Comparative analysis of different load balancing algorithms in cloud computing

Prof.Kapil Tajane Department of Computer Engineering,

Pimpri Chinchwad College of Engineering, Pune, Maharashtra [Kapil.tanaje@pccoepune.org](mailto:Kapil.tanaje@pccoepune.org)

Utkarsh Gardi Department of Computer Engineering,

Pimpri Chinchwad College of Engineering, Pune, Maharashtra [Utkarsh.gardi21@pccoepune.org](mailto:Utkarsh.gardi21@pccoepune.org)

Sanskar Ghule Department of Computer Engineering,

Pimpri Chinchwad College of Engineering, Pune, Maharashtra [Sanskar.ghule21@pccoepune.org](mailto:Sanskar.ghule21@pccoepune.org)

Utkarsha Kamble Department of Computer Engineering,

Pimpri Chinchwad College of Engineering, Pune, Maharashtra [Utkarsha.kamble21@pccoepune.org](mailto:Utkarsha.kamble21@pccoepune.org)

Aneesh Kanhere Department of Computer Engineering,

Pimpri Chinchwad College of Engineering, Pune, Maharashtra [Aneesh.kanhere21@pccoepune.org](mailto:Aneesh.kanhere21@pccoepune.org)

***Abstract*—** Cloud computing has emerged as a transformative technology, revolutionizing the way businesses manage and process data. In this dynamic landscape, efficient resource utilization is paramount. Load balancing, a critical component of cloud computing, ensures optimal distribution of tasks across servers, thereby enhancing performance and resource utilization. This research paper delves into a comprehensive exploration of both static and dynamic load balancing algorithms in cloud computing environments.

The study begins by elucidating the fundamental concepts of cloud computing and the pivotal role of load balancing. A meticulous review of the existing literature highlights various static algorithms emphasizing their advantages and limitations. Similarly, dynamic algorithms are examined in depth, offering insights into their real-time adaptability and efficiency.

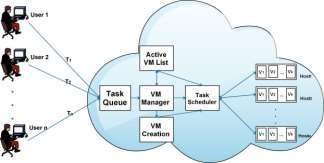
A systematic comparative analysis is conducted, leveraging an extensive array of metrics including response time, throughput, and resource utilization. Through rigorous experimentation and evaluation, the performance, scalability, and applicability of these algorithms in diverse cloud scenarios are rigorously assessed. The findings provide a nuanced understanding of the strengths and weaknesses of each algorithm, guiding practitioners, and researchers toward informed decision-making in their cloud deployments.

***Keywords— Cloud computing; Honey bee algorithm; Load balancing; Opportunistic; Active clustering; Min-min; Round Robin; Max-min; Random biased sampling; Ant colony***

# INTRODUCTION

Cloud computing is one of the most popular technologies adopted by both academia and industry, which provides an efficient and flexible method for storing and accessing files. Cloud Computing is Internet-based computing, in which resources, software and information are shared to computers and other devices on-demand. Cloud provides a facility for users to access information at any time and from anywhere. It is not required for the user to be in the same location as the hardware that stores data. When the user has the internet connection, they can access the services of the cloud. It delivers all the services dynamically through the internet according to the user requirements.

Use the enter key to start a new paragraph. The appropriate spacing and indent are automatically applied.



*Figure.1. load balancing in cloud computing*

* 1. *Importance Of Load Balancing In Cloud Computing*

A website or web application can attract numerous users simultaneously, creating a challenge in managing these requests and preventing system breakdowns. This is where the significance of load balancers becomes apparent. Load balancing is the process of efficiently distributing the workload across multiple computers or servers, ensuring faster task completion and optimal resource utilization. By dividing the workload effectively, no single computer becomes overloaded while others remain underutilized or idle.

Load balancing plays a crucial role in ensuring that every computer or server in the network shares a relatively equal workload. This approach guarantees that no specific node is burdened with excessive tasks, while others handle minimal work, thereby maintaining a balanced distribution of computing resources. In the context of cloud computing, where the primary goals include improving response time, reducing costs, and enhancing overall performance, load balancing becomes indispensable. Hence, cloud computing is often referred to as a pool of services, aligning resources effectively to meet user demands efficiently.

* 1. *Algorithms That Support Load Balancing*

Load balancing algorithms are methods designed to distribute network or application traffic across multiple servers or resources. Their primary purpose is to prevent any single server from becoming overwhelmed with too much traffic, ensuring even utilization of resources. There are mainly two types of load balancing algorithm, those are as follows:

2 Static Algorithms: The static algorithm requires prior knowledge about the system like memory, processing power and performance. These algorithms do not need the information regarding the current state of the node.

