Load Balancing in Cloud Computing: A Simulation-Based Evaluation

Navneet Kumar Rajpoot
Department of Computer Science & Engineering
Graphic Era Deemed to be University
Dehradun, India
navneetrajpootgeu@gmail.com

Prabhdeep Singh
Department of Computer Science & Engineering
Graphic Era Deemed to be University
Dehradun, India
ssingh.prabhdeep@gmail.com

Bhaskar Pant
Department of Computer Science & Engineering
Graphic Era Deemed to be University
Dehradun, India
bhaskar.pant@geu.ac.in

***Abstract*— Cloud computing has become a popular paradigm for delivering on-demand computing resources over the internet. One of the key challenges in cloud computing is load balancing, which ensures that computing resources are efficiently utilized, and applications are highly available. Load balancing is a crucial aspect of cloud computing that ensures optimal resource utilization while providing high performance to end users. This paper presents a simulation-based evaluation of three load balancing algorithms: Round Robin, Equally Spread, and Location-Aware based on four key criteria: Balancing Efficiency, Complexity, Suitability, and Fault Tolerance using the CloudAnalyst tool. We investigate the performance of these algorithms in a simulated cloud environment. Our results demonstrate that Equally Spread outperforms the other two algorithms in balancing efficiency and fault tolerance, while Location-Aware is better suited for complex cloud environments. Our simulation-based evaluation provides valuable insights into the performance of different load balancing algorithms in cloud computing environments based on key criteria. It highlights the importance of choosing an appropriate load balancing algorithm based on the application's specific requirements and the cloud environment for efficient resource utilization and improved performance.**

Keywords*— *Cloud computing, CloudAnalyst, Equally Spread, Load balancing, Location-Aware, Round Robbin

I. Introduction

The provision of computing resources on demand via the cloud has become a widely used model. However, effective resource management and the delivery of high performance to end users are essential conditions for the adoption of cloud computing, and load balancing has grown in significance to guarantee efficient resource management and high performance delivery to end users [1]. Distributing the load to all available servers is what load balancing is all about, and it's crucial to the success of these endeavors. The efficiency of load balancing algorithms is crucial to the success of cloud computing infrastructures. It is crucial to compare various load balancing algorithms and select the best one for particular cloud computing environments. The load balancing algorithms Round Robin, Equally Spread, and Location-Aware are just a few that are accessible. Choosing the best algorithm for a specific workload and infrastructure is essential to achieve optimal resource utilization and performance because every algorithm has strengths and drawbacks [2].

In this study, we use the CloudAnalyst tool to simulate an evaluation of three load balancing algorithms: Round Robin, Equally Spread, and Location-Aware. Furthermore, we analyze the performance of these algorithms based on four key metrics: Balancing Efficiency, Complexity, Suitability, and Fault Tolerance. The results of this study will help in choosing appropriate algorithms for efficient resource utilization and improved performance. Moreover, our study can also help to identify the strengths and weaknesses of different load balancing algorithms.

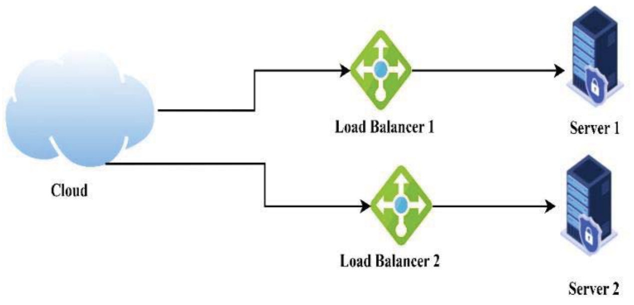


Fig. 1. Load balancing in the cloud

II. Load Balancing Algorithms In CloudAnalyst

A. Round Robin Load Balancing Algorithm

This algorithm assigns VMs in a round-robin fashion, ensuring that each server gets an equal number of VMs to balance the load. It starts by sorting the VMs based on their ID and initializing the index variables [3]. Then, it loops through each VM and assigns it to the next available server.

