Chapter 4
Distributed Computing in Fog Computing

IoT and Cloud Computing based health care applications suffer from slower decision-making due to network delays, less availability of bandwidth, transmission delays, processing delays, and data de-noising. To avoid this, Fog Computing can be applied as a middle layer between the IoT and Cloud layers. Fog Computing greatly reduces transmission delays, but Fog devices lag in computing capabilities due to their limited processing power. This can be solved by deploying multiple devices and synchronizing their computation to enable parallel execution. In this chapter, a single Raspberry Pi is used as a Fog Node and multiple such Raspberry Pis are deployed in the cluster form. The Dispy Python framework is used to allow parallel processing. Scalability is easy to achieve with Dispy, and also it provides different features like automated node discovery, job distribution, processing function distribution, remote access, and enhanced security. The system is proposed and implemented to test the hypothesis that it improves the real-time computation in health care applications. The final results are compared with a traditional processing system and it is found that the Raspberry PI cluster, and Dispy can enhance the computation performance in Health Care.

4.1 Distributed Computing and Dispy

Distributed Systems are a group of independent computers that connect to offer various services. They share and store data without physically sharing memory or processor with other computers. These machines have a shared state, operate concurrently and can fail independently without affecting the whole system's uptime. A distributed computer system comprises multiple software components that run on multiple computers but act as a single machine. A distributed system allows reduced computational time over massive datasets [93]. Distributed systems are classified into two major categories: centralized (client-server) and decentralized (peer to peer) system. To facilitate Distributed Computing, dispy is a very good tool. The dispy tool is developed in python and it is available for different needs and greed. The dispy tool [94], is available for different Distributed Computing tasks.

4.1.1 Why dispy?

To implement the Distributed Computing for Raspberry Pi cluster in Fog Computing "dispy" is selected. Dispy is developed in python and python works very well with Raspbian operating systems. The main problem faced in deployments of Distributed Computing is that each slave node has to be configured for a particular application context. If more slaves add-up then it needs more configuration. This makes scalability in Distributed Computing a bit difficult, but in case of dispy only master node has to be configured and on all slave nodes, only dispy should be installed. No need to configure every slave. This makes a distributed system more scalable if we use dispy. In Fog Computing, Distributed Computing is achieved by using the cluster of Raspberry-Pi nodes.

4.2 Raspberry Pi and R-Pi cluster

Raspberry Pi was introduced by the Raspberry Pi Foundation and is a series of small single-board computers [95]. Although they were intended to be used for teaching the basics of computer science, now they have surpassed it. Today it can be used for everything from Home Entertainment System to Cryptocurrency Mining [96-97]. A Raspberry Pi cluster is a collection of Raspberry Pi, all running as a single computer using parallel computing [98]. Raspberry Pi Clusters use an architecture involving a Master node and Slave node, the master node is responsible for giving instructions for the tasks that need to be performed [99-100].

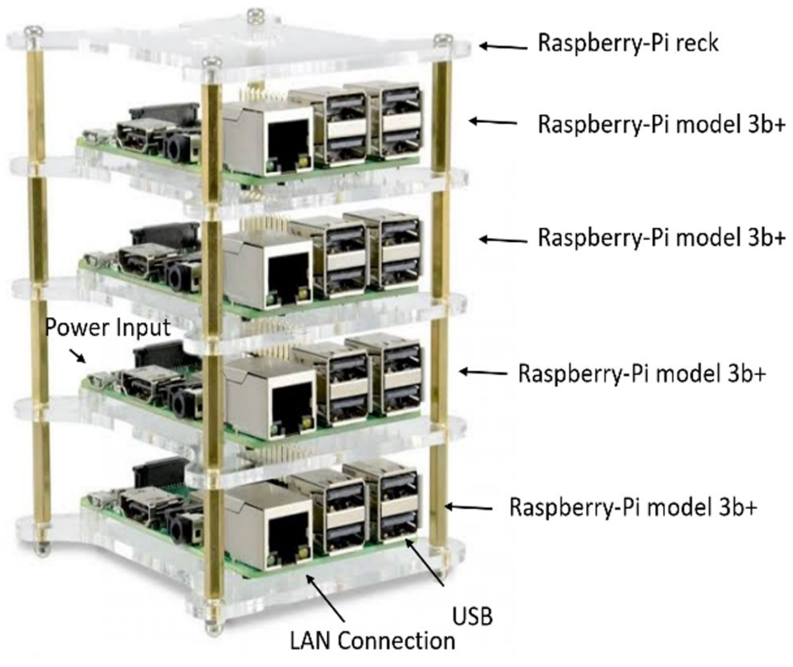


Figure 4.1 Raspberry Pi Cluster used in the system

Experiment and research conducted in the field of Cloud Computing have shown that it is possible to run edge computing on these Raspberry Pi clusters [99, 101]. These clusters can have a variety of different applications from running as MQTT Broker [100], Fog Computing framework for Wearable IoT Devices [102] to teaching Distributed Computing [103]. The presence of multiple Raspberry Pi nodes in the system improves the reliability of the system.

