An Ensemble Load Balancing Algorithm to Process the Multiple Transactions Over Banking

Raghunadha Reddi Dornala
Cloud Architect, USA

Abstract*—The banking industry has been transformed by cloud computing, which has provided scalable and cost-effective solutions for managing large volumes of transactions. However, as the number of transactions grow, the need for efficient load-balancing algorithms to ensure optimal utilization of cloud resources and improve system performance becomes critical. This paper proposes an ensemble cloud load-balancing (ECBA) algorithm specifically designed to process multiple banking transactions. The proposed algorithm combines the strengths of several load-balancing techniques to achieve a balanced distribution of transaction loads in various cloud servers. It considers factors such as transaction types, server capacities, and network conditions to make intelligent load distribution decisions. The algorithm dynamically adapts to changing workload patterns and optimizes resource allocation by leveraging machine learning and predictive analytics. A simulation environment that mimics the banking system's transaction processing workflow is created to evaluate the performance of the ensemble load balancing algorithm. Extensive experiments with various workload scenarios are conducted to assess the algorithm's effectiveness in load balancing, response time, resource utilization, and overall system performance. The results show that the proposed ECBA outperforms traditional banking load-balancing approaches. It reduces response time, improves resource utilization, and ensures every server is adequately funded with a few transactions. The algorithm's adaptability and scalability make it well-suited for handling dynamic and fluctuating workloads, thus providing a robust solution for processing multiple transactions in the banking sector.

Keywords*—Cloud computing; load balancing; ensemble algorithm; banking; transaction processing; resource utilization; response time; scalability

I. Introduction

Load balancing techniques are critical for ensuring efficient resource utilization and maintaining optimal performance in cloud computing environments [1] [2]. Cloud computing provides users with vast computational resources and services. Still, the dynamic nature of cloud environments, the variability of workloads, and the scale of resources present several load-balancing challenges. This section investigates the issues and challenges associated with cloud computing load-balancing techniques. Discuss the complexities introduced by cloud systems' distributed nature, workload variability, scalability, and fault tolerance and the importance of ensuring fairness and resource utilization [3]. Understanding these difficulties is critical for developing effective load-balancing mechanisms in cloud environments. Cloud computing environments typically comprise several geographically dispersed data centers or clusters [4]. Managing the load across these disparate resources is a difficult task. Load-balancing algorithms must consider network latency, bandwidth constraints, and the availability of resources across multiple locations. Coordination of workload distribution and efficient resource utilization in such environments is a significant challenge [5].

Because of the dynamic nature of user demands, cloud systems experience highly variable workloads. The load on cloud resources can fluctuate dramatically, necessitating real-time load-balancing techniques. Predicting and managing these workload variations is critical to avoid resource underutilization or overload situations. Load-balancing algorithms must consider historical and real-time workload data to make intelligent workload distribution decisions [6]. Cloud computing is intended to scale horizontally, allowing for adding or removing resources in response to demand. Load balancing techniques must be able to adjust workload distribution as resources scale up or down dynamically [7].

Furthermore, cloud systems are vulnerable to failures and faults. Load balancers should be fault-tolerant, detecting and redirecting workloads away from failed or degraded resources to ensure high availability and reliability. Load-balancing algorithms should strive for equity and fair resource distribution among users or applications. Maintaining user satisfaction and avoiding performance bottlenecks requires ensuring each user or application receives a fair share of resources. Balancing workload distribution while considering workload priorities, user requirements, and resource constraints, can be difficult.

In the fast-paced world of banking and finance, effective transaction management is critical for ensuring customer satisfaction, operational stability, and data security. Traditional load-balancing algorithms frequently fall short of distributing workloads evenly across resources, owing to the increasing reliance on digital transactions and the ever-increasing volume of data. This paper proposes an innovative ensemble load-balancing algorithm designed to optimize banking transaction processing. The proposed algorithm aims to improve system performance, ensure resource utilization, and provide a seamless customer experience by combining the strengths of multiple load-balancing techniques. The banking sector is critical to global economic activity, processing a wide range of transactions daily. As more customers use digital banking services, there is a greater need for seamless and quick transaction processing. Banks rely heavily on information technology systems and networks to meet these demands, making efficient load balancing critical for ensuring optimal performance.