While distributing the load a static algorithm does not use the system information and is less complex. These algorithms work properly in a system with low variation of load.

2.1.1 Dynamic algorithms: In dynamic algorithms decisions are made based on the current state of system i.e., it will not consider the prior knowledge about the system. Dynamic algorithms consider different policies like transfer policy, selection policy, location policy and information policy to balance the load. It also considers the changes in the state of the nodes dynamically. At any time if a node is having heavy load it is transferred to a node with light load.

* 1. *Load Balancing Metrics*

Various parameters are considered in the existing load balancing algorithms:

* + - *Response Time:* The time taken to respond to a request. Lower response time indicates better performance and user experience.
    - *Throughput:* The number of requests processed by the system in each timeframe. Higher throughput signifies the system's ability to handle a large number of requests.
    - *Server Utilization:* Measures the extent to which a server's resources (CPU, memory, etc.) are being used. Balanced utilization across servers ensures efficient resource utilization.
    - *Latency:* The delay between initiating a request and receiving a response. Lower latency means faster data processing and better user experience.
    - *Error Rate:* The frequency of errors occurring during request processing. Lower error rates indicate system stability and reliability.
    - *Scalability:* The system's ability to handle increasing loads by adding more resources. A scalable system can accommodate growing demands without compromising performance.
    - *Adaptability:* The ability of the load balancing algorithm to adapt to changing traffic patterns and adjust the workload distribution accordingly.

# REVIEW

This research delves into the heart of cloud computing infrastructure, presenting a systematic and comprehensive analysis of various load balancing algorithms, both static and dynamic. Our aim is to scrutinize the strengths and weaknesses of these algorithms, shedding light on their suitability in different scenarios and their potential impact on cloud service performance.

## Honey Bee algorithm:

The Honey Bee Algorithm [1] is a population-based optimization technique that can be adapted for load balancing in cloud computing. In this context, the algorithm models the

cloud servers as food sources, and the tasks to be allocated to these servers as the nectar. The goal is to efficiently distribute tasks among servers to minimize the overall system load and improve resource utilization.

* + 1. *Basic Steps of the Honey Bee Algorithm:*
       - *Initialization:* The algorithm starts with an initial population of artificial bees representing potential solutions.
       - *Employed Bees Phase:* In this phase, employed bees search for new solutions. They evaluate the quality of these solutions based on a fitness function (usually related to load balancing objectives) and share information with other bees about the quality of the solutions.
       - *Onlooker Bees Phase:* Onlooker bees select solutions based on the information shared by employed bees. Solutions with higher quality (lower load or better resource utilization) are more likely to be chosen.
       - *Scout Bees Phase:* Scout bees identify solutions that have not improved for a certain number of iterations and abandon them, exploring new solutions to replace them.
       - *Update Solution:* After all phases, the algorithm updates the solutions to improve load balancing. This may involve redistributing tasks among servers.
    2. *Mathematical Expressions:*

The mathematical expressions used in the Honey Bee Algorithm for load balancing can vary based on the specific load balancing objectives and fitness functions chosen for your research. A common objective is to minimize the variance of the server loads while ensuring that no server is overloaded. The fitness function, often represented as F(x), can be customized to reflect these objectives.

Here's a simplified example:

F(x) = ∑(w\_i \* L\_i) / n

Where:

F(x) is the fitness function to be minimized. w\_i represents the weight of task i.

L\_i is the load on server i. n is the number of servers.

This fitness function can be used to guide the Honey Bee Algorithm in redistributing tasks to minimize the load variance among servers.

## Active clustering:

Active clustering is a dynamic load balancing algorithm [9] that adapts to changing workloads and resource availability in a cloud computing environment. The core idea behind active clustering is to organize resources into clusters and assign tasks to these clusters based on certain criteria. These criteria may include factors such as CPU utilization, memory usage, network latency, and task priority.