4.3 Computing Job description in dispy

The ECG wave is made up of different intervals like PR, ST, QRS, and QT, shown below. The real-time signal is detected and digitized using the methods specified in so that it can be processed in the computers. After digitizing, different intervals are found using the windowing algorithm [74]. After these intervals are calculated, these are compared with standard chart, which allows us to decide whether the given ECG signal is normal or abnormal in nature.

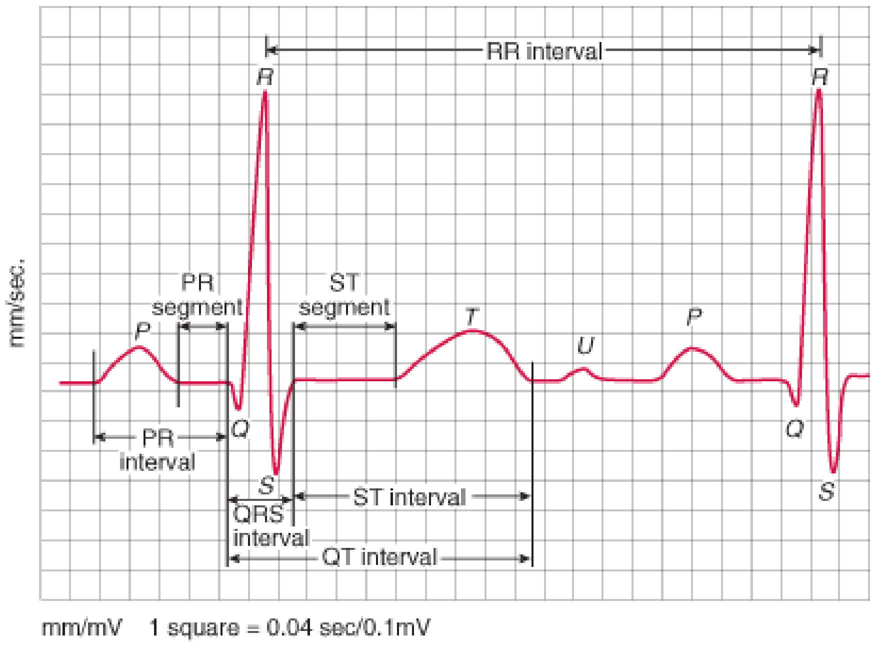


Figure 4.2 ECG wave with PQRSTU reference points and different intervals

Here, the ECG waveform R-R is considered as one wave. Datasets of 1000, 2000, 3000, 4000, and 5000 waves are passed on for processing. These data streams also contain abnormal waves. The set of waves are processed until the last wave to find the normality and abnormality. The total time is measured as “Job Starting time to Job ending time” in milliseconds.

4.4 Implementing and Analyzing ECG waves in dispy

We have implemented the ECG analyzing system with Dispy, with the head node dividing the data into multiple jobs. The jobs are given to the slave nodes one by one. The Dispy system working flowchart is given below. The Dispy master node will first find the available nodes and make the connection. While doing the setup as a Dispy slave node, the encryption keys need to be given to ensure secure communication. Once the Dispy clients and servers are ready, then the function to analyze the ECG signal is to be given along with ECG waves. Dispy server node will send the data and the function to the slave nodes to perform the analyzing task.