The organization of this paper is as follows. Section II explains various cloud models applied on multiple datasets to analyze the performance. Section III describes the weighted Round Robin algorithm in the banking sector. Section IV presents the adaptive load balancing and its functionalities. The fifth section explains the proposed combined approach and its functionalities. Section VI gives the performance metrics used in this paper. Section VII shows the comparative performances of various existing and proposed algorithms. The final section provides the conclusion of the overall research work.

II. Literature Survey

X. Wei et al. [8] introduced the popularity-based position technique that maps data components and edge servers to retrieve the virtual coordinate in the plane. The proposed model performance is improved to tackle the load balancing between edge servers via offloading. Experiments show that the proposed model effectively reduces the average path length for data access, and load-balancing techniques provide better options for overloaded servers. T. Liu et al. [9] introduced the novel Q-networks that will allocate resources using resource computation. The main objective of novel q-networks used to decrease the latency over a long time. The performance of Novel q-networks shows better performance in terms of performance. S. Nath et al. [10] proposed an automated scheduling model that incorporates Deep Reinforcement Learning (DRL) and the Deep Deterministic Policy Gradient (DDPG) method. The DDPG extracts optimized models to develop multi-cell MEC systems by leveraging cooperation among neighboring MEC servers. When compared to DDPG, the existing model produced better results. J. Zhang et al. [11] discussed several comparative performances based on the security and privacy threats that edge computing can address. This work primarily focused on discussing various issues and factors that aid in developing several edge computing applications. The study also provides solutions for several privacy and security issues based on edge-related paradigms. T. Li et al. [12] introduced privacy grouping issues that reduce problems in edge clouds and thus reduce edge cloud maintenance. The grouping techniques carefully developed the optimized goal for two models such as tree-based hierarchical (TBH) and graph-based interconnected (GBI) edge clouds. The proposed model obtained better outcomes on two benchmark data sets. B. Pourghebleh et al. [13] introduced the meta-heuristic model that solves the VM unification issue compared with the existing models based on the influential factors. E. H. Houssein et al. [14] introduced a different model belongs to task scheduling that classifies the cloud applications based on the scheduling issues which is one or multiple objectives. S. K. Mishra et al. [15] proposed an iterative approach that optimized the metrics such as latency and energy consumption and unloading the work and associated VM for the execution of the task. If the particular edge center is not ready to provide the resources, the user's request will send to the other cloud system. B. Alankar et al. [16] developed a combined approach that solves various issues in load balancing based on HAProxy, clusters in VM, and cloud servers. Results show that the proposed model obtained better performance regarding load balancing metrics. A. A. Abdelltif et al. [17] proposed the SDN-based load balancing that reduces the high usage of resources and decreases the computation time. The proposed model executes the program on top of the SDN model and controls the tasks. The manager in this model maintains the transmission messages, maintains hosting pools, and checks the load status at peak time. M. A. Mukwevho et al. [18] presented a comparative survey of work on fault tolerance applications used in the cloud environment. A better future model is required to improve the proposed approach's performance. B. Cao et al. [19] detected the replacement issue by using the edge servers (ESs) in the IoV to design various objectives used to deploy the applications and measure the task loading, energy usage, deployment expenses, etc. The proposed model also solves the ES deployment issue. B. Lin et al. [20] introduced the GA-DPSO combined with the genetic approach that optimizes data transmission based on the proposed flow. GA-DPSO is the combined domains such as edge computing and cloud computing. S. Yang et al. [21] proposed that the issue belongs to VNFs on the combined platform and analyzed the VM traffic present in the VMs. It is an integrated approach that uses various technologies that help improve the detection of abnormal traffic in cloud servers and reduce the traffic by controlling the user's requests. J. Zhang et al. [22] introduced the model deployed in a cloud server with an advanced domain called vehicular networks. Several high-potential models, such as fiber-wireless (FiWi), improve the vehicular edge computing networks (VECNs), analyze the task loading, and measure the vehicle's delay. The proposed model obtained superior results compared with existing approaches.

