Journal of Data Acquisition and Processing
ISSN: 1004-9037
https://sjcjycl.cn/
DOI: 10.5281/zenodo.777744

A GENETIC-BASED LOAD BALANCING ALGORITHM FOR ENERGY-EFFICIENT RESOURCE ALLOCATION IN CLOUD ENVIRONMENTS

Reema Singh[1]
Assistant professor, Smt. Z.S. Patel College of Computer Application.

Dr. Narendra Sharma[2]
Assistant professor, Department of CSE, Sri Satya Sai University of Technology & Medical Sciences

1. INTRODUCTION

1.1 Background of Research

The project describes the data about genetic-based load balancing algorithm for energy efficient resources allocation and this balancing process is done in the cloud environment. This approach has been used for the balance of the cloud environment, and this genetic-based algorithm is used for various types of solutions regarding optimization issues (Kaur and Anand, 2022). To maximize resource utilization and prevent cloud computing resources from becoming overloaded with traffic problems, load balancing can be regarded as a fundamental strategy. Fundamentally, the adoption of load-balancing techniques aids in the equitable distribution of "network traffic" among the pool of resources supplied by an application.

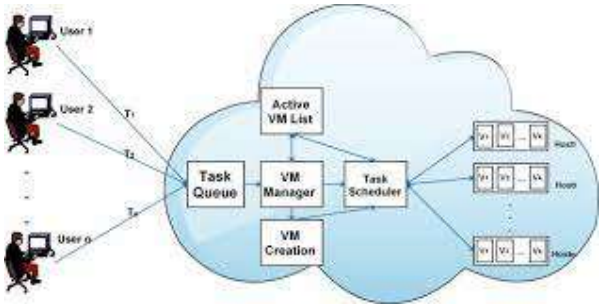

Figure 1.1.1: Load Balancing in a cloud environment
(Source: Ilankumaran and Narayanan, 2023)

The focus of this project is the creation of a genetically based load balancing algorithm in green cloud computing to improve energy efficiency. Here researcher has used the Python language to meet their software requirements, here also provides data about the load balancing algorithm. Using the software researcher identifies the crucial factors of load unbalancing in the cloud environment.

1.2 Aim and Objectives

Aim

The main focus of this project is ***to develop a "genetic-based load balancing algorithm" and execute the energy-efficient load balancing using a cloud environment.***

Objectives

- To develop the load-balancing algorithm and identify the disadvantages of unbalancing in the cloud environment.

- To check the usability of the algorithm that can produce the most efficient outcome.
- To identify the most feasible algorithm for load balancing in cloud computing**.**
- To provide the proper solution regarding load unbalancing problems in a cloud environment.
- To display the benefits of utilizing a cloud environment that delivers energy efficiency along with an eco-friendly environment.

1.3 Project Specification

The project has provided different types of algorithms for load balancing and increased the ability to allocate resources in a cloud environment. Thus, also decreasing job completion times, task costs, and many other factors. Additionally, it will be feasible to reduce the flow of carbon from computing resources by using the "genetic-based algorithm" of "load balancing". Further, the implementation of load balancers and dynamic resource allocation will fundamentally aid in reducing under- and overutilization of computing resources in cloud computing settings (Singh *et al.* 2022). The occurrence of load unbalancing in a cloud environment poses several difficulties, including inadequate performance monitoring, limitations on resource scheduling, "QoS management," failed virtual machines, and many more. The inability to accurately "calculate anticipated loads", "Total costs of ownership", and many other issues can lead to load unbalancing. A cloud balancing method based on genetics that should be energy-efficient has been developed using green cloud computing (Somasundaram *et al.* 2022). Here researcher provides the various data regarding the cloud environment. Importance of the cloud balancing and its various challenges are also provided. Thus, there is also provided which type of issues has been created in the cloud environment during the research and their solution.

2. LITERATURE REVIEW

2.1 Overview

This technical reporter is based on developing ***"a genetic-based load-balancing algorithm for energy-efficient resource allocation in cloud computing"***. The primary purpose of this is to understand the challenges that can be faced due to the unbalanced load and modify an algorithm to mitigate that issue. Several algorithms are available to mitigate the unbalance of load but that has consumed a huge amount of energy. The identification of the algorithm should be based on energy efficiency. The importance of load balancing in a cloud environment should be identified and understood (Sriram, 2022). Several load-balancing techniques are addressed for the cloud, and challenges and constraints have been determined as well. The algorithms that are popular to mitigate load balancing-related issues are,

- ***Round-Robin Algorithm***
- ***Equally Spread Current Execution Algorithm***
- ***Max-Min Algorithm***
- ***Opportunistic Algorithm***
- ***Min-Min Algorithm***
- ***Active Monitoring Algorithm***

"Round robin algorithm" is the most efficient and simple algorithm among the others as well as the energy consumption is minimum. This algorithm has been used for the execution of the load balancing as it can be the most energy-efficient option. It can forward the request of clients

in turns to each server. The load balancer has been looped back and went down in the list again after reaching the end of the list to send the next request. Every processor is involved at the same time and a minimum waiting time can be seen for each request (Pradhan *et al.* 2022). The main advantage of this algorithm is the execution process for this is simple and the outcome that has been produced is more accurate.