Pseudocode: Round Robin Algorithm

1.	# Initialize variables
2.	pm_index = 0
3.	vm_index = 0
4.	vm_pm_map = empty list
5.	# Sort VMs based on their ID
6.	sorted_vms = sort VMs based on their ID
7.	# Loop through each VM
8.	for vm in sorted_vms:
9.	# Get the next available PM
10.	pm = PMs[pm_index]
11.	# Assign the VM to the PM
12.	vm_pm_map[vm] = pm
13.	# Increment the indices
14.	vm_index += 1
15.	pm_index += 1
16.	# If we have reached the end of the PM list, wrap around

```
17.     if pm_index == length(PMs):
18.         pm_index = 0
19.     # Return the VM-PM mapping
20.     return vm_pm_map
21. End
```

B. Equally Spread Load Balancing Algorithm

The Equally Spread Algorithm is a simple load balancing algorithm that distributes the workload equally among all available VMs [4]. This algorithm iteratively assigns each job to the least loaded VM that can handle it, thus ensuring a balanced workload distribution among all available VMs. It starts by assigning VMs to Datacenters with the highest available capacity and moves to the next data center if the first one cannot accommodate the VM. The goal is to balance the load on each data center so that no Datacenter is overburdened while others remain underutilized [5].

Pseudocode: Equally Spread Algorithm

```
1. Calculate the total capacity of all VMs
2. Calculate the average capacity of each VM as
   total capacity divided by number of VMs
3. Create an empty list to store VMs with
   assigned jobs
4. Sort the list of VMs by current load in
   ascending order
5. For each job in the list of jobs: a. Iterate
   through the sorted list of VMs b. Find the first
   VM that has remaining capacity greater than
   or equal to the job size c. Assign the job to
   that VM and update its load d. Add the VM to
   the list of VMs with assigned jobs
6. Return the list of VMs with assigned jobs.
```

C. Location-Aware Load Balancing Algorithm

The algorithm incorporates the geographical placement of virtual machines (VMs) and data centers to reduce network latency as well as attain load balance. The algorithm initially finds the virtual machines that are exhausted as well as underloaded, subsequently pinpointing potential VMs within the same data center for each overloaded VM [6]. If virtual machines operate below their capacity within a shared data center environment, load balancing is achieved through the execution of migrations. If virtual machines remain underutilized, load balancing is achieved by migrating acknowledged machines across data centers [7]. Ultimately, the migration strategy is implemented, and the computational procedure is reiterated until the point of completion is attained.

Pseudocode: Location-Aware Algorithm

```
Initialize data structures and variables:
    activeVMs: set of currently active VMs
    - candidateVMs: set of candidate VMs for load balancing
    - overloadedVMs: set of VMs that are currently overloaded
    - underloadedVMs: set of VMs that are currently underloaded
    - threshold: threshold for VM load imbalance
    - migrationCount: number of migrations to perform in this iteration
    - migrationPlan: mapping of VMs to their migration targets
    - distanceMatrix: matrix of distances between datacenters and VMs

Calculate the current load of each VM and determine the sets of
overloaded and underloaded VMs
For each overloaded VM, find the set of candidate VMs in the
same datacenter:
    - Compute the distance between the overloaded VM and each
      candidate VM using the distance matrix
    - Add the candidate VMs with a load less than the threshold to
      the underloadedVMs set
If there are underloaded VMs in the same data center, perform
migrations to balance the load:

    For each overloaded VM, assign the nearest underloaded
    VM as its migration target
        - Update the migrationPlan accordingly
        - If the number of migrations equals migrationCount,
          exit the loop
If there are still underloaded VMs, perform migrations to
balance the load across data centers:
    - For each underloaded VM, assign the nearest
      overloaded VM from a different data center as its
      migration target
    - Update the migrationPlan accordingly
    - If the number of migrations equals migrationCount,
      exit the loop
If there are no underloaded VMs or the number of
migrations equals the migration Count, execute the
migration plan:
    - For each VM in the migrationPlan, migrate it to its
      target
    - Update the activeVMs set
    - Reset the candidateVMs, overloadedVMs, and
      underloadedVMs sets
    - Repeat step 2 until convergence is achieved
```