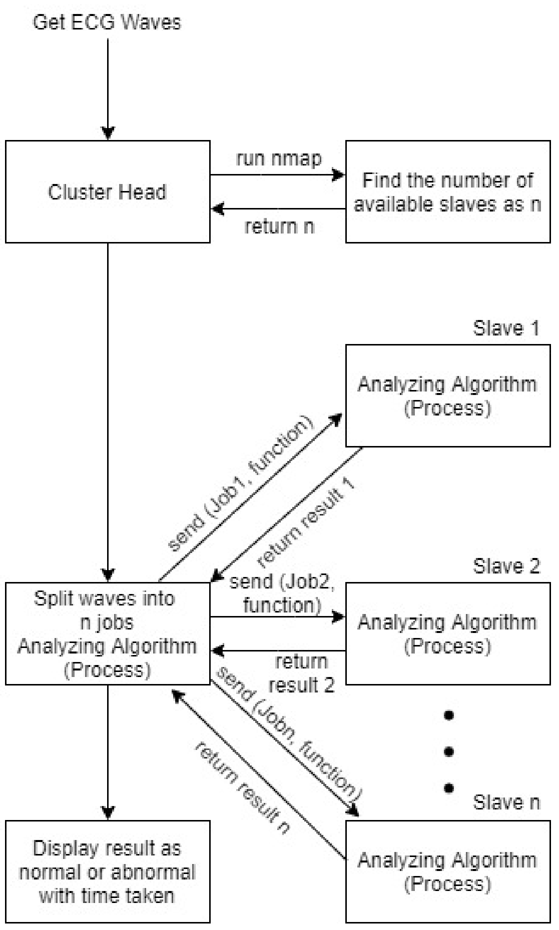
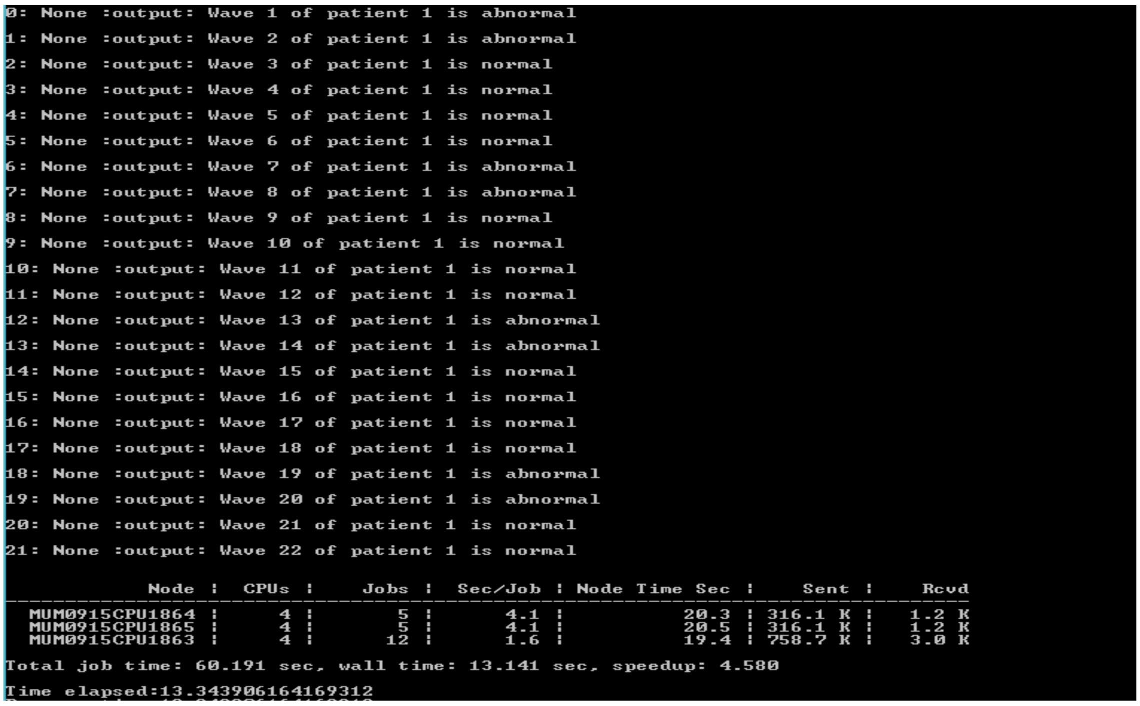

Get ECG Waves
Cluster Head
run nmap
Find the number of available slaves as n
return n
Slave 1
Analyzing Algorithm (Process)
send (Job1, function)
return result 1
Slave 2
Split waves into n jobs Analyzing Algorithm (Process)
send (Job2, function)
Analyzing Algorithm (Process)
return result 2
send (Jobn, function)
return result n
Slave n
Display result as normal or abnormal with time taken
Analyzing Algorithm (Process)