2.2 Importance of Load Balancing in Cloud Environment

"Cloud Load Balancing" has the technique of circulating workloads astride computing aids in the ***"Cloud Computing Environment"*** or ***"Carefully Balancing"*** the web traffic entrance to various aids. ***"Load Balancing"*** facilitates associations to assemble report directions by routing approaching gridlock of considerable servers, networks, and different resources while enhancing implementation and defending convulsions in services. ***"Load Balancing"*** is also created it conceivable to disperse workloads astride geographic areas. ***"Cloud Load Balancing"*** has supported establishments to perform elevated implementation classes for potentially lower prices than conventional assumptions ***"Load balancing"*** technology (Dong *et al.* 2021). ***"Cloud Load Balancing"*** has assumed the importance of the cloud's scalability or agility to assemble the needs of dispersed workloads with increased numerals of customer associations. It again enhances overall availability, advances throughout or facilitates latency. The additional to workload or traffic allocation, ***"Cloud Load Balancing"*** benefits generally propose different components, like application health examinations, automated scallions or failover, and combined certification management. ***"Cloud Load Balancing"*** is accepted as a software-based technique for spreading network traffic across aids, as fought to hardware-based shipment balancing, which has more ordinary in company data centres.

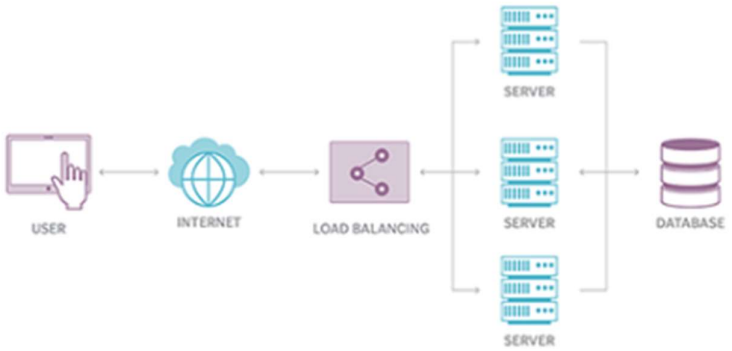

Figure 2.2.1: Impotence of Load Balancing Cloud Work
(Source: Dai *et al.* 2021)

"Cloud Load Balancing" accepts incoming traffic or routes various recommendations to occupied marks established on a configured approach. The ***"Load balancing"*** service again observes the health of the particular marks to confirm that different resources have fully functional. The association operates the increased-traffic website or application, and database that acquires a bunch of queries. ***"Load balancing"*** has provided considerable advantages by optimizing resource usage, data delivery, or response duration. Increased-traffic environments, ***"Load balancing"*** has created user demands to proceed smoothly and accurately (Wang *et al.* 2019). ***"Load balancing"*** again recreates a fundamental function in preventing downtime or

facilitating security, decreasing the possibility of lost productivity or lost profits for the association. The different importance of the ***"Load balancing"*** the observing:

- ***Redundancy*:** The dispersing of traffic astride a team of servers, ***"Load balancing"*** delivers created-in sameness. The learner can automatically deliver the shipment to operating servers to underestimate the effect on users.

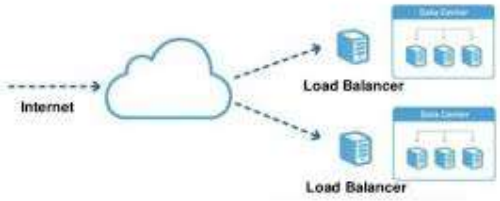

Figure 2.2.2: Importance of Load Balancing Cloud Work

(Source: Han *et al.* 2019)

- ***Scalability*:** The utilization of an application and website growths, the boots in traffic could hinder its implementation if not handled correctly. ***"Load balancing"***, the learner has acquired the capacity to count a physical and virtual server to adjust needs without generating a service disturbance. The unique servers arrive online, and the load balancer remembers or seamlessly possesses the approach. This technique has preferable to driving a website from the congested server to the new one, which frequently needs some quantity of downtime (Khaleghzadeh *et al.* 2022).
- Flexibility: Excluding controlling traffic to maximize efficiency, the ***"Load balancing"*** produces the flexibility to count or subtract servers as need dictates. It again creates it feasible to execute server supervision without driving disturbance for users since gridlock acquires delivered to different servers during supervision.

2.3 Load Balancing Algorithm Techniques in the Cloud Environment

"Load balancing" has a technique of circulating web traffic equally over a collection of aids that help an application. Current applications considerably procedure millions of users simultaneously or replace the accurate text, videos, pictures, and different data for individual users in a quick or dependable technique. To manage such elevated magnitudes of traffic, considerable applications have numerous aid servers with identical data (Khalgui *et al.* 2019). The ***"Load Balancer"*** has an instrument that seats into the user or the server status or functions as an imperceptible facilitator, providing that every resource servers have utilized equally. The ***"Load Balancer"*** technique is of 2 Varieties with Software load balancers and hardware load balancers.

- ***Software Load Balancing*: *"Software-based Load Balancing"*** have applications that function in all "***Load balancing"*** processes. The learner cab established on any server and access an exhaustively controlled third-party service. The different ***"Software-based Load Balancing"*** has considerably more flexible. There can scale up and down smoothly or have more consistent with current ***"Cloud Computing Environments"***. There are again prices less to set up, operate, and utilized time.