III. Weighted Round Robin Algorithm for Banking Transactions

Maintaining unlimited banking transactions requires practical load-balancing algorithms that guarantee the network can handle many transactions without complex resources. The Weighted Round Robin (WRR) algorithm is a scheduling technique commonly used in banking systems to allocate resources for processing transactions. It aims to provide fairness and efficient utilization of resources based on predefined weights assigned to each transaction type. Here's an explanation of the WRR algorithm using equations:

A. *Define Inputs*

N: Number of transaction types.

W[i]: Weight assigned to each transaction type i, where i ranges from 1 to N.

Calculate the Weighted Round Robin Quantum (Q):

Calculate the total weight sum (TW) as the sum of all transaction weights:

$$\mathrm{TW} = \mathrm{W}[1] + \mathrm{W}[2] + \cdots \mathrm{W}[\mathrm{N}] \quad (1)$$

Set the Quantum (Q) as the least common multiple (LCM) of all transaction weights:

$$\mathrm{Q} = \mathrm{LCM}(\mathrm{W}[1] + \mathrm{W}[2] + \cdots \mathrm{W}[\mathrm{N}]) \quad (2)$$

B. *Initialize Variables*

Current_quantum: The current quantum being processed initially set to 0.

Current_transaction_type: The index of the currently selected transaction type initially set to 0.

C. *Transaction Scheduling*

- Iterate through the transactions in a loop:
- For each transaction, perform the following steps:
  - Increment the current_quantum by 1.
  - If current_quantum is equal to or exceeds the Quantum (Q), perform the following steps:
  - Reset current_quantum to 0.
  - Move to the next transaction type by incrementing current_transaction_type by 1.
  - If current_transaction_type exceeds the maximum index (N), set current_transaction_type back to 1.
  - Select the transaction type indicated by current_transaction_type for processing.

Output: The output of the WRR algorithm is the sequence of transaction types selected for processing, based on their assigned weights and the calculated quantum.

The Weighted Round Robin algorithm ensures that transactions with higher weights receive a proportionally higher share of the available processing resources. By defining appropriate weights for each transaction type, the algorithm can be customized to prioritize certain types of transactions or balance the workload across different transaction types.

IV. Adaptive Load Balancing (ALB) for Banking Transactions in Cloud Computing

ALB mainly focuses on transactions made by the users in the cloud server, which involves several distributed workload among multiple servers to advance performance and shows effective transactions. Here are the steps and equations involved in adaptive load balancing:

1) Measure server performance: The first step is to collect data on the performance of each server in the cloud environment. This can include metrics such as CPU utilization, memory usage, network bandwidth, and response time.

2) Determine the load balancing algorithm: Choose an appropriate load balancing algorithm based on the specific requirements of banking transactions. Some commonly used algorithms include round-robin, least connections, weighted round-robin, and least response time.

3) Define the criteria for load balancing: Establish the criteria for load balancing decisions. This can include thresholds for server utilization, response time, or other performance metrics. For example, if a server's CPU utilization exceeds a certain threshold, it may be considered overloaded.

4) Calculate server weights: If using a weighted load balancing algorithm, assign weights to each server based on its capacity and performance characteristics. This allows for more fine-grained control over the distribution of the workload.

5) Monitor server performance: Continuously monitor the performance of each server in the cloud environment. This can be done using monitoring tools or by collecting real-time performance metrics.

6) Evaluate server conditions: Compare the performance metrics of each server against the defined criteria for load balancing. If a server exceeds the thresholds or is underutilized, it may be a candidate for load redistribution.

7) Calculate the load factor: Calculate the load factor for each server based on its current workload and capacity. The load factor can be calculated using various equations, such as:

$$\text{Load Factor} = \frac{\text{Current Server Load}}{\text{Maximum Server Capacity}} \quad (3)$$

This equation gives a normalized value between 0 and 1, representing the relative load on each server.

