ɋ} \quad (5)$$

4.5 Scheduling Time

It effectively schedules them for incoming requests and allots time for them to complete their computational resources. The act of scheduling involves grouping incoming requests into certain time slots in order to maximise the use of the available resources. According to Equation 6, the scheduler must take into account a variety of factors, such as the job's nature, size, execution time, resource availability, task queue, and resource load at the moment of scheduling, in addition to other factors. $\mathcal{R}$ is the complete utilization of all available resources. High resource utilisation ratios are necessary for improved performance. It is determined by task execution timeT_{i^e} and $\mathcal{R}$.

$$\mathcal{S}\tau = \mathrm{T} + T_{i^e} + \mathcal{R} \quad (6)$$

4.2.6 Resource Utilization

It is the arrangement of all the available resources for complete utilization. High resource utilisation ratios are necessary for improved performance. It is calculated using the below Equation 7.

$$\text{Resource Utilization} = \frac{\text{ßʂ}+\text{Ộƚ}}{\text{ʄʂs}} \times 100\ \% \quad (7)$$

4.2.7 Load Balancing Level

To perform better, a high load balancing level should be necessary. A load balancing level is determined using the equation below Equation 8.

$$\text{Load balancing level} = \frac{\text{ßʂ}}{\text{ʄʂs}} \times 100\% \quad (8)$$

Where, the number of FNs that are balanced is denoted by ßʂ and the number of all ʄʂs that are available is denoted by 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 ← currEstimatedResponseTime(dc) <
currestimatedResponsetime ? 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.vm.allocations < MAX (selectedDC.vms.allocations)
23.                            allocate(selectedDC.vm)
24.                 end if
25.           end for
26.     end if
```

The primary objectives of DynaReLoad are to locate the closest data centre among all those that are available and to identify good quality of service parameters, such as the quickest response time, the shortest scheduling time, the best resource utilisation, the best load balancing level, and the lowest latency. When there are multiple nearby data centres, a data centre is selected at random from a list of closeness. When the response time of the closest data centre is deteriorating, it will initially look for or discover that data centre. It will be designated as a quickset data centre because of its faster reaction time. The nearest data centre is chosen as the destination data centre after choosing the fastest data centre. The data centre will be chosen at random if the closest data centre and the fastest data centre are not comparable. Initially the null values are assigned to the current estimation response time and the data centre current estimation response time. Finding the nearest data centre from among those that are available is the first objective. Do the quickest response time calculation now using a ternary operator.
It will then check to see if the closest DC is the same as the current estimated response time. The chosen DC is then assigned, and the data centre is located. Suppose it determines that data centre 1 is the closest. The following step is to choose Vm. The following section shows the application scenario and architecture based on the requirements.

5. Proposed Architecture And Approach

This section explains proposed architecture of fog computing and approach. Figure 1 shows the proposed architecture. The approach, together with load balancing to solve platform-level issues in fog computing, would be appropriate for the pandemic situation since it would help resolve the latency issues and guarantee that the processing is as rapid and efficient as feasible. Layered representation is the best way to describe fog architecture. Many analysers have analysed different fog architectures for process implementation i.e. three [22], four [11] and five [4].

5.1 Components of Fog Computing Architecture

The proposed framework DynaReLoad contains three-layer fog architecture for implementation purpose.

5.1.1 End- User Layer

All of the components that potentially serve as the fog computing system's primary data source are found at the bottom layer. These would gather all necessary physical data and include blood pressure monitoring sensors, Oxygen monitoring sensors, ECG sensors, pulse monitoring sensors etc. The physical layer also includes virtual sensors, which can be used to assess a process's conditions based on readings from the physical sensors and take prompt action as required. The sensors in the proposed framework collect vital signs information from the patients.

5.1.2 Fog Layer

Fog devices, Fog servers, dynamic load balancer and gateway make up fog layer. The virtual sensors, actual sensors, and fog server would all be connected to the fog devices. Every fog device linked to the same server will have access to one another when necessary. The various applications computational needs are also handled by this layer. Any compute load would be dispersed evenly among all of the nodes by the dynamic load balancer, preventing any node from becoming overloaded.