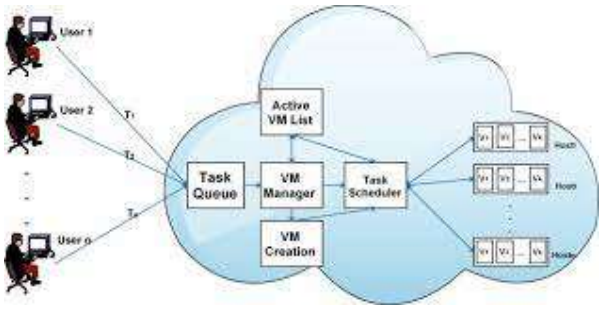

Figure 2.3.1: Technique of Cloud Environment in Load Balancing

(Source: Sleem *et al.* 2020)

- ***Hardware Load Balancing*:** The ***"Hardware-based Load Balancing"*** has a hardware appliance that could securely process or turn gigabytes of gridlock into hundreds of other servers. The learner can accumulate it in the data commands or utilize virtualization to develop considerable digital and virtual "***Load balancing"*** which the learner can centrally control. ***"Hardware-based Load Balancing"*** has needed an initial acquisition, design, or continued supervision. The learner can again not utilize complete capability, specifically if the learner buys one exclusively to manage peak-time traffic points. The traffic significantly grows unexpectedly above its present capability, this can impact users until the learner can buy or set up a different load balancer (Eltresy *et al.* 2019).

There have other varieties of ***"Load balancing Algorithms"*** that ***IT*** groups reach for counting on the allocation of load more its limitation on the web and application layer. The section on choosing backend servers to deliver the traffic has established the ***"Load balancing Algorithm"*** utilized. The algorithm assumes between deliberation of 2 elements of the server with ***"Predefined Condition"*** and ***"Server Health"***. Occasional ordinary ***"Load balancing Algorithm"*** has been utilized continually by ***IT*** group possess.

- ***Weighted Least Connections*:** The ***"Weighted Least Connections"***, load allocation has established on both elements the numerals of present and dynamic association to individual servers and the comparative capability of the server.

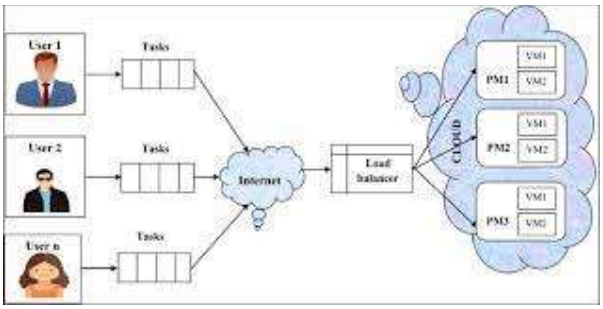

Figure 2.3.2: Technique of Cloud Environment in Load Balancing

(Source: Khaleghzadeh *et al.* 2019)

- ***Round Robin Algorithm*: *"Round Robin Algorithm"*** has the circular allocation of appeals to company servers in an arrangement. There have ***2*** classifications of ***"Round Robin - Weighted Round Robin"*** or ***"Dynamic Round Robin"***. Utilized specifically for a collection of various waitpeople, in ***"Weighted Round Robin"*** a particular server has designated a weight hanging on its design. ***"Dynamic Round Robin"*** have utilized

to dispatch recommendations to associated waitpeople established on actual time estimation of given server weights.

- ***Source IP Hash*: *"Load balancing"*** a server has chosen or established on the distinctive ***"Hash Key"***. The ***"Hash Key"*** is developed by assuming the source or goal of the recommendation.
- ***The Custom Load Method***
- ***The Least Response Time***
- ***Least Connections***
- ***URL Hash***
- ***The Least Bandwidth Method***

2.4 Challenges and Constraints in the Load Balancing Process

"Cloud Computing Environment" has a location in the occupation of ***IT***. ***"Load balancing"*** has one of the primary challenges in ***"Cloud Computing Environment"***. It has a strategy that has needed to spread the dynamic workload to considerable nodes to assure that no single node has overfilled. Usually web traffic, database, application key, and different items which have serious loads could utilize the ***"Load Balancer"*** software to help the user with any interruption. ***"Load Balancing Technique"*** support in the optimal utilization of aid or thus in the importance of the implementation of the technique (Dai *et al.* 2021). The purpose of "***Load balancing"*** has minimized aid consumption which can additionally decrease power consumption or carbon emission speed which has the dire requirement of a ***"Cloud Computing Environment"***. Describing the ***"Load Balancing Algorithm"*** of the cloud environment it has needed to recognize some fundamental challenges or problems that impact the implementation of the ***"Load Balancing Algorithm"***. Observing have some significant challenges that can be enhanced for a more acceptable implementation of the ***"Load Balancer"***.