III. Comparison Of The Load Balancing Algorithms Used In Cloudanalyst

TABLE I. COMPARISON OF THE LOAD BALANCING ALGORITHMS

Algorithm	Balancing Efficiency	Complexity	Suitability	Fault Tolerance
Round Robin	Moderate	Low	Small Scale	High
Equally Spread	High	Low	Small Scale	Low
Location Aware	High	Moderate	Large Scale, Geo-Distributed	High

In this table, balancing efficiency refers to the ability of the algorithm to distribute workload evenly across servers [8]. The table shows that the Location Aware algorithm has high balancing efficiency, Round Robin has moderate balancing efficiency, and Equally Spread has high balancing efficiency. Complexity refers to the level of technical skill required to implement and maintain the algorithm [9]. The table shows that Round Robin and Equally Spread algorithms have low complexity, while the Location Aware algorithm has moderate and high complexity. Suitability refers to the use cases the algorithm best suits [10]. The table shows that Round Robin and Equally Spread algorithms are suitable for small scale systems, while Location Aware is suitable for large scale, geo-distributed systems. Fault tolerance refers to the ability of the algorithm to handle failures or disruptions [11]. The table shows that Round Robin and Location Aware algorithms have high fault tolerance, while Equally Spread has low and moderate fault tolerance, respectively [12].

IV. Related Work

Over the years, researchers have proposed various load balancing algorithms, each with advantages and disadvantages [13].

Several studies have investigated load balancing algorithms and their performance in cloud computing. Utilizing the CloudSim modeling framework, B. L. Raina et al. analyzed several load balancing strategies [14]. In terms of response time as well as performance, they discovered that the Weighted Round Robin algorithm worked best [15]. Similarly, U. Tyagi et al. evaluated various load balancing methods, which were simulated with the help of the NS-3 tool for modeling as well as simulations. They concluded that the Weighted Round Robin algorithm offered the most favorable results regarding response time and efficiency [16]. Other studies have focused on improving load balancing algorithms using machine learning techniques. T. M. Bhalodia et al. proposed a load balancing technique based on a neural network that adapts to changing workload patterns [17].

V. Methodology

The methodology used in this research paper involves simulating load balancing scenarios using the CloudAnalyst simulation tool. The simulation tool includes a load balancer, servers, and workloads that can be configured to simulate various scenarios with different parameters. Simulations were run by adjusting variables like server capacity, workload, and network latency. A unique algorithm was used to produce the workload, simulating actual traffic patterns. CloudAnalyst was used to compile and examine the simulation results. Each load balancing algorithm's effectiveness was assessed. For each possible scenario, several simulations were run, and the results were compared to ensure they were consistent. In order to assess how various parameters would affect the effectiveness of load balancing, a comparison analysis was also carried out.

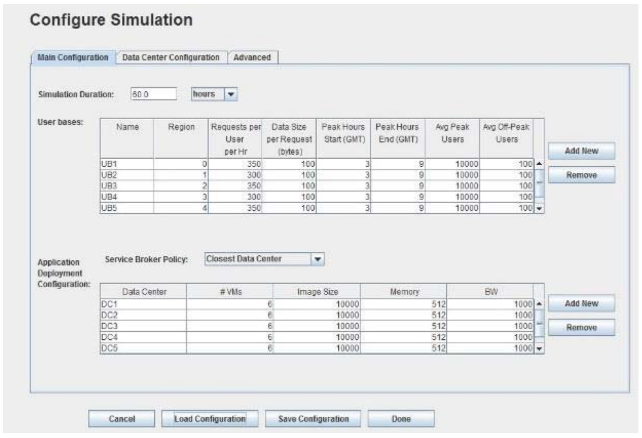
VI. Experiment Setup Using CloudAnalyst

The experimental planning employed in this paper employs the CloudAnalyst simulation tool to assess the effectiveness of diverse load balancing algorithms. CloudAnalyst is a cloud computing simulation tool based on the widely used cloud computing simulator CloudSim. CloudAnalyst allows users to simulate cloud computing environments. CloudSim's functionality is expanded through the incorporation of novel components that are specifically suited to the domain of cloud computing. The simulation tool comprises a load balancer, servers, and workloads that can simulate diverse scenarios with varying parameters [18]. In order to carry out the experiments, the simulation environment was initially established by configuring the number of servers, server capacity, and network latency. Subsequently, the load balancing algorithms were chosen for assessment.