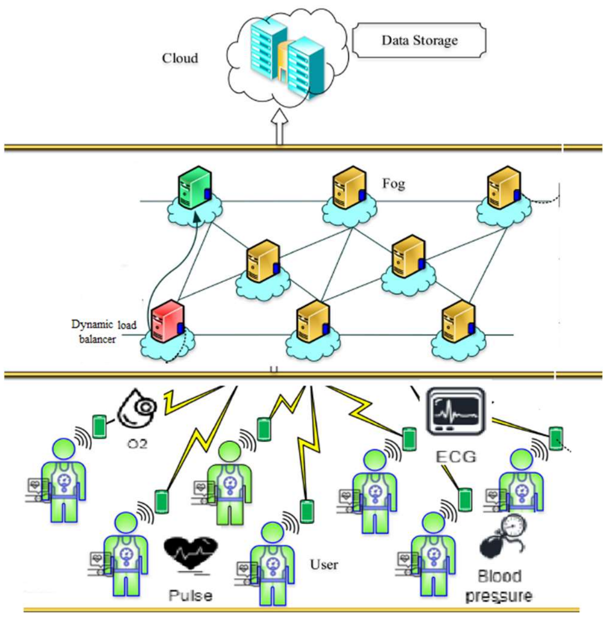

Figure 1. DynaReLoad architecture for the Covid-19 pandemic situation critical emergency patient vital sign monitoring system

5.1.3 Cloud Layer

The fog nodes send data to the cloud layer that needs to be processed and stored for a long time. The data flow component does not decide whether data should be kept locally in fog layer or for long term in the cloud after processing in the fog layer. Using edge processing to reduce data volume is the key problem. Based on the architecture, the proposed application scenario and algorithm is prepared in the next section.

5.2 Proposed Application scenario and proposed Algorithm (DynaReLoad)

Healthcare systems see large numbers of patients and chronic disorders as a significant factor. In the majority of hospitals, biometric parameters values are measured manually. The manual process wastes a lot of time. Through the use of automation, we can cut back on both expenses and processing time. The effectiveness and quality attained by the incorporation of dynamic load balancing and fog computing in healthcare systems. Imagine cloud computing in the context of healthcare. If all of the patients' storage and computing needs are handled by a single, central cloud server, it will lead to a number of issues, including traffic congestion, significant end-to-end delays, and high network utilisation.

To tackle these problems, fog nodes are being employed in a geographically dispersed manner. If we conduct several tasks on a single fog node, congestion will result. We have tested scenarios where one machine or server cannot manage all of the patient admission and discharge information in the multispecialty hospital. One node cannot fulfil all of the requests.

To create a system for tracking an emergency patient's vital signs as a result of the COVID-19 pandemic and other serious illnesses, the proposed algorithm and framework (DynaReLoad) is used.

It is almost hard for hospitals to accommodate all patients since the number of impacted people is growing quickly. A large number of patients receive delayed care as a result, which raises the death rate. The suggested framework will send a prompt notice to the hospital and clinicians in case of any irregularities, preventing any delay in treatment.
For a multi-story, multi-specialty hospital, acquiring and processing data presents huge difficulties. For instance, if there are 1000 patients and each patient attached with 4 to 5 sensors, it is possible to calculate the number of requests per minute for each patient as follows:
1 min equals to 60 seconds, Each sensor will provide data every second, therefore the total amount of data collected in a minute will be 240000 requests. The calculation is done as = $60 \times 1000 \times 4 = 240000$.

To handle efficiently, we assign every 200 patients one fog node. Requests are therefore spread to fog nodes by restricting the amount of requests that touch a cloud. In that instance, for the prevention of flooding requests to one node, the load balancing technique is used as a solution. Numerous quality-of-service requirements must be met, including low latency, quick execution, quick response, quick request service time by the data centre, cost-effectiveness, to guarantee that a prompt response may be delivered.

A patient with COVID-19 or another serious illness is isolated in a critical care unit, which is outfitted with sensors to track the patient's vital signs, as shown in Figure 2. Most affected individuals experience significant interior symptoms in addition to modest exterior symptoms including sneezing, coughing, shortness of breath, throat soreness, fatigue, headaches, and fever. These internal vitals will be monitored by the sensors using an oximeter, blood pressure and pulse oximeter, and an ECG. It is crucial to keep an eye on these symptoms because they could increase or decrease a patient's risk. Many patients may appear healthy and have no visible symptoms, but they could nevertheless have bad oxygen levels causes issues related to lung. The doctors will be informed if a patient's oxygen level falls below what is considered normal (95 percent, according to WHO norms). Similar to how an adult should, children and adolescents should have pulse rates between 70 and 100 beats per minute. (The formula beats per minute = 220 - age can be used to determine a person's maximal heart rate.)