8) Make load balancing decisions: Based on the load factors and the chosen load balancing algorithm, make decisions on redistributing the workload. This involves determining which server should receive new requests or reassigning existing requests from overloaded servers to underutilized ones.

9) Redirect requests: Implement the load balancing decisions by redirecting incoming requests to the selected servers. This can be achieved through DNS-based load balancing, where the DNS server maps the domain name to the IP address of the appropriate server.

10) Monitor and adjust: Continuously monitor the system performance and reevaluate load balancing decisions as the workload and server conditions change. Adjust the load balancing parameters, such as thresholds or weights, if necessary, to further optimize performance.

V. Weighted Round Robin Algorithm Combined with Adaptive Load Balancing in Cloud Banking Application

Cloud computing has transformed how businesses operate in today's digital age, and the banking sector is no exception. Cloud banking applications take advantage of cloud infrastructure's scalability, flexibility, and cost-effectiveness to provide customers with efficient and dependable services. However, as the number of users and transactions grow, ensuring optimal performance and load balancing becomes increasingly essential for the smooth operation of these applications. The WRR algorithm and Adaptive Load Balancing techniques can be used to address this challenge. This effective combination provides improved resource allocation, faster response times, and more efficient use of cloud resources.

WRR is a load-balancing algorithm that cyclically distributes incoming requests across multiple servers. Each server is given a weight that corresponds to its processing capacity. The algorithm considers these weights during load-balancing and distributes requests proportionally to each server. The WRR algorithm ensures that the workload is distributed relatively by assigning higher weights to more powerful servers, preventing any individual server from becoming overwhelmed.

The WRR algorithm is supplemented by Adaptive Load Balancing, which continuously monitors system performance metrics such as CPU utilization, memory usage, network traffic, and response times. The load balancer dynamically adjusts the weights assigned to each server based on these metrics. If a server begins to experience increased workloads or performance degradation, the load balancer can reduce its weight, redirecting traffic to other servers with lower workloads. If a server's performance or load improves, its importance can be increased, allowing it to handle more requests (see Fig. 1).

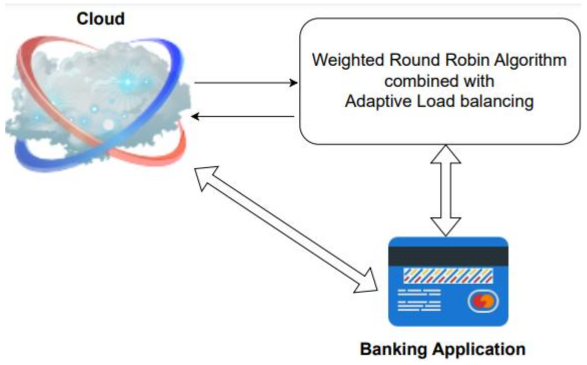
Cloud
Weighted Round Robin Algorithm combined with Adaptive Load balancing
Banking Application

Fig. 1. Proposed architecture.

Combining the WRR algorithm with Adaptive Load Balancing in a cloud banking application provides numerous benefits. For starters, it ensures that each server is used to its full potential, avoiding bottlenecks and improving overall system performance. Second, it enables the application to adapt to changing traffic patterns and allocate resources dynamically based on real-time conditions, resulting in faster response times and a better user experience. Finally, it improves the application's scalability by allowing additional servers to be easily added or removed from the pool without disrupting the load-balancing mechanism.

VI. Performance Metrics

Load balancing is a critical technique in cloud computing to distribute workloads across multiple servers or resources to optimize performance and ensure efficient resource utilization. The effectiveness of load-balancing algorithms is assessed using a variety of performance metrics. Here is some common load balancing performance metrics along with their equations:

1) Response Time (RT): It measures the time taken by a server to respond to a client request. Lower response time indicates better performance.

$$RT = (T_f - T_s) + T_w$$

Where T_f the time of the last response is received, T_s is the time of the first request sent, and T_w is the waiting time in the queue.