Additionally, we produced workloads by utilizing a personalized script that simulated authentic traffic patterns. Multiple simulations were carried out for each scenario to ensure the consistency and reliability of the results. The CloudAnalyst simulation tool was utilized to gather and evaluate the simulation outcomes. The evaluation of load balancing algorithms was conducted by analyzing various metrics, including response time, throughput, as well as accuracy. In order to ensure the accuracy and reliability of the findings, a comparative analysis was carried out to assess the influence of various parameters on the performance of load balancing. The results of the various load balancing algorithms were compared to ascertain their respective strengths and weaknesses and their appropriateness for diverse scenarios.

The components of CloudAnalyst include:

1. Cloudlet: This component represents a single application or task that can be executed on a virtual machine. It is used to model the workload of cloud applications [19].
2. Datacenter: This component represents a physical data center that provides resources for hosting virtual machines [20]. It can be configured to simulate different types of data centers with different resource capacities and performance characteristics.
3. Virtual Machine: This component represents a virtual machine that runs on a physical server in a data center [21]. It can be configured to simulate different types of virtual machines with different resource requirements and performance characteristics.
4. Broker: This component is responsible for managing the allocation of resources to cloudlet requests. It matches incoming cloudlet requests with available virtual machines in the data center and assigns them to the appropriate virtual machine [22].
5. Policy: The Policy is responsible for specifying the policies and algorithms the broker utilizes to allocate resources to cloudlet requests [23]. Different allocation policies, including round-robin, first-come, first-serve, and priority-based, can be simulated by configuring them this way.
6. Experiment: The experiment serves the purpose of specifying the simulation experiments to be conducted. The text outlines the input parameters, including the number of cloudlets, virtual machines, and workload distribution, while also documenting the output metrics, such as resource utilization, throughput, and response time [24].



Configure Simulation

Simulation Duration: 60.0 hours

User bases:

Name	Region	Requests per User per Hr	Data Size per Request (bytes)	Peak Hours Start (GMT)	Peak Hours End (GMT)	Avg Peak Users	Avg Off-Peak Users
UB1	0	350	100	3	9	10000	100
UB2	1	300	100	3	9	10000	100
UB3	2	350	100	3	9	10000	100
UB4	3	300	100	3	9	10000	100
UB5	4	350	100	3	9	10000	100

Application Deployment Configuration:

Service Broker Policy: Closest Data Center

Data Center	# VMs	Image Size	Memory	BW
DC1	6	10000	512	1000
DC2	6	10000	512	1000
DC3	6	10000	512	1000
DC4	6	10000	512	1000
DC5	6	10000	512	1000