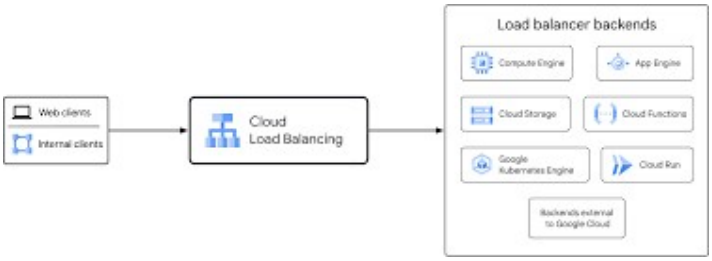

Figure 2.3.1: Challenges of Cloud Load Balancing

(Source: Khalgui *et al.* 2019)

- ***Response Time*:** A circulated technique, it has the duration accepted by a separate load balancing technique to reply.
- Fault Tolerant: the learner can describe it as the capability to achieve ***"Load balancing"*** by the applicable algorithm without an incidental connection and node defeat. All ***"Load Balancing Algorithms"*** would have a suitable defect tolerance technique.
- ***Associated Overhead*:** It defines the quantity of overhead during the performance of the ***"Load Balancing Algorithm"***. It has a document of indication of the report, inter-procedure transmission, or inter-processor (Han *et al.* 2019).
- ***Migration Time*:** It has the quantity of the duration for a procedure to be transmitted from one technique node to a different node for performance

- ***Scalability*:** It has the capability for a ***"Load Balancing Algorithm"*** of a procedure with some limited numeral of processors or devices.

Figure 2.3.2: Challenges of Cloud Load Balancing

(Source: Wang *et al.* 2019)

- ***Performance*:** It has the prevailing efficiency of the technique that all the parameters have enhanced then the prevailing technique implementation can be enhanced. Analysis in a ***"Cloud Computing Environment"*** has always been in its premature steps, or any scientific challenges stay unsolved in a scientific society, especially ***"Load balancing"*** challenges.
- ***Throughput*:** It has a full numeral of reports that have terminated performance for a provided scale period. It needed to have increased via put for a more acceptable implementation of the technique (Khaleghzadeh *et al.* 2019).
- ***Resource Utilization*:** It has the parameter that provides the data in that extant the aid has operated.

2.5 Literature Gap

A gap has been found due to the study of literature for the technical report about developing ***"a genetic-based load-balancing algorithm for energy-efficient resource allocation in cloud computing".*** This gap can be addressed as the literature gap for the project that can occur due to several issues. Some data can not be accessed due to maintaining confidentiality and some of them can not be found in libraries or the internet. Some research papers are secured with passwords that can not be accessed. This gap can be fulfilled to some extent using the practical evidence for the project about cloud computing.

2.6 Summary

The overall concept of cloud computing has been addressed in this section and the load balancing of the system is also understood. Several algorithms can be applied for the load balancing and the importance of that has been determined as well. Several load-balancing techniques are illustrated and the challenges are identified too. The development of the genetic-based load balancing algorithm is used and the most efficient has been analyzed using software. The energy consumption for this algorithm can vary so the best option that is used should be most simple and energy efficient. The round-robin algorithm can be used for load balancing as this is the simplest option and the outcome that produced by this algorithm has been the most accurate. The energy consumption has been minimum in it so it has been used as the most energy-efficient algorithm for load balancing in cloud computing. The literature gap is also addressed and an implementation of practical evidence should be done to mitigate that gap.

3. METHODOLOGY

3.1 Choice of Methods

Energy-Efficient Resource Allocation in Cloud Environments project preparation, research, data collection, and implementation process are performed under the agile methodology. It allows for research of proper subjects through the proper sources, optimizing adequate objectives to execute the entire process, and providing accurate outcomes. The Project is based on the secondary-mixed research process and secondary data collection process. Different research papers and literature has been observed to evaluate tangible insights into the subject. Secondary research allows to obtain adequate data with relevant information and understanding suggested by the researchers.

Figure 3.1.1: Mixed research process

(Source: Qureshi *et al. 2020*)

The mixed research process allows to obtain accurate and optimized data of relevant resources. It helps to analyze data based on the required research objectives to implement the project effectively. As the project involved implementing the cloud algorithm to balance the load on a server, it requires an extended understanding of the fats of the phenomenon. Python programming language has been utilized to obtain significant outcomes. It can be optimized that the chosen methods for this project operations are significant to their operations.

3.2 Justification of choice and support

Each of the methods has been considered as per their functionalities and purposes significant to the project. Optimizing an accurate cloud algorithm is necessary for the effective implementation of load balancing. Python language provides a wide range of libraries that allows the operation of different analysis and processes. The "Round-Robin" algorithm of load balancing is utilized to acquire the facility to balance the client request in distributed servers. As per the view of Kaur ***et al. (***2019), the client request is actively balanced by the algorithm in each turn. Prevention of extra load on the network can be achieved by this algorithm. Also, the requirement for resources is fewer with respect to other load-balancing algorithms. Effective operation over the traffic control of a network can be achieved by this particular algorithm. As per the view of Qureshi ***et al. 2020***), these mechanisms justify the choice of considering this algorithm for this purpose. Additionally, the mixed and secondary research

approach can be justified with accurate insight extraction on the entire process. Significant data and adequate information extraction from the research papers allow for optimizing different factors relevant to the implementation process of the project work.

3.3 Data collection

A secondary data collection process has been deployed in this project which helps in analyzing potential gaps and requirements of further investigation and implementation on the project work. This particular process allows for collecting data from legitimate resources such as government and social organizational data and researched optimized data.