* + 1. *Key Components of Active Clustering:*
       - *Resource Monitoring:* Active clustering continuously monitors the system's resources to gather real-time data on CPU, memory, and network usage. This information is crucial for making informed decisions regarding task assignment.
       - *Cluster Formation:* The algorithm groups resources into clusters based on their proximity in terms of resource usage. Clusters can be dynamically formed or adjusted as the system state changes.
       - *Task Assignment:* Tasks or workloads are assigned to clusters based on specific criteria. For instance, tasks with high computational requirements may be assigned to clusters with idle CPUs.
       - *Load Balancing Metric:* Active clustering employs load balancing metrics, often denoted as "LB," to determine which resources or clusters are suitable for task assignment. The choice of this metric depends on the specific requirements of the system and the algorithm's design.
       - *Dynamic Resource Allocation:* Resources within a cluster may be dynamically allocated to tasks based on their requirements. This ensures that resources are optimally utilized, and tasks are executed efficiently.
    2. *Mathematical Expressions:*

Active clustering algorithms can incorporate mathematical expressions to make load balancing decisions. For example, the load balancing metric (LB) can be expressed mathematically as a function of resource utilization metrics:

LB = f(CPU utilization, Memory utilization, Network latency, Task priority)

Here, "f" represents a function that combines these metrics to calculate a load balancing score. The task or workload is then assigned to the cluster with the highest load balancing score.

## Ant colony:

The Ant Colony Load Balancing Algorithm [12] works by modelling the problem as an optimization task, where ants represent computational tasks and servers represent the locations where ants can deposit tasks. Each ant (task) searches for the best server (path) to deposit its task based on a combination of pheromone levels and heuristics.

* + 1. *Key Components of ACO Load Balancing:*
       - *Ants:* In the context of load balancing, ants represent computational tasks or workloads that need to be allocated to servers.
       - *Servers:* These are the available resources that can process the tasks. Servers can be characterized by their current workload, processing capacity, and other relevant factors.
       - *Pheromone Trails:* Pheromone levels are associated with the paths or connections between ants (tasks)

and servers. Ants deposit pheromones on their chosen paths, and these pheromone levels influence the choices of subsequent ants.

* + - * *Heuristics:* Heuristics are used by ants to estimate the desirability of a server based on factors such as server load, processing capacity, and task characteristics.
      * *Probabilistic Decision-Making:* Ants make probabilistic decisions when choosing a server for task allocation. The probability of selecting a particular server is influenced by both pheromone levels and heuristics.
    1. *Mathematical Expressions:*

ACO-based load balancing algorithms use mathematical expressions to guide ants' decision-making. These expressions combine pheromone levels and heuristics to calculate the desirability of selecting a particular server. Here's a simplified example of a mathematical expression for the desirability (D) of choosing server i by an ant:

D(i) = (T(i)^α) \* (H(i)^β) Where,

D(i) represents the desirability of choosing server i.

T(i) is the pheromone level on the path leading to server i. H(i) is a heuristic measure representing the attractiveness of server i.

α and β are parameters that control the relative influence of pheromones and heuristics. These parameters can be adjusted to fine-tune the algorithm's behavior.

## Random biased sampling:

Random biased sampling is a dynamic load balancing algorithm [20] employed in distributed systems and cloud computing environments.

The primary goal of this algorithm is to evenly distribute tasks or requests among available resources while considering their current workloads.

It prevents overloading heavily used resources and ensures that underutilized resources are brought into play.

It is particularly useful in situations where resource capabilities or workloads are not uniform.

* + 1. *Key Components and Working Principles:*
       - *Resource Monitoring:* The algorithm continuously monitors the resources in the system, collecting data on their current workloads. This data is typically based on metrics like CPU utilization, memory usage, or network load.
       - *Random Selection with Bias:* When a new task or request needs to be assigned to a resource, the algorithm employs a random selection process. However, this selection process is biased in favor of resources with lower workloads. This bias aims to prevent overloading already busy resources and balance the workload more evenly.
       - *Load Balancing Metric:* The algorithm calculates a load balancing metric for each resource, often denoted as "LB," which may be expressed as:

LB = f(Resource workload)

Here, "f" represents a function that calculates the load balancing score based on the resource's workload. A lower workload results in a higher load balancing score, making it more likely for that resource to be selected.

* + - * *Task Assignment:* Tasks or requests are assigned to resources based on the load balancing score. Resources with lower workloads are favored, but the selection remains random to introduce an element of unpredictability.
    1. *Mathematical Expression:*

The mathematical expression for the load balancing metric (LB) could be as simple as:

LB = 1 / (1 + Resource workload

In this expression, a resource with a higher workload (indicating higher resource utilization) will have a smaller load balancing score, making it less likely to be selected. Conversely, a resource with a lower workload will have a higher score, increasing its chances of being chosen.