An alarm will be sent to the concerned person if the pulse rate is found to be lower than anticipated. The primary determining factor for cardiovascular disorders is the troponin level, and the ECG sensor would function similarly to identify any irregularity discovered in the troponin level. In order to determine the present condition of the admitted patient and will intimate status record to doctor outside ICU.

This would enable the medical professionals to check on the patient in a secure setting with less need of periodic physical diagnosis. Before a patient is quarantined to the ICU, their medical history will also be entered into the system, which will be useful for analysing their health status. The fog nodes receive data from the collected sensors and then the dynamic load balancer deliver updates of remote monitoring after processing data on several fog nodes.

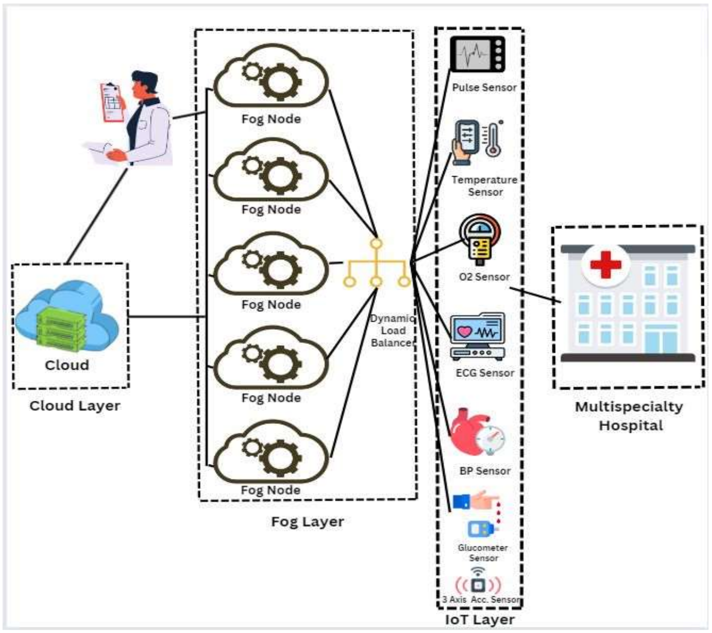

Figure 2. Proposed system Framework for multispeciality hospital

In accordance with the framework, the multispecialty hospital is divided into smaller cells, each of which contains a number of fog nodes. The fog nodes would be chosen at random by a dynamic load balancer. If the load balancer is positioned above every sensor, by dividing the hospital into several clusters, these problems of time wastage, resource scarcity, and transmission time can be resolved.
In the hypothetical situation, the hospital receives numerous fog nodes for every 200 patients. The associated custom load balancer receives the initial transmission of the vital sign sensors' accumulated data. Each floor-wise fog node with a designated function has a balancer attached. The nodes are linked to the cloud, where further processing is required in the fog computing. The doctors will receive a prompt alert if any irregularity is found. As a result, the sufferers would be able to access medical care quickly [14], avoiding the virus's early demise. The results of the simulation are shown in the following section.

6. Simulation Setup

Three alternative implementations of the suggested framework and algorithm are used: fog-based, cloud-based, and dynamic load balancing fog-based. Here, various scenarios are analysed and evaluated with the proposed application simulations. Blood pressure, ECG, oxygen level, and pulse are all detected using four different sensors. The data and signals that were gathered are sent to the fog nodes. The received data is subsequently processed once more by fog nodes to determine the state. Our application scenarios have been evaluated in terms of latency, resource usage, and response time using iFogSim simulation software. The experimental setting parameters and the setup of the fog nodes are shown in Table 3.

Table 3. Setting of Parameters for experimental purpose

Number of users	65/min
Number of sensors per user	Covid patients- Four sensors Emergency patients: According to requirement, it may vary.
Fog nodes	Every 200 patients assign to one fog node
Storage cost	0.1
Fog Node configuration	
Bandwidth	1000 MBs
Storage	1GB
Memory Cost	0.05
Resource Cost	1.0

In order to analyse dynamic load balancing using Fog-based architecture in accordance with the definitions in Algorithm 1 and the proposed framework represented in Figure 3.