Figure 3.3.2: Secondary data collection process

(Source: Roh *et al.* 2019)

There are numerous benefits to utilizing this process of data collection including, time-saving, cost-efficiency, relevance, and optimized factors. As per the view of Roh *et al.* (2019), previously done research papers allow to understand the gaps and deficiencies in the research and optimization of the process adopted which allows innovation. Thus, it can be referred to as the most efficient and relevant data collection process referring to this project implementation. The entire work is based on the acquired knowledge from previously done research papers and results.

3.4 Validation

All the requirements of this project have been achieved with the integration of a significant choice of methods from different aspects and processes. As per the view of Jamil *et al.* (2022), Proper validation allows overviewing of all the factors necessary in the implementation process. Hence, all the choice of language and algorithms has been satisfied with the requirements. Determination of the facts helps in validating the entire project objectives.

3.5 Commercial Risk and risk management

Ethical and commercial risks associated with the project include adequate optimization of tools and techniques and satisfaction of the objectives with relevant data and methods. The most significant commercial risk associated with this project is the security risk as it involves the web network and server operations (Suleiman and Hamdan, 2021). Also, the ethical risk associated with this project can be referred to as legitimate and unaltered data from the resources.

3.6 Tools and Techniques

Several tools and techniques are involved in the analysis and implementation of the result practically. A better understanding can be done using a practical approach and for that suitable software should be used to produce accurate outcomes (Rahmani ***et al. 2020***). The software that has been used for the design and implementation of the load-balancing algorithm for cloud computing is ***"Python"***. Python is a well-known programming language that can be used for data analysis and visualization. The algorithm which has been considered as the most suitable one for the project is the ***"Round robin algorithm"*** as it is easy to use and energy efficient as well.

3.7 Key Consideration

Several basic requirements are considered critical for the project in the research methodology. The strategies of the research should be prominent to achieve an accurate outcome and the philosophy of the study should be positive to have a good impact on society. The data collection should be done very carefully by that the practical approach of this project and analysis process has an effective outcome. The tools and techniques should be used on point to generate a precise outcome and the conclusion for the project has been done prominently. The literature study has been done for a better understanding and to gather information about the research.

4. RESULT AND DISCUSSION

4.1 Design and Implementation

The development of the algorithm for the load balance for cloud computing which should be energy efficient. The overall process has been done using Python and the calculation should be executed in two different steps. The checking of load status in the processors should have been done in the first step and in the second step, the server availability has been checked as well.

```
import threading
import time

r1 = 0
```

Figure 4.1.1: Import libraries to check the processors

The libraries that should be required for the checking of the processors' availability have been imported. The value of ***r1*** has been assumed as ***zero*** in the initial state of the 1st step of the program.

```
print("Assume two processers")

l1 = int(input("limit of processor 1: "))
l2 = int(input("limit of processor 2: "))
n = int(input("Number of processes: "))

n = n + r1
```

Figure 4.1.2: Print to enter for load checking

The limit of these two processors is set as well as the number of processes is also defined. This is the code that is used to print the values as the values are user-defined (Javadpour *et al.* 2022). The input of the processors is represented as ***l1*** and ***l2***, on the other hand, the number of processes has been denoted as ***n*** in the program. The change of the value of n concerning r1 has been done using the code given above.

```
def run1():

    print()
    print(f"Processor 1 (limit = {l1}):")

    if(n == 0):
        print("No processes are remaining")

    else:

        if(l1 > n):
            print("Underloaded")
            print(f"{n} processes are executed")

        elif (l1 == n):
            print("Normal")
            print(f"{n} processes are executed")
            print("No need to forward any process to next processor.")

        elif(l1 < n):
            print("Overloaded")
            print(f"{n} processes are executed")
            rem = n - l1
            print(f"{rem} will be forwarded to next processor")

    global r1
    r1 = rem
```

Figure 4.1.3: Code to check load in processor 1

The conditions for overload, normal, and underload have been defined for ***processor one***, using the codes given in the above figure. The number of the processes has been defined by the user and if that is not equal to zero then the conditions are executed. The processor is said to be

underloaded if the limit is greater than the processes, on the other hand, it is said as ***overloaded*** if the limit is less than the number of processes. The limit is equal to the process in the ***normal*** load and all the results should be printed according to the code.

```
def run2():

    time.sleep(5)

    print()
    print(f"Processor 2 (limit = {l2}):")

    if(r1 == 0):
        print("No processes are remaining")

    else:

        if(l2 > r1):
            print("Underloaded")
            print(f"{r1} processes are executed")

        elif (l2 == r1):
            print("Normal")
            print(f"{r1} processes are executed")
            print("No need to forward any process to next processor.")

        elif(l1 < r1):
            print("Overloaded")
            print(f"{n} processes are executed")
            rem = n - (l1 -l2)
            print(f"{rem} will be forwarded to next processor")
```