## Round Robin:

Round Robin [18] is a popular and straightforward load balancing algorithm used in cloud computing environments and distributed systems.

Its primary objective is to distribute incoming tasks or requests equally among available resources in a cyclic manner.

The Round Robin algorithm ensures that each resource has an equal opportunity to serve incoming requests.

* + 1. *Key Components and Working Principles:*
       - *Resource Queue:* The algorithm maintains a queue of available resources, such as servers or virtual machines.
       - *Cyclic Assignment:* Tasks or requests are assigned to resources in a cyclical order. Each new task is assigned to the next resource in the queue.
       - *Load Balancing Metric:* The Round Robin algorithm doesn't consider resource workloads or capabilities when making assignments. Instead, it follows a strict cyclic order.
       - *Request Handling:* Resources process assigned tasks in the order they are received, allowing each resource to have an equal share of the workload.
    2. *Mathematical Expression:*

The Round Robin algorithm is relatively simple and doesn't require a complex mathematical expression for load balancing decisions. Assignments are made in a round-robin

fashion, so the mathematical expression for this algorithm can be summarized as:

Next\_resource = (current\_resource + 1) % total\_resources

In this expression, "current\_resource" represents the resource currently processing a task, "total\_resources" is the total number of available resources, and "Next\_resource" is the resource to which the next task will be assigned.

## Max-Min:

The Max-Min algorithm [15] is a static load balancing technique designed to evenly distribute the workload among resources in a cloud computing environment.

The primary objective of this algorithm is to maximize the minimum utilization among resources, ensuring that no resource is heavily loaded while others have spare capacity. It is particularly useful when resource capabilities are known in advance or when resources have varying capacities.

* + 1. *Key Components and Working Principles:*
       - *Resource Specification:* To implement the Max-Min algorithm, resource specifications are required, indicating the capacity and capabilities of each resource. These specifications can include factors like CPU power, memory, bandwidth, etc.
       - *Task Assignment:* The algorithm assigns incoming tasks or requests to resources by selecting the resource with the maximum available capacity, also known as the "largest-minimum" principle. The goal is to maximize the minimum capacity of the resources, ensuring that no resource becomes a bottleneck.
       - *Load Balancing Metric:* The load balancing metric used by the Max-Min algorithm is the available capacity of each resource. This can be expressed mathematically as:

Capacity(Resource\_i) = C\_i

Where C\_i represents the available capacity of resource i.

* + - * *Task Assignment Procedure:* When a new task or request arrives, the algorithm selects the resource with the highest available capacity (maximum C\_i) and assigns the task to that resource.
    1. *Mathematical Expression:*

The Max-Min algorithm primarily relies on the "largest- minimum" principle, which can be expressed mathematically as:

Assign(Task) = arg max (min(C\_i)), for all i

Here, "Assign(Task)" represents the resource to which the task is assigned, and "arg max (min(C\_i))" denotes the resource with the maximum minimum available capacity among all resources.

## Min-min:

The Min-Min algorithm [7] is a popular heuristic approach employed in cloud computing environments to optimize task scheduling. It aims to minimize the makespan, which is the total time taken to complete all tasks, by assigning tasks to suitable resources based on their estimated execution times. The Min-Min algorithm is a variant of the Shortest Job First (SJF) scheduling algorithm.

* + 1. *Working Principles:*
       - *Task Estimation:* Min-Min begins by estimating the execution time for each pending task. This estimation can be based on historical data, task characteristics, or other relevant metrics.
       - *Resource Selection:* The algorithm selects resources for task assignment by identifying the resource with the shortest expected completion time for each task.
       - *Task Assignment:* Tasks are assigned to resources with the minimum expected execution time, which minimizes the makespan and enhances overall system efficiency.
       - *Dynamic Reevaluation:* The algorithm reevaluates and updates task assignments as tasks are completed and new tasks arrive. This dynamic approach ensures that the system can adapt to changing workloads.
    2. *Mathematical Expression:*

The Min-Min algorithm does not rely on a specific mathematical expression for task assignment, but rather on the estimation of execution times for each task. However, the selection of resources for task assignment can be expressed using the following mathematical representation:

Let T\_i be the set of tasks, and R\_j be the set of resources. The expected execution time of task T\_i on resource R\_j can be expressed as EET(T\_i, R\_j).

The Min-Min algorithm selects the resource R\_j for task T\_i that minimizes the expected execution time:

R\_i = argmin(R\_j ∈ R) {EET(T\_i, R\_j)}

In this expression, R\_i represents the selected resource for task T\_i, and argmin denotes the resource in the set R that minimizes the expected execution time.