Latency, network use, reaction time, and other metrics are being monitored. Sensors are installed in each of the produced fog nodes to measure the patient's vital signs. Virtual machines located at the fog nodes process the data produced by the sensors. To analyse data quickly, a dynamic load balancing method is employed.

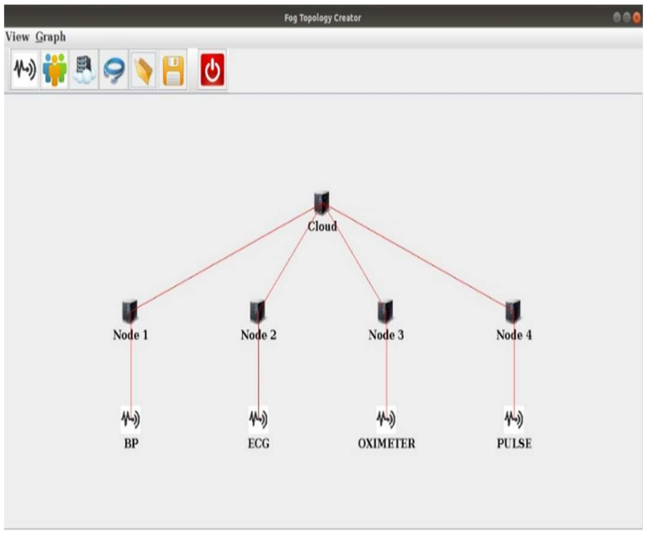

Figure 3. The topology of the vital sign monitor for an emergency patient

The parameters for evaluating dynamic load balancing fog based implementation in iFogSim include the CPU's computing power measured in MIPS, the uplink's bandwidth, the architecture level, the RAM, Processing speed, power status and downlink bandwidth.
Importing the topology into already-created iFogsim will allow you to execute the final simulations.
We will contrast dynamic load balancing, execution based on cloud and Fog with the nearby data centre. The fog nodes are part of the topology and are connected to a load balancer through a network. Onto the underlying fog nodes, this load balancer distributes the traffic.

7. Result Analysis comparison using QoS Model

We have referred to one of base research papers, where fog computing is divided into separate clusters [20]. In this scenario, my request goes to cluster head only. Cluster head manages all the requests sequentially. First request moves to the first cluster head then it will check the resource availability by request accept or reject signal. Let's assume that the first cluster is full and if still a request goes there. It will not be able to give resources and request is forwarded to another cluster. So, it leads to problems like waste of time, reduced chance of service migrate on and reduced time of data transmission. To overcome these drawbacks we have placed a load balancer on top of all the sensors. Second base paper is based on Priority based scheduling [24] Here the research gap is because lower priority activities could starve and get put off forever if high priority processes take up a lot of CPU time. Starvation is the state in which a procedure is never scheduled to run. Choosing which process receives which priority level is a different issue. Numerous algorithms have been developed with different requirements and limitations. Based on the comparison of the literature survey on load balancing algorithms, the loopholes have been identified and finally compared proposed approach with Round robin[16], Throttled[16], Active monitoring and DRAM[20] for implementation purpose. Next subsection also illustrations of load balancing algorithms comparison after the cloud based and fog based approach comparison.

7.1 Latency

The cloud-based design considerably increases latency with an increase in the number of sensors. Cloud latency increases since the cloud server needs to handle all of the sensor queries. In a fog-based arrangement, the fog nodes only process the detected data from the sensors that are connected to them. Here, from Figure 4, the approach has achieved latency of 500.23ms for cloud based configuration, 39.57ms for fog based configuration and 19.29ms for Dynamic load balancing based Fog configuration. The proposed approach is 96% and 51% superior to cloud and fog based configuration for average-value.

With more sensors, there is a significant increase of latency in cloud as it needs to handle all the requests. Fog, on the other hand, processes just the data sensed by the sensors attached to it, which reduces latency. This is known as a dynamic load balancing solution. The following chart compares load balancing algorithms with the suggested strategy.