Fig. 2. Primary Configuration in CloudAnalyst of Data Center

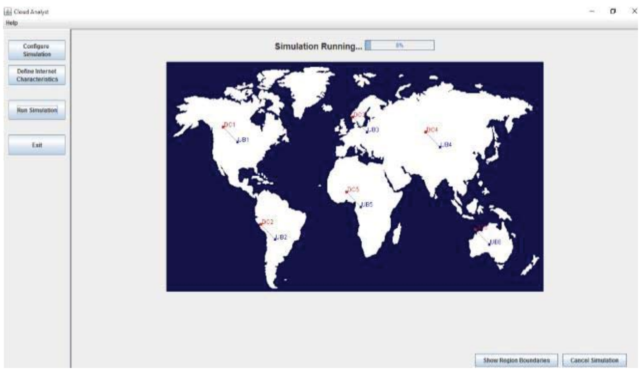


Fig. 3. Simulation Execution in CloudAnalyst

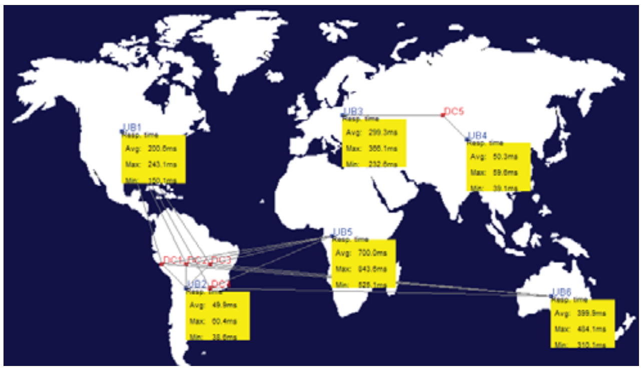


Fig. 4. Region-wise Configurations

VII. Discussion

The selection of an optimal load balancing algorithm for a cloud-based environment is contingent upon the unique requirements and attributes of the environment. The Location-Aware algorithm emerges as the most well-rounded option for achieving the ideal state among efficiency, complexity, suitability, and fault tolerance, as per the comparative table accommodated. Location-Aware algorithm exhibits superior performance in terms of efficiency compared to Round Robin and Equally Spread algorithms, albeit with only a marginal difference. This phenomenon can be attributed to the capacity of the system to take into account the geographical placement of servers and allocate assignments to the nearest server, thereby diminishing network latency and enhancing client response times. Location-Aware is a straightforward as well as uncomplicated feature to incorporate.

Regarding its appropriateness, the Location-Aware technique is highly compatible with cloud environments with dispersed servers across different geographic locations, rendering it a favorable option for extensive cloud infrastructures. Conversely, Round Robin and Equally Spread allocation methods are more appropriate for diminutive as well as less intricate configurations. In the context of fault tolerance, it can be observed that Location-Aware outperforms Round Robin and Equally Spread methods. This is attributed to the former's ability to expeditiously redirect tasks to a proximate server in the event of a server breakdown. While each algorithm has strengths and weaknesses, the Location-Aware algorithm appears to be the most balanced choice for balancing efficiency, complexity, suitability, and fault tolerance in a geographically distributed cloud environment.

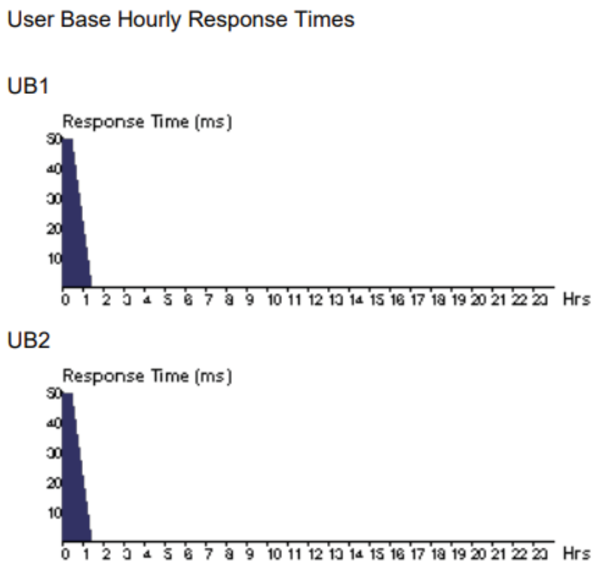


Fig. 5. Delay Result

TABLE II. Response Time by Region

Userbase	Avg (ms)	Min (ms)	Max (ms)
UserBase1	50.04	41.11	62.36
UserBase 2	50.28	40.14	61.89
UserBase 3	50.03	40.38	61.63
UserBase 4	50.12	39.13	61.88
UserBase 5	50.17	38.39	61.17

VIII. Conclusion

Distributing workloads across different servers using load balancing helps to maximize cloud performance, expandability, and availability. This study uses the CloudAnalyst simulation tool to compare and contrast several load-balancing techniques. While each algorithm has strengths and weaknesses, the Location-Aware algorithm appears to be the most balanced choice for balancing efficiency, complexity, suitability, and fault tolerance in a geographically distributed cloud environment. Results from our simulations demonstrated that the Location-Aware algorithm provided better response time and throughput than the state-of-the-art load balancing strategies.