Figure 4.3 Dispy: working mechanism

4.4.1 Analysis of ECG Waves using Dispy API

The series of ECG waves are given to Dispy to process each wave in terms of its normality and abnormality. The figure 4.4, shows the Dispy system output where it indicates the total number of waves present, the nature of each wave detected as normal or abnormal, number of nodes present, number of cores present in each node, number of jobs processed by each node, data size sent and received, and time is taken or the wall time to process the job.



```
0: None :output: Wave 1 of patient 1 is abnormal
1: None :output: Wave 2 of patient 1 is abnormal
2: None :output: Wave 3 of patient 1 is normal
3: None :output: Wave 4 of patient 1 is normal
4: None :output: Wave 5 of patient 1 is normal
5: None :output: Wave 6 of patient 1 is normal
6: None :output: Wave 7 of patient 1 is abnormal
7: None :output: Wave 8 of patient 1 is abnormal
8: None :output: Wave 9 of patient 1 is normal
9: None :output: Wave 10 of patient 1 is normal
10: None :output: Wave 11 of patient 1 is normal
11: None :output: Wave 12 of patient 1 is normal
12: None :output: Wave 13 of patient 1 is abnormal
13: None :output: Wave 14 of patient 1 is abnormal
14: None :output: Wave 15 of patient 1 is normal
15: None :output: Wave 16 of patient 1 is normal
16: None :output: Wave 17 of patient 1 is normal
17: None :output: Wave 18 of patient 1 is normal
18: None :output: Wave 19 of patient 1 is abnormal
19: None :output: Wave 20 of patient 1 is abnormal
20: None :output: Wave 21 of patient 1 is normal
21: None :output: Wave 22 of patient 1 is normal

           Node |  CPUs |     Jobs |  Sec/Job | Node Time Sec |     Sent |     Rcvd
------------------------------------------------------------------------------------
  MUM0915CPU1864 |     4 |        5 |      4.1 |           20.3 | 316.1 K |    1.2 K
  MUM0915CPU1865 |     4 |        5 |      4.1 |           20.5 | 316.1 K |    1.2 K
  MUM0915CPU1863 |     4 |       12 |      1.6 |           19.4 | 758.7 K |    3.0 K

Total job time: 60.191 sec, wall time: 13.141 sec, speedup: 4.580

Time elapsed:13.343906164169312
```

Figure 4.4 The output of dispy system

4.4.2 Test Bed Configurations

To perform the ECG analytics in normal computing and Fog Computing scenarios, a different set of systems are used. One of the systems is a cluster of four Ubuntu machines called "Ubuntu Cluster" (dual boot) and the other set is a Raspberry Pi cluster, consisting of 4 Raspberry Pis of Model 3 B+. The detailed hardware configurations of these systems are given below in Table 6.

Table 6: Test Bed configuration

Parameters	Ubuntu System Cluster	Raspberry Pi Cluster
Operating Systems	Ubuntu	Raspbian
Processor	I3 – Forth Gene.	Broadcom BCM2837B0, Cortex-A53 (ARMv8) 64-bit SoC
RAM	4 GB	1 GB
Operating Frequency	2.90 GHz	1.4 GHz
Number of Cores	4	4

4.4.3 Results

A set of ECG waves are given to the Head node in the cluster. Where it has divided the ECG waves into individual waves and these waves are given to each and every node to process it. The overall decision time is measured as computation time. In Dispy it is called wall time, which is starting to end processing of each wave of a particular set of 1000, 2000, 3000, 4000, and 5000. The measuring time in ms is noted and presented below in Table 7.

System Performance in terms of Graph is shown below. Here, the X-axis shows the number of ECG waves and the Y-axis shows the time taken in milliseconds.

Table 7: Dispy System Experimental Results

No. of ECG waves	Ubuntu system cluster	Raspberry Pi cluster
1000	448	155
2000	512	391
3000	945	757
4000	1378	1029
5000	1963	1407

To understand the computing nature of both the systems at different data sets, the trend line is drawn. The trend line shows that as the number of waves is increasing the Raspberry Pi performance is increasing with respect to the Ubuntu cluster.