Example: If a client sends a request at time Ts = 10s, the last response is received at time Tf = 15s, and the waiting time in the queue is T_w = 2s, then the response time would be RT = (15 - 10) + 2 = 7s.

2) Throughput (TH): It represents the number of requests processed per unit of time. Higher throughput indicates better performance.

$$TH = \frac{\text{Total requests processed}}{\text{Time period}}$$

Example: If a server processes 1000 requests in 10 minutes, then the throughput would be TH = 1000 / (10 * 60) = 1.67 requests/s.

3) Utilization (U): It measures the extent to which a server or resource is utilized. It is often represented as a percentage. Higher utilization indicates efficient resource usage.

$$U = \frac{\text{Total busy time}}{\text{Total time}} \times 100$$

Example: If a server is busy for eight hours (28,800 seconds) out of a total of 24 hours (86,400 seconds), then the utilization would be U = (28,800 / 86,400) * 100 = 33.33%.

4) Queue length (QL): It represents the number of requests waiting in the queue for processing. Lower queue length indicates better performance.

$$QL = \lambda * W$$

Where λ is the arrival rate of requests (requests per unit of time) and W is the average waiting time in the queue.

Example: If the arrival rate is λ = 10 requests per second and the average waiting time in the queue is W = 5 seconds, then the queue length would be QL = 10 * 5 = 50 requests.

5) Server load (SL): It measures the amount of work being processed by a server or resource. It is often represented as a percentage. Lower server load indicates better performance.

$$SL = \frac{C}{T} \times 100$$

Where C is the average number of requests being processed by the server and T is the total capacity of the server (maximum number of requests it can handle in a given time period).

Example: If the average number of requests being processed by the server is C = 30 and the total capacity of the server is T = 50, then the server load would be SL = (30 / 50) * 100 = 60%.

VII. Experimental Results

Experiments were conducted using the cloud banking application developed using Python. The proposed integrated load balancing is implemented at the client and server sides. The comparative performance shows that the proposed model obtained better results. The performance of existing and proposed models was observed on 1k, 5k, and 10k transactions at a time. The existing model's round-robin (RR), Least Connection (LC), and Adaptive Load Balancing (ABL) compared with ECBA.

TABLE I. Comparative Performances in Terms of Following Metrics for 1K Transactions

Metrics	RR	LC	ALB	ECBA
Response time (Sec)	22	20	18	15.1
TH(R/S)	16.7	18.9	20.1	22.9
U (%)	70.7	72.9	75.3	80.3
QL (%)	60	55.6	52.1	49.1
SL (%)	70.8	66.7	63.1	59.3

In banking sector, daily multiple number of transactions will take place online. To process the large transactions an ECBA model obtain high results for 1k transactions given in Table I and graph shown in Fig. 2.

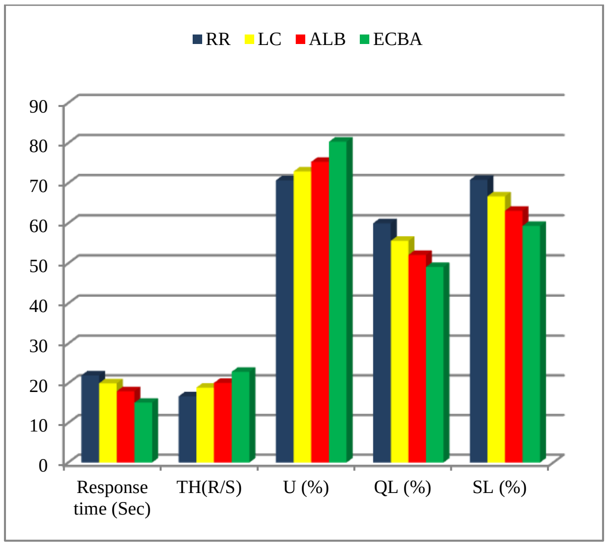

Fig. 2. Comparative performances in terms of following metrics for 1k transactions.