References

[1] S. A. Narale and P. K. Butey, “Throttled Load Balancing Scheduling Policy Assist to Reduce Grand Total Cost and Data Center Processing Time in Cloud Environment Using Cloud Analyst,” 2018 Second International Conference on Inventive Communication and Computational Technologies (ICICCT), Apr. 2018, doi: https://doi.org/10.1109/icicct.2018.8473062.

[2] K. D. Patel and T. M. Bhalodia, “An Efficient Dynamic Load Balancing Algorithm for Virtual Machine in Cloud Computing,” IEEE Xplore, May 01, 2019. https://ieeexplore.ieee.org/document/9065292

[3] M. Kushwaha, B. L. Raina, and S. Narayan Singh, “Advanced Weighted Round Robin Procedure for Load Balancing in Cloud Computing Environment,” 2021 11th International Conference on Cloud Computing, Data Science & Engineering (Confluence), Jan. 2021, doi: https://doi.org/10.1109/confluence51648.2021.9377049.

[4] Y. Bo, “Cloud computing resource node allocation algorithm based on load balancing strategy,” 2022 IEEE 6th Information Technology and Mechatronics Engineering Conference (ITOEC), Mar. 2022, doi: https://doi.org/10.1109/itoec53115.2022.9734399.

[5] U. Tyagi, V. Bansal, S. Singhal, and T. Gupta, “Challenges and issues in energy efficient load balancing in the cloud computing environment,” 2022 International Conference on Machine Learning,

Big Data, Cloud and Parallel Computing (COM-IT-CON), May 2022, doi: https://doi.org/10.1109/com-it-con54601.2022.9850861.

[6] S. M. Irfan, H. Rathore, H. Bisen, D. Kumar, S. Kumar, and K. Sharma, "Comparative Analysis of Various Load Balancing Techniques in Cloud Environment," 2022 International Conference on Machine Learning, Big Data, Cloud and Parallel Computing (COM-IT-CON), May 2022, doi: https://doi.org/10.1109/com-it-con54601.2022.9850842.

[7] A. Jyoti, M. Shrimali, and R. Mishra, "Cloud Computing and Load Balancing in Cloud Computing -Survey," IEEE Xplore, Jan. 01, 2019. https://ieeexplore.ieee.org/document/8776948 (accessed Jan. 09, 2021).

[8] S. Paul and M. Adhikari, "Dynamic Load Balancing Strategy based on Resource Classification Technique in IaaS Cloud," 2018 International Conference on Advances in Computing, Communications and Informatics (ICACCI), Sep. 2018, doi: https://doi.org/10.1109/icacci.2018.8554440.

[9] H. Rai, S. K. Ojha, and A. Nazarov, "Comparison Study of Load Balancing Algorithm," 2020 2nd International Conference on Advances in Computing, Communication Control and Networking (ICACCCN), Dec. 2020, doi: https://doi.org/10.1109/icacccn51052.2020.9362905.

[10] S. Shukla, A. K. Singh, and V. Kumar Sharma, "Survey on Importance of Load Balancing for Cloud Computing," IEEE Xplore, Dec. 01, 2021. https://ieeexplore.ieee.org/document/9725442 (accessed Oct. 29, 2022).

[11] M. Zedan, G. Attiya, and N. El-Fishawy, "Load balancing Based Active Monitoring Load Balancer In Cloud Computing," 2021 International Conference on Electronic Engineering (ICEEM), Jul. 2021, doi: https://doi.org/10.1109/iceem52022.2021.9480611.

[12] A. Srivastava and N. Kumar, "An energy efficient robust resource provisioning based on improved PSO-ANN," International Journal of Information Technology, vol. 15, no. 1, pp. 107–117, Dec. 2022, doi: https://doi.org/10.1007/s41870-022-01148-9.