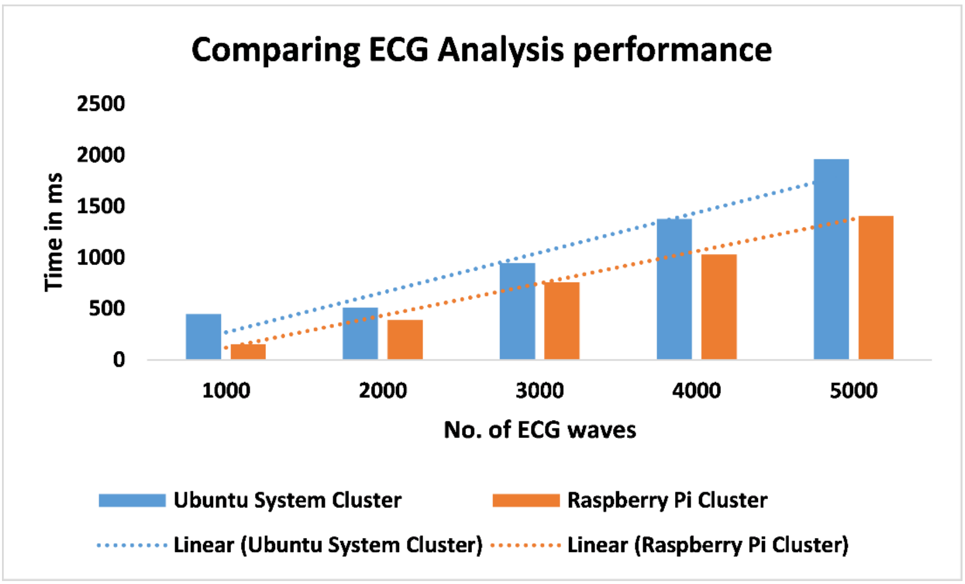


Figure 4.5 Dispy System Analogy

4.5 Improving Pattern matching performance in Genome sequences using Run Length Encoding in Distributed Raspberry Pi Clustering Environment

A genome is the complete set of genetic information in an organism. The genome essentially is an encoded representation of a human body. It is stored in long molecules of DNA called chromosomes [104]. Chromosomes are made up of genome sequences which are nothing but a list of nucleotides - A, C, G, T (T for DNA genomes). In a nutshell, a human can be represented or encoded by its genome sequence as a continuous stream of letters. Genome has found its great applications in the medical areas like Genetic Testing, Tumour Profiling and Pharmacogenomics.

4.5.1 Problems with the traditional Genome Pattern matching

There are numerous researches and experiments on genomes and DNA. Due to which the availability of genomic data of various creatures and living organisms, are naturally very large in sizes and exhaustive pattern matching algorithms are needed to process such large chunks of data. Not to forget the heavy processing and analyses will require good processing power as well. Many algorithms, traditional as well as new ones have been used for pattern matching and several programming languages and their packages exist that can achieve precise string matching in the biological patterns. However, they are bounded to exploring for specific and predefined strings in a series or sequence.

4.5.2 Results and Implementation discussions

To prove the usefulness of Fog Computing and the impact of Distributed Computing in Fog Computing, in the field of medical data processing, here the Genome sequences are analyzed using Raspberry Pi. The implementation has been done in two fashions. One is using a single node and the second one is using the distributed approach of Raspberry Pi clustering.

All results are considered here in terms of time complexity and the percentage of time improvement [114] is shown in table 8 w.r.t the time taken by pattern matching on a single R-Pi node and on the R-Pi cluster.

Table 8: Complete Time Complexity measures and percentage of improvements

Result. No	**No. of Nodes**			**RLE Applied?**	**Time is taken In Seconds**	**Improvement in % w.r.t to Result.No.1**
		Master Node	Slave Node			
1	1	-	-	No	267.209179	-
2	1	-	-	Yes	97.185996	63.62924494
3	3	1	2	No	50.422721	81.12986942
4	3	1	2	Yes	33.618577	87.41862943

The results found out are confirming the good effect of Run Length encoding and Distributed Computing on the Genome pattern matching process. In this, only three Raspberry Pi with Model 3 b+ configurations are used and the improvement is noticeable. This shows that the Fog Node is suitable to process the medical field data and for larger data to handle, one can use cluster of devices with Distributed Computing.

Concluding Remark

Health care data is time receptive in nature, and it should be processed in real-time to obtain the results as early as possible. Delay in such decision making affects the QoS in health care. The real-time ECG analysis system using Cloud Computing suffers from transmission delays. To overcome this, Fog Computing can be used to reduce energy consumption in the total decision-making process and also saves network bandwidth. Raspberry Pi is a good and efficient Fog Computing node. However, a single Raspberry Pi does not have sufficient computational capacity, so a Raspberry Pi cluster is used for distributed and parallel computing purposes. To avoid issues in scalability in the Pi cluster, the Dispy tool is used to provide encryption, remote node connectivity, processing sharing, and data distribution with processing function. Finally, different sets of ECG waves are given to different systems for processing the job. It is found that the Raspberry PI cluster is capable of processing real-time ECG signals faster than the existing Ubuntu-based cluster systems. Hence, the Raspberry Pi cluster can be used as a Fog Computing node to boost the computational power and to provide real-time decision making in the Fog Computing scenarios for varying number of ECG channels, each channel representing a patient. Raspberry Pi cluster provides scalability, and dispy serves secured and faster processing. This cluster can also be in other medical or real-time processing systems like heavy computations of Genome sequencing, big data analytics on Fog, or in real time road traffic monitoring systems.