Figure 4.1.4: Code to check load in processor 2

A similar process can be seen for processor 2 and for that, the code has been developed, which can be seen in the figure above. The conditions are executed if ***r1*** is not equal to zero and the result has been printed according to the code. The processor has been considered ***underloaded*** if the limit of the second processor is greater than ***r1***. Similarly, when the limit is less than ***r1*** then that is called ***overloaded***. The ***normal load*** has been said when the limit is equal to ***r1*** and all the conditions are defined using the codes above.

```
def run():

    t1 = threading.Thread(run1())
    t2 = threading.Thread(run2())

    t1.start()
    t2.start()


run()
```

Figure 4.1.5: Code to check the threads in both processors

Two threading can be seen for two processors that are denoted as ***t1*** and ***t2*** and after the value determination the program should be ready to run.

```
Assume two processers
limit of processor 1: 10
limit of processor 2: 15
Number of processes: 20

Processor 1 (limit = 10):
Overloaded
20 processes are executed
10 will be forwarded to next processor

Processor 2 (limit = 15):
Underloaded
10 processes are executed
```

Figure 4.1.6: Outcome of checking load in both processors

The Outcome is to check the load status of each processor and as the program is set to be user-defined the values can be defined in the result. The limit for processor one is assumed as ***10*** and for processor two it will be ***15***. The number of processes that have been defined is ***20*** so according to the program the process number is greater than the limit so it is ***overloaded***. The excess ***10*** processes have been done through processor two whose limit is ***15***. The processor two is ***underloaded*** as can be seen in the figure above.

```
import heapq
from matplotlib import pyplot as plt
```

Figure 4.1.7: Import libraries for load balancing

The availability check of the servers has been done that is used for the load balancing. The algorithm that has been used for the load balancing is the ***Round Robin algorithm***, which can produce the most accurate outcome. The libraries have been imported for the program which can be seen in the figure above.

```
def print_load_on_each_server(m, load_on_server):
  for i in range(m):
    print(f"{i+1}st Server -> {load_on_server[i]}.")
```

Figure 4.1.8: Code for the load on each server

The code to print the load for each server has been done and can be seen in the above figure as well as the number of servers is defined later in this programming.

```
def load_balancing(n, m, arrival_time, process_time):
  load_on_server = [0] * m
  busy_servers = []
  available_servers = set(range(m))
  for i in range(n):
    end_time = arrival_time[i] + process_time[i]
    while busy_servers and busy_servers[0][0] <= arrival_time[i]:
      released_server = heapq.heappop(busy_servers)
      available_servers.add(released_server[1])
    if not available_servers:
      continue
    demanded_server = i % m
    assigned_server = min(available_servers, key=lambda x: (x - demanded_server) % m)
    load_on_server[assigned_server] += 1
    available_servers.remove(assigned_server)
    heapq.heappush(busy_servers, (end_time, assigned_server))
  print_load_on_each_server(m, load_on_server)
```

Figure 4.1.9: Code for load balancing and check the availability

The above figure shows the coding that has been created to check the availability of each server. The identification of the busy server and available servers has been done using the conditions (Simon *et al.* 2022). The arrival time and end time should be required for the calculation for each server.

```
if __name__ == "__main__":
  arrival_time = [1, 2, 6, 5]
  process_time = [9, 5, 2, 1]

  n = len(arrival_time)
  m = 6
  load_balancing(n, m, arrival_time, process_time)
```

Figure 4.1.10: Code to calculate load balancing

The number of servers that are denoted by ***m*** has been defined as ***6*** and the arrival as well as process time has been determined. The load balancing can be done and the outcome has been produced accordingly.

```
plt.plot(arrival_time, process_time)
plt.title("load_balancing")
plt.ylabel('Y axis')
plt.xlabel('X axis')
plt.show()
```

Figure 4.1.11: Code to visualize the outcome

The above figure shows the code to develop a graphical representation of the arrival time and process times.

```
1st Server -> 1.
2st Server -> 1.
3st Server -> 1.
4st Server -> 1.
5st Server -> 0.
6st Server -> 0.
```

Figure 4.1.12: Outcome to show server status

The outcome of the server availability and load balancing can be seen in the figure above. It is the value of ***n*** that can be calculated by ***arrival time*** and the compiled format of ***m*** and ***n*** can be seen as a result.

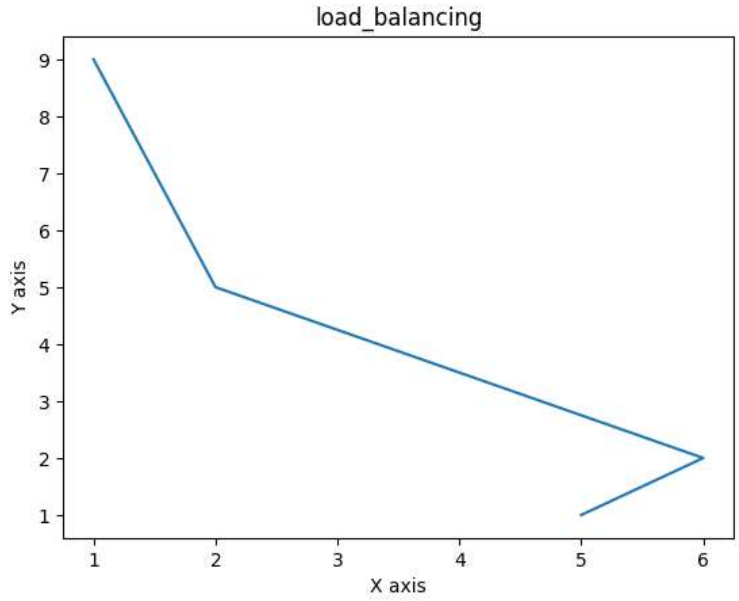

Figure 4.1.13: Graphical representation of the outcome

The graphical representation of the outcome of the load balancing using the ***"Round Robin algorithm"*** has been done and is seen in the figure above.