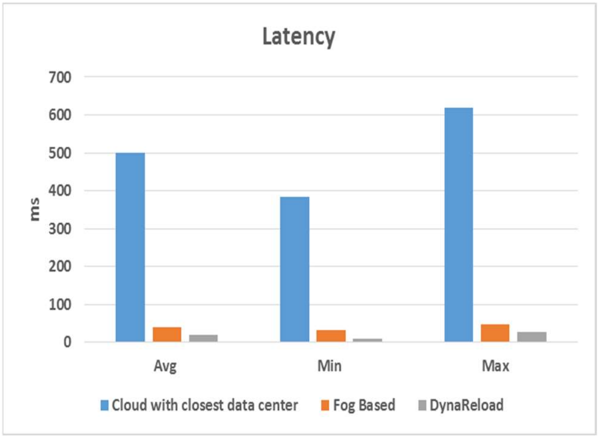

Figure 4. Latency comparison for different configuration

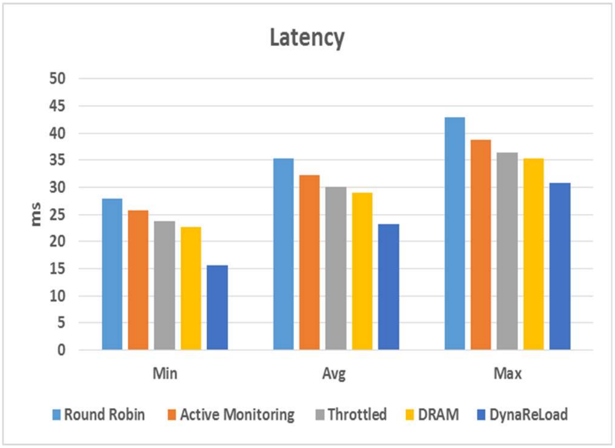

Figure 5. Comparison of Load balancing Algorithms

Dynamic load balancing-based Fog scenario provides 34%, 28%, 22%, 20% superior latency results as compared to other load balancing algorithms. Similarly, Figure 5 compares load balancing techniques. According to the findings, the DynaReload technique has a lower latency as compared to other load balancing algorithms.

7.2 Response Time

Simply explained, response time is the amount of time it takes a fog resource to accomplish a task. Figure 6 shows the response time of vital signs monitoring sensors. The proposed approach is 96% and 50% superior to cloud and fog based configuration for average-value.

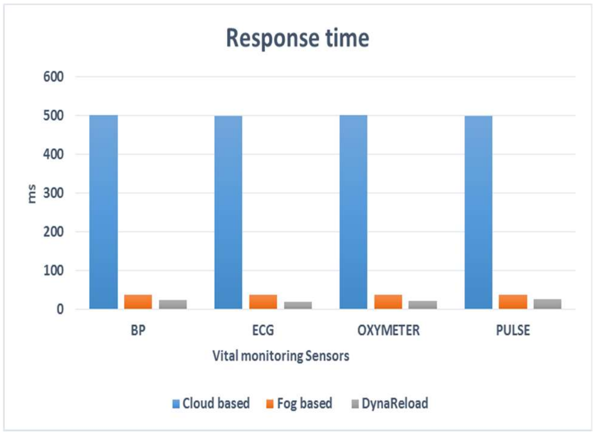

Figure 6. Response time of vital sign monitoring sensors

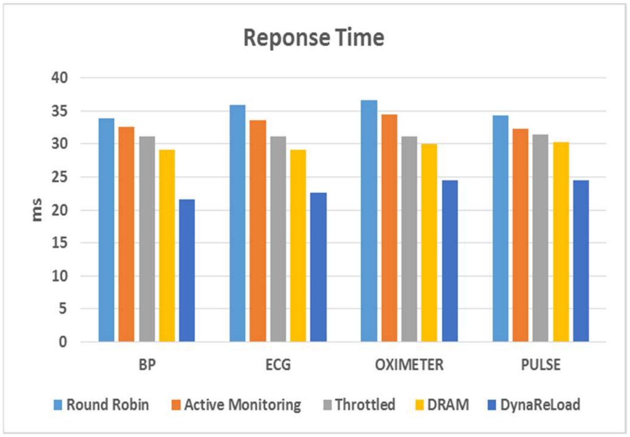

Figure 7. Response time comparison of load balancing algorithms

Figure 7 depicts the response time by various vital signs monitoring sensors for round-robin, active monitoring, throttled, DRAM-based, and dynamic load balancing-based architecture. Here, DynaReload approach provides 36%, 34%, 31% and 26% better response time as compared to RR, Active monitoring, throttled and DRAM load balancing algorithms.