TABLE II. Comparative Performances in Terms of Following Metrics for 5K Transactions

Metrics	RR	LC	ALB	ECBA
Response time (Sec)	34	31.2	28	25.1
TH(R/S)	27.7	29.9	33.1	36.9
U (%)	75.7	77.4	79.2	83.3
QL (%)	69	65.6	61.1	55.1
SL (%)	80.8	66.7	63.1	59.3

Table II and Fig. 3 show the comparative performances of existing and proposed approaches based on given parameters. The overall transactions analyzed by the model are 5k. The high performance is achieved by ECBA and low performance is achieved by RR.

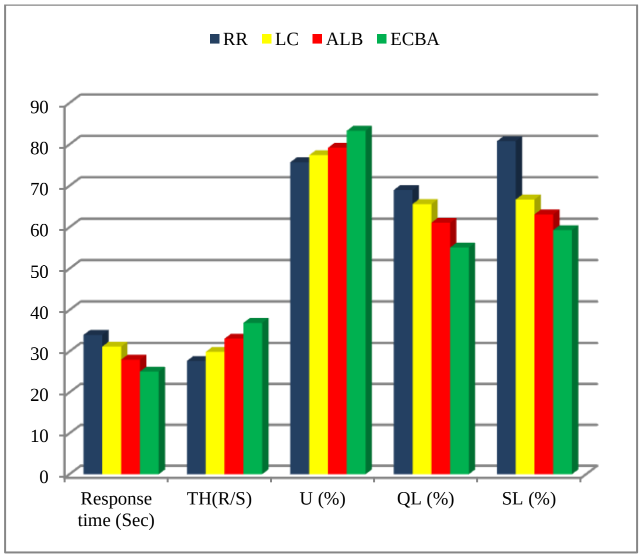

Fig. 3. Comparative performances in terms of following metrics for 5k transactions.

VIII. Conclusion

In ECBA, the WRR can balance the workload among multiple servers or instances in a cloud banking application to ensure efficient handling of customer requests. The server weights can assign memory capacity and network bandwidth based on the processing power. WRR can optimize resource utilization and prevent server overloading by allocating requests proportionally to the weights of the servers. Another technique used in cloud environments is adaptive load balancing (ALB), which dynamically adjusts the load-balancing algorithm based on real-time conditions. It continuously monitors server and network traffic performance and makes modifications to guarantee the best possible utilization of resources. ALB can assist in automatically modifying the load balancing algorithm in the context of a cloud banking application based on factors such as server response times, server availability, or network congestion. For example, suppose a server has long response times or is temporarily unavailable. In that case, ALB can route incoming requests to other available servers with lower loads, ensuring consistent performance and minimizing service disruptions. The combination of WRR and ALB can improve a cloud banking application's performance, scalability, and fault tolerance, ensuring efficient resource utilization and optimal customer experience. It enables the application to handle varying workloads while maintaining high availability, allowing dynamic adaptation to changing conditions. In the future, an ensemble load-balancing approach combined with security would be developed to achieve better performance.

REFERENCES

[1] Kumar, P.; Kumar, R. Issues and challenges of load balancing techniques in cloud computing: A survey. ACM Comput. Surv. (CSUR) 2019, 51, 1–35.

[2] Princess, G.A.P.; Radhamani, A. A Hybrid Meta-Heuristic for Optimal Load Balancing in Cloud Computing. J. Grid Comput. 2021, 19, 1–22.

[3] Elmagzoub, M.A.; Syed, D.; Shaikh, A.; Islam, N.; Alghamdi, A.; Rizwan, S. A Survey of Swarm Intelligence Based Load Balancing Techniques in Cloud Computing Environment. Electronics 2021, 10, 2718. https://doi.org/10.3390/electronics10212718.

[4] Mishra, S.K.; Sahoo, B.; Parida, P.P. Load balancing in cloud computing: A big picture. J. King Saud Univ.-Comput. Inf. Sci. 2020, 32, 149–158.

[5] Junaid, M.; Sohail, A.; Ahmed, A.; Baz, A.; Khan, I.A.; Alhakami, H. A hybrid model for load