4.2 Discussion

The algorithm that has been used to implement the overall project is the Round Robin algorithm which can be considered the most efficient among all other algorithms. The status of loads has been checked for the processors first and then the load balancing has been done by checking the availability of the servers. Initially, an auxiliary array has been created to store the loads to the servers. The arrival time and process time for each request have been determined which is used to calculate the end time. The busiest server has been popped out from the queue of priority that has the end time which passed the current end time. The current request has been dropped if the server availability is zero (Biswas *et al.* 2023). The suitable servers have been searched by the lower bound function and if it is pointing to the end then the first server has been chosen. The load counter has been increased on the chosen server and after all the steps the loads have been printed as the result according to the algorithm of ***Round Robin.***

5. CONCLUSION

5.1 Critical Evaluation

The project provides the data about genetic method for load balancing in a cloud environment. Here user highlights the crucial data regarding the cloud environment load balancing, hence it also carried the software work. Here researcher has used the Python coding language, using python researcher creating two processes and there are mentioned the limit. Thus, the researcher provides the proper report after that analysis and this report also contains the various load-balancing techniques. The method of load balancing is that distribute the network traffic and use equal resources which has help to support the application.

There are various types of load balancing methods available ***"Round Robin", "Weighted Round Robin", "Least Connection", "Resource-based"*** and many more methods. The load balancing of the cloud computing method is also beneficial for the system's workability and

performance. Proper load balancing provides ***"better workload scalability and performance", "better work reliability", "better business continuity (BC) and governance"*** and many more. Hence researcher also provides the challenges regarding the cloud load balancing method. Load unbalancing has created various problems which provide an effect on the workability of the system. Here researcher provides the proper method of the work which has containing the approach for the particular study, their using strategy, and many more.

5.2 Recommendation

Some of the suggestions for enhancing load-balancing methods and algorithms are highlighted in this report. Lack of informality causes research to be contradictory and it can prevent new ideas from being incorporated into projects, which lowers efficiency. Therefore, standardization development is crucial since it can contribute to the creation of cutting-edge algorithms and approaches that could lessen complexity and enhance the operability of the cloud environment. Load balancing suffers greatly from differences between simulators and real-wors situations, a hybrid simulation environment should be established for testing procedures (Rawas and Zekri, 2021). The load balancers' full potential, which can be used to enhance and upgrade the balancers regularly, is realized in hybrid cloud setups. This facilitates the use of load balancing across many platforms, improving performance and efficiently lowering energy use. It is crucial to use AI in the sectors of developing load balancers since, as everyone is aware, it functions effectively when dealing with techniques and algorithms. AI and machine learning algorithms can be trained under multiple scenarios to improve decision-making effectiveness and efficiency while maintaining an even workload across all servers. Forecasting resource demands and their allocation, machine learning techniques, and algorithms can be helpful (Hiremath and Rekha, 2023).

5.3 Future Scope

Load balancing in cloud environments appears to have a promising future which can give ideas of different approaches related to the development and improvement of load-balancing g algorithms. The results have been efficient and effective, making work easier while maintaining balance. More advanced and improved algorithms, edge computing, containerization, and hybrid and secure load balancing are some technologies that may make load balancing the optimum method for managing workloads. The storage has been preserved for further information on recycling cycle processes, making the green cloud the sustainable alternative for the future. With these advancements, the load-balancing system will be able to produce more intelligent, effective, and efficient results.

REFERENCE

Biswas, D., Samsuddoha, M., Asif, M.R.A. and Ahmed, M.M., 2023. Optimized Round Robin Scheduling Algorithm Using Dynamic Time Quantum Approach in Cloud Computing Environment. *Int. J. Intell. Syst. Appl*, *15*, pp.22-34.

Dai, S., Meng, F., Dai, H., Wang, Q. and Chen, X., 2021. Electrical peak demand forecasting-A review. *arXiv preprint arXiv:2108.01393*.

Dong, X., Kedziora, D.J., Musial, K. and Gabrys, B., 2021. Automated deep learning: Neural architecture search is not the end. *arXiv preprint arXiv:2112.09245*.

Eltresy, N.A., Dardeer, O.M., Al-Habal, A., Elhariri, E., Hassan, A.H., Khattab, A., Elsheakh, D.N., Taie, S.A., Mostafa, H., Elsadek, H.A. and Abdallah, E.A., 2019. RF energy harvesting IoT system for museum ambience control with deep learning. *Sensors*, *19*(20), p.4465.

Han, S., Gu, M., Yang, B., Lin, J., Hong, H. and Kong, M., 2019. A secure trust-based key distribution with self-healing for internet of things. *IEEE Access*, *7*, pp.114060-114076.