7.3 Scheduling Time

The average, lowest, and maximum scheduling times for the three methods are shown in Figure 8. In comparison to fog with nearest data centre architecture and cloud-based design, the dynamic load balancing approach requires less scheduling time. The scheduling time of various load balancing strategies is shown in Figure 9. The proposed approach is 95% and 48% superior to cloud and fog based configuration for average-value.

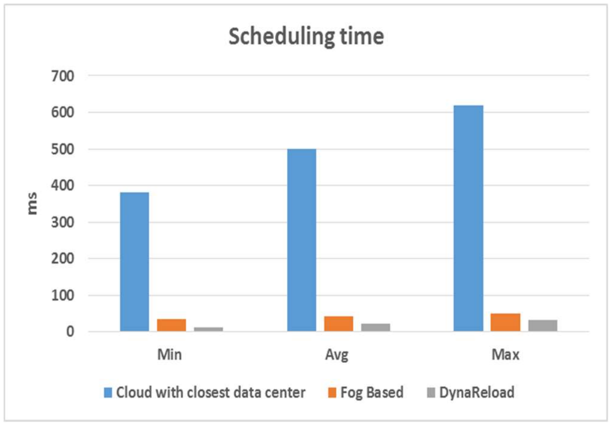

Figure 8. Comparison of Scheduling Time for different configuration

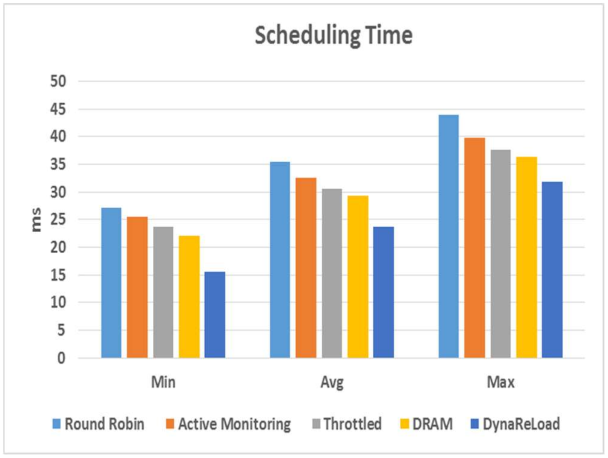

Figure 9. Comparison of Scheduling time for load balancing algorithm

In this case, the DynaReload method processes data more quickly than the Round-robin, Throttled, Active monitoring, and DRAM algorithms during scheduling. Dynamic load balancing-based Fog scenario provides 33%, 27%, 22%, 19% better scheduling results as compared to RR, AM, Throttled and DRAM.

7.4 Make Span, Resource Utilization and Load balancing level

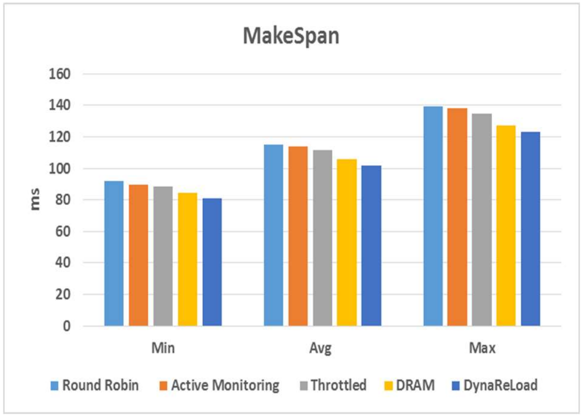

Figure 10. Comparison of Makespan for different load balancing algorithms

As compared to other load balancing algorithms, the DynaReload technique offers a smaller MakeSpan. Dynamic load balancing-based Fog scenario provides 12%, 10%, 9%, 4% superior makespan results as compared to RR, AM, Throttled and DRAM. As seen in Figure 10, DynaReload also offers quicker makespan times than other load balancing techniques.