## Opportunistic algorithm:

Opportunistic load balancing algorithms [2] are designed to adapt to changing workloads and resource availability in cloud computing environments.

These algorithms take advantage of favorable conditions or opportunities to optimize resource allocation and task distribution.

* + 1. *Key Components and Working Principles:*
       - *Resource Monitoring:* Continuous monitoring of system resources is a fundamental aspect of opportunistic load balancing. This monitoring gathers real-time data on resource utilization, which

includes metrics such as CPU usage, memory availability, and network latency.

* + - * *Opportunistic Decision-Making:* The algorithm makes decisions on task or workload assignment based on the real-time data it collects. It actively looks for opportunities where resources are underutilized or available for immediate use.
      * *Load Balancing Metric:* A load balancing metric (often denoted as "LB") is used to assess the suitability of a resource for task assignment. This metric can be expressed mathematically, taking into account resource utilization metrics:

LB = f(CPU\_utilization, Memory\_utilization, Network\_latency)

* + - * The specific function "f" used in this expression can vary based on the algorithm's design and the criteria chosen to define opportunities. A lower value for the load balancing metric typically indicates a better opportunity.
      * *Dynamic Resource Allocation:* Resources may be allocated dynamically to tasks or workloads based on the availability of resources and the specific opportunities identified by the algorithm.
    1. *Mathematical Expression:*

A mathematical expression for the load balancing metric (LB) can be based on resource utilization metrics and weighted according to the algorithm's priorities.

For example:

LB = (α \* CPU\_utilization + β \* Memory\_utilization + γ \* Network\_latency)

In this expression, α, β, and γ represent weights that can be adjusted based on the specific goals of the load balancing algorithm. Different weights reflect the algorithm's preference for different resources or conditions.

# COMPARATIVE STUDY TABLE

|  |  |  |  |
| --- | --- | --- | --- |
| Load Balancing  Algorithm | Static/ Dynamic | Merits | Demerits |
| Round Robin | Static | Fixed time slice Better performance  for short CPU bursts | Larger tasks takes more time  for completion |
| Max-Min | Static | It works better as the requirements  are known in prior | Takes long time for task completion |

challenges may arise, especially when handling a diverse array of nodes in the system.

|  |  |  |  |
| --- | --- | --- | --- |
| Min-Min | Static | Smallest completion time Gives best results for small tasks | Smallest completion time Gives best results  for small tasks |
| Opportunis tic Load Balancing | Static | Improved performance  Resource utilization | Takes more time for  task completion |
| Ant Colony Optimizati on | Dynamic | Computationall y fast  Minimizes make span | Search takes long  time Complex |
| Honey Bee Foraging | Dynamic | Reduced response time  Increases throughput | Low priority  load takes more time |
| Biased Random Sampling | Dynamic | Improved performance Improved  resource utilization | Response time is more |
| Active clustering | Dynamic | Similar nodes are grouped together | when there is an increase in variety of nodes the performanc e of the  algorithm is poor |

The choice of load balancing algorithm should align with the specific cloud computing environment, task characteristics, and resource dynamics. This research equips practitioners and researchers with insights into algorithm strengths and limitations, facilitating informed decisions for their cloud deployments. Overall, the findings contribute to the ongoing efforts to enhance cloud services, resource management, and user experiences in the ever-evolving technological landscape.

**5. REFERENCES**

* 1. Hashem, Walaa, Heba Nashaat, and Rawya Rizk. "Honey bee based load balancing in cloud computing." *KSII Transactions on Internet & Information Systems* 11.12 (2017).
  2. Khair, Younes, and Haouari Benlabbes. "Opportunistic Load Balancing for Virtual Machines Scheduling in a Cloud Environment." *Engineering Proceedings* 29.1 (2023): 1.

In conclusion, investigation

# CONCLUSION

this research paper presents a thorough of load balancing algorithms in cloud

1. Ebadifard, Fatemeh, Seyed Morteza Babamir, and Sedighe Barani. "A dynamic task scheduling algorithm improved by load balancing in cloud computing." *2020 6th International Conference on Web Research (ICWR)*. IEEE, 2020.
2. Sharma, Parul, and Abhilasha Sharma. "Honeybee

computing, encompassing both static and dynamic approaches. Load balancing is a vital component in ensuring efficient resource utilization, minimizing response times, and optimizing overall system performance.