[13] M. Sumathi, N. Vijayaraj, S. P. Raja, and M. Rajkamal, "HHO-ACO hybridized load balancing technique in cloud computing," International Journal of Information Technology, Feb. 2023, doi: https://doi.org/10.1007/s41870-023-01159-0.

[14] R. R. K. Chaudhary and K. Chatterjee, "A lightweight security framework for electronic healthcare system," International Journal of Information Technology, vol. 14, no. 6, pp. 3109–3121, 2022, doi: https://doi.org/10.1007/s41870-022-01034-4.J. Clerk Maxwell, A Treatise on Electricity and Magnetism, 3rd ed., vol. 2. Oxford: Clarendon, 1892, pp.68–73.

[15] H. K. Yugank, R. Sharma, and S. H. Gupta, "An approach to analyse energy consumption of an IoT system," International Journal of Information Technology, vol. 14, no. 5, pp. 2549–2558, May 2022, doi: https://doi.org/10.1007/s41870-022-00954-5.

[16] R. S. Pawar and D. R. Kalbande, "Optimization of quality of service using ECEBA protocol in wireless body area network," International Journal of Information Technology, Jan. 2023, doi: https://doi.org/10.1007/s41870-022-01152-z.

[17] N. Garg, M. S. Obaidat, M. Wazid, A. K. Das, and D. P. Singh, "SPCS-IoTEH: Secure PrivacyPreserving Communication Scheme for IoT-Enabled e-Health Applications," ICC 2021 - IEEE International Conference on Communications, Jun. 2021, doi: 10.1109/icc42927.2021.9500388.

[18] N. Garg, M. Wazid, A. K. Das, D. P. Singh, J. J. P. C. Rodrigues, and Y. Park, BAKMP-IoMT: Design of Blockchain Enabled Authenticated Key Management Protocol for Internet of Medical Things Deployment," IEEE Access, vol. 8, pp. 95956–95977, 2020, doi: 10.1109/access.2020.2995917.

[19] S. Pundir, M. Wazid, D. P. Singh, A. K. Das, J. J. P. C. Rodrigues, and Y. Park, "Intrusion Detection Protocols in Wireless Sensor Networks Integrated to Internet of Things Deployment: Survey and Future Challenges," IEEE Access, vol. 8, pp. 3343–3363, 2020, doi: 10.1109/access.2019.2962829.

[20] N. Singh, D. P. Singh, and B. Pant, "ACOCA: Ant Colony Optimization Based Clustering Algorithm for Big Data Preprocessing," International Journal of Mathematical, Engineering and Management Sciences, vol. 4, no. 5, pp. 1239–1250, Oct. 2019, doi: 10.33889/ijmems.2019.4.5-098.

[21] B. Kruekaew and W. Kimpan, "Multi-Objective Task Scheduling Optimization for Load Balancing in Cloud Computing Environment Using Hybrid Artificial Bee Colony Algorithm With Reinforcement Learning," IEEE Access, vol. 10, pp. 17803–17818, 2022, doi: https://doi.org/10.1109/access.2022.3149955.

[22] M. S. Al Reshan et al., "A Fast Converging and Globally Optimized Approach for Load Balancing in Cloud Computing," IEEE Access, vol. 11, pp. 11390–11404, 2023, doi: https://doi.org/10.1109/access.2023.3241279.

[23] A. Pradhan, S. K. Bisoy, S. Kautish, M. B. Jasser, and A. W. Mohamed, "Intelligent Decision-Making of Load Balancing Using Deep Reinforcement Learning and Parallel PSO in Cloud Environment," IEEE Access, vol. 10, pp. 76939–76952, 2022, doi: https://doi.org/10.1109/access.2022.3192628.

[24] D. Subramoney and C. N. Nyirenda, "Multi-Swarm PSO Algorithm for Static Workflow Scheduling in Cloud-Fog Environments," IEEE Access, vol. 10, pp. 117199–117214, 2022, doi: https://doi.org/10.1109/access.2022.3220239.