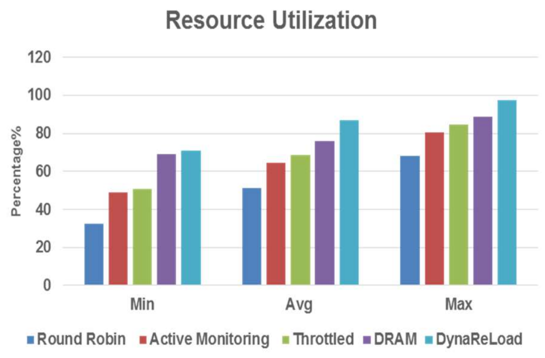

Figure 11. Comparison of Resource Utilization for different load balancing algorithms

Figures 11 and 12 above explained that the DynaReload method outperforms the RR, Active Monitoring, Throttled, and DRAM algorithms in terms of performance. Dynamic load balancing-based Fog scenario provides 69%, 35%, 27%, 2% superior resource utilization results as compared to other load balancing algorithms as per Figure 11. As per Figure 12, Dynamic load balancing-based Fog scenario provides 51%, 28%, 27%, 3% better load balancing results as compared to RR, AM, TH and DRAM. Here, DynaReload algorithm delivers higher Resource utilization and higher load balancing level.

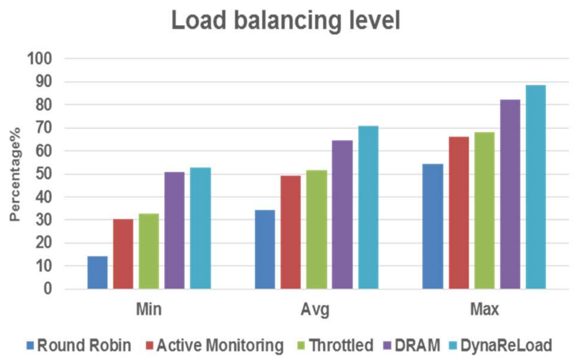

Figure 12. Comparison of Load balancing level of load balancing algorithms

The provided data compares scheduling time, latency, response time, resource utilisation, and load balancing level for each of the five methods. The results help us understand that the largest improvement can be made if DynaReload is used for the specific case, allowing for quick patient care.

The effectiveness of the suggested architecture for the execution of emergency patient's vital sign monitoring system has been confirmed by simulation results carried out at different scales. The next section summarises the study and establishes the parameters for subsequent research.

8. Conclusion

In order to ensure that every patient receives medical care, a demand for remote health monitoring systems has emerged due to the rising number of illnesses and pandemics in society. There are a few cloud-based remote healthcare solutions on the market, but they present significant challenges due to their high latency, decreased privacy, increased energy usage, and weaker data security. In order to overcome these challenges, fog computing is considered as an alternative. It works as a middle-ware by moving computing resources and application services closer to the edge where the data is being generated.

The research's goal is to assign fog servers between end users and the cloud to efficiently use resources with the least amount of delay. An evenly distributed load across all servers would prevent overloaded servers from failing. This is accomplished by evenly distributing the load at the fog layer, which shifts the load from the overloaded server to a neighbouring idle one.

Each sort of computer node in the cloud and fog can have its load balanced dynamically thanks to the DynaReload. Comparing the DynaReload strategy to the cloud-based approach and the conventional fog-based approach, better results are obtained in terms of response time, scheduling time and latency. Again, compared to other load balancing algorithms, it provides good quality of service parameters outcomes in terms of response time, latency, Makespan, scheduling time and load balancing level. The proposed approach is additionally created for multispecialty hospitals. The authors are eager to create a system in the future that may prescribe medications after analysing sensor results in order to improve a patient's general health.

References

1. Mahmud, R.; Buyya, R. Fog computing: A taxonomy, survey and future directions. Internet of Everything, Springer 2017.

2. Gao, L.; Luan, T.; Yu, S.; Zhou, W., Liu, B. Fogroute: Dtn-based data dissemination model in fog computing. IEEE Internet of Things Journal, 2017, Volume. 4, pp. 225–235.

3. Hu, P.; Dhelim,S.; Ning, H.; Qiu, T. Survey on fog computing: architecture, key technologies, applications and open issues, Journal of Network and Computer Applications, 2017.

4. Varshney, P.; Simmhan, Y. Demystifying fog computing: Characterizing architectures applications and abstractions, In Fog and Edge Computing (ICFEC), IEEE 1st International Conference on. IEEE, 2017, pp. 115–124.

5. Perera, C.; Qin, Y.; Estrella,J. C.; Reiff-Marganiec,S.; Vasilakos, A. Fog computing for sustainable smart cities: A survey, ACM Computing Surveys (CSUR), 2017, Volume. 50, pp. 32.