Static algorithms such as Round Robin, Max-Min, Min-Min, and Opportunistic Load Balancing are beneficial when system requirements are well-known. Round Robin offers equitable task distribution, Max-Min aims to maximize resource utilization, Min-Min prioritizes minimal completion times, and Opportunistic Load Balancing enhances performance. However, these methods have limitations, such as longer task completion times or potential resource underutilization.

Dynamic algorithms, including Ant Colony Optimization, Honey Bee Foraging, Biased Random Sampling, and Active Clustering, adapt to variable workloads and resource availability. Ant Colony Optimization minimizes make span, Honey Bee Foraging reduces response time, Biased Random Sampling improves resource utilization, and Active Clustering organizes resources effectively. Nevertheless,

Inspired Load Balancing Algorithm for Cloud Computing Environment." *Available at SSRN 4025366* (2022).

1. Mäkeläinen, Marko, et al. "Algorithms for opportunistic load balancing cognitive engine." *2014 9th International Conference on Cognitive Radio Oriented Wireless Networks and Communications (CROWNCOM)*. IEEE, 2014.
2. Deepa, T., and Dhanaraj Cheelu. "A comparative study of static and dynamic load balancing algorithms in cloud computing." *2017 International Conference on Energy, Communication, Data Analytics and Soft Computing (ICECDS)*. IEEE, 2017.
3. Arshad Ali, Syed, and Mansaf Alam. "Resource-Aware Min-Min (RAMM) Algorithm for Resource Allocation in Cloud Computing Environment." *arXiv e-prints* (2018): arXiv-1803.
4. Chen, Huankai, et al. "User-priority guided Min-Min scheduling algorithm for load balancing in cloud computing." *2013 National Conference on Parallel computing technologies (PARCOMPTECH)*. IEEE, 2013.
5. Muthusamy, Geetha, and Suganthe Ravi Chandran. "Cluster-based task scheduling using K-means clustering for load balancing in cloud datacenters." *Journal of Internet Technology* 22.1 (2021): 121-130.
6. Li, Youguo, and Haiyan Wu. "A clustering method based on K-means algorithm." *Physics Procedia* 25 (2012): 1104-1109.
7. Hasan, MD Mehedi, and Sungoh Kwon. "Cluster-based load balancing algorithm for ultra-dense heterogeneous networks." *IEEE Access* 8 (2019): 2153-2162.
8. Duan, Ping, and A. I. Yong. "Research on an improved ant colony optimization algorithm and its application." International Journal of Hybrid Information Technology 9.4 (2016): 223-234.
9. Nishant, Kumar, et al. "Load balancing of nodes in cloud using ant colony optimization." 2012 UKSim 14th international conference on computer modelling and simulation. IEEE, 2012.
10. Nasir, Husna Jamal Abdul, Ku Ruhana Ku-Mahamud, and Aniza Mohamed Din. "Load balancing using enhanced ant algorithm in grid computing." 2010 Second International Conference on Computational Intelligence, Modelling and Simulation. IEEE, 2010.
11. Suntharam, S. M. S. "Load balancing by max-min algorithm in private cloud environment." Int. J. Sci. Res.(IJSR) 4 (2013): 438.
12. Kaur, Rajwinder, and Pawan Luthra. "Load balancing in cloud system using max min and min min algorithm." International Journal of Computer Applications 975 (2014): 8887.
13. Hung, Tran Cong, et al. "MMSIA: improved max-min scheduling algorithm for load balancing on cloud computing." Proceedings of the 3rd International Conference on Machine Learning and Soft Computing. 2019.
14. Hidayat, Taufik, Yasep Azzery, and Rahutomo Mahardiko. "Load balancing network by using round Robin algorithm: a systematic literature review." Jurnal Online Informatika 4.2 (2019): 85-89.
15. Mohapatra, Subasish, Subhadarshini Mohanty, and K. Smruti Rekha. "Analysis of different variants in round robin algorithms for load balancing in cloud computing." International Journal of Computer Applications 69.22 (2013): 17-21.
16. Rahmeh, O. Abu, P. Johnson, and A. Taleb-Bendiab. "A dynamic biased random sampling scheme for scalable and reliable grid networks." INFOCOMP journal of computer science 7.4 (2008): 1-10.
17. Manakattu, Sheeja S., and SD Madhu Kumar. "An improved biased random sampling algorithm for load balancing in cloud based systems." Proceedings of the International Conference on Advances in Computing, Communications and Informatics. 2012.