Hiremath, T.C. and Rekha, K.S., 2023. Energy Efficient Data Migration Concerning Interoperability Using Optimized Deep Learning in Container-Based Heterogeneous Cloud Computing. Advances in Engineering Software, 183, p.103496.

Ilankumaran, A. and Narayanan, S.J., 2023. An Energy-Aware QoS Load Balance Scheduling Using Hybrid GAACO Algorithm for Cloud. Cybernetics and Information Technologies, 23(1), pp.161-177.

Jamil, B., Ijaz, H., Shojafar, M., Munir, K. and Buyya, R., 2022. Resource allocation and task scheduling in fog computing and internet of everything environments: A taxonomy, review, and future directions. ACM Computing Surveys (CSUR), 54(11s), pp.1-38.

Javadpour, A., Abadi, A.M.H., Rezaei, S., Zomorodian, M. and Rostami, A.S., 2022. Improving load balancing for data-duplication in big data cloud computing networks. *Cluster Computing*, *25*(4), pp.2613-2631.

Karaboga, D., Akay, B. and Karaboga, N., 2020. A survey on the studies employing machine learning (ML) for enhancing artificial bee colony (ABC) optimization algorithm. *Cogent Engineering*, *7*(1), p.1855741.

Kaur, A., Gupta, P., Singh, M. and Nayyar, A., 2019. Data placement in era of cloud computing: a survey, taxonomy and open research issues. Scalable Computing: Practice and Experience, 20(2), pp.377-398.

Kaur, H. and Anand, A., 2022. Review and analysis of secure energy efficient resource optimization approaches for virtual machine migration in cloud computing. Measurement: Sensors, p.100504.

Khaleghzadeh, H., Fahad, M., Shahid, A., Manumachu, R.R. and Lastovetsky, A., 2019. Bi-objective optimisation of data-parallel applications on heterogeneous platforms for performance and energy via workload distribution. *arXiv preprint arXiv:1907.04080*.

Khaleghzadeh, H., Manumachu, R.R. and Lastovetsky, A., 2022. Novel bi-objective optimization algorithms minimizing the max and sum of vectors of functions. *arXiv preprint arXiv:2209.02475*.

Khalgui, M., Mosbahi, O. and Li, Z., 2019. On reconfiguration theory of discrete-event systems: From initial specification until final deployment. *IEEE Access*, *7*, pp.18219-18233.

Pradhan, A., Bisoy, S.K., Kautish, S., Jasser, M.B. and Mohamed, A.W., 2022. Intelligent decision-making of load balancing using deep reinforcement learning and parallel PSO in cloud environment. *IEEE Access*, *10*, pp.76939-76952.

Qureshi, M.S., Qureshi, M.B., Fayaz, M., Mashwani, W.K., Belhaouari, S.B., Hassan, S. and Shah, A., 2020. A comparative analysis of resource allocation schemes for real-time services in high-performance computing systems. International Journal of Distributed Sensor Networks, 16(8), p.1550147720932750.

Rahmani, F., Joloudari, J.H., Shamshirband, S. and Mostafavi, S., 2020. Game theory and evolutionary-optimization methods applied to resource allocation problems in emerging computing environments: A survey. arXiv preprint arXiv:2012.11355.

Rawas, S. and Zekri, A., 2021. EEBA: Energy-Efficient and Bandwidth-Aware Workload Allocation Method for Data-intensive Applications in Cloud Data Centers. IAENG International Journal of Computer Science, 48(3).

Roh, Y., Heo, G. and Whang, S.E., 2019. A survey on data collection for machine learning: a big data-ai integration perspective. IEEE Transactions on Knowledge and Data Engineering, 33(4), pp.1328-1347.

Simon, A., Dams, G.L. and Danjuma, S., 2022. An improved half life variable quantum time with mean time slice round robin CPU scheduling (IMHLVQTRR). *Science World Journal*, *17*(2), pp.248-253.

Singh, S.P., Kumar, R., Sharma, A. and Nayyar, A., 2022. Leveraging energy-efficient load balancing algorithms in fog computing. Concurrency and Computation: Practice and Experience, 34(13), p.e5913.

Sleem, L., Noura, H.N. and Couturier, R., 2020. Towards a secure ITS: Overview, challenges and solutions. *Journal of Information Security and Applications*, *55*, p.102637.

Somasundaram, K.S.G.A., Arun, M., Saranya, N.N., Prabha, R. and Babu, D.V., 2022, January. A novel hybrid GAACO algorithm for cloud computing using energy aware load balance

scheduling. In 2022 International Conference on Computer Communication and Informatics (ICCCI) (pp. 1-5). IEEE.

Sriram, G.S., 2022. Edge computing vs. Cloud computing: an overview of big data challenges and opportunities for large enterprises. *International Research Journal of Modernization in Engineering Technology and Science*, *4*(1), pp.1331-1337.

Suleiman, H. and Hamdan, M., 2021. Adaptive probabilistic model for energy-efficient distance-based clustering in WSNs (Adapt-P): a LEACH-based analytical study. arXiv preprint arXiv:2110.13300.

Wang, Y., Zhang, Y., Tao, F., Chen, T., Cheng, Y. and Yang, S., 2019. Logistics-aware manufacturing service collaboration optimisation towards industrial internet platform. *International Journal of Production Research*, *57*(12), pp.4007-4026.