6. Bhavsar, S.; Modi, K. Design and Development of Framework for Platform Level Issues in Fog Computing, International Journal of Electronics, Communications, and Measurement Engineering (IJECME), 2019, Volume.8, pp. 1-20.

7. Kai, K.; Cong, W.; Tao, L. Fog computing for vehicular Ad-hoc networks: paradigms, scenarios, and issues, J. China Univ. Posts Telecom., 2016, Volume. 23, pp. 56–96.

8. Peng, M.; Yan, S.; Zhang, K.; Wang, C. Fog-computing-based radio access networks: issues and challenges, IEEE Netw., Vol. 30, No. 4, pp. 46–53, 2016.

9. Dasgupta, A.; Gill, A. Fog Computing Challenges: A Systematic Review, Australasian Conference on Information Systems, 2017.
10. 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), 27th International. IEEE, 2017, pp. 1–6.
11. Mit, H. R.; Diyanat, A.; Pourkhalili, A. Mist: Fog-based data analytics scheme with cost-efficient resource provisioning for iot crowdsensing applications, Journal of Network and Computer Applications, Volume. 82, 2017, pp. 152–165.
12. Mahmud, R.; Koch, F. L.; Buyya, R. Cloud-fog interoperability in iot-enabled healthcare solutions, In Proceedings of the 19th International Conference on Distributed.
13. Shah, Y.; Thakkar, E.; Bhavsar, S. Fault Tolerance in Cloud and Fog Computing—A Holistic View, In Data Science and Intelligent Applications, Springer, Singapore, pp. 415-422, 2021.
14. Bhavsar, S.; Pandit, B.; Modi, K. Social Internet of Things, In Integrating the Internet of Things Into Software Engineering Practices, pp. 199-218, IGI Global, 2019.
15. Rangras, J.; Bhavsar, S. Design of Framework for Disaster Recovery in Cloud Computing, In Data Science and Intelligent Applications, pp. 439-449, Springer, Singapore, 2021.
16. Bukhsh, R.; Javaid, N.; Ali Khan, Z.; Ishmanov, F.; Afzal, M.; Wadud, Z. Towards fast response, reduced processing and balanced load in fog-based data-driven smart grid. Energies, pp. 3345, 2018
17. Wan, J.; Chen, B.; Wang, S. Fog computing for energy-aware load balancing and scheduling in smart factory. IEEE Transactions on Industrial Informatics, pp. 4548-4556, 2018.
18. Al-Khafajiy, M.; Baker, T.; Chalmers, C.; Asim, M.; Kolivand, H.; Fahim, M.; Waraich, A. Remote health monitoring of elderly through wearable sensors. Multimedia Tools and Applications, 78(17), pp. 24681-24706, 2019.
19. Babou, C.; Fall, D.; Kashihara, S.; Taenaka, Y.; Bhuyan, I.; Niang, Y.; Kadobayashi, Hierarchical load balancing and clustering technique for home edge computing, IEEE Access, 8, pp. 127593-127607,2020.
20. Xu. X.; Fu, S.; Cai, Q.; Tian, W.; Liu, W.; Dou, W.; Liu, A. Dynamic resource allocation for load balancing in fog environment, Wireless Communications and Mobile Computing, 2018.
21. Hassan, S.; Ahmad, I.; Ahmad, S.; Alfaify, A.; Shafiq, M. Remote pain monitoring using fog computing for e-healthcare: An efficient architecture. Sensors, pp. 6574, 2020.
22. Aladwani, T. Scheduling IoT Healthcare Tasks in Fog Computing Based on their Importance, Procedia Computer Science, pp. 560-569, 2019.
23. Velásquez, W.; Munoz-Arcentales, A.; Salvachúa, A. Fast-data architecture proposal to alert people in emergency, In 2018 IEEE 8th Annual Computing and Communication Workshop and Conference (CCWC),pp. 165-168, IEEE, 2018.
24. Choudhari, T.; Moh, M. Prioritized task scheduling in fog computing, In Proceedings of the ACMSE 2018 Conference, pp. 1-8, 2018.
25. Fan, Q. ; Ansari, N. Towards workload balancing in fog computing empowered IoT, IEEE Transactions on Network Science and Engineering, 7(1), pp.253-262, 2018.

26. Shukla, S.; Hassan, M.; Khan, M.; Jung, L.; Awang, A. An analytical model to minimize the latency in healthcare internet-of-things in fog computing environment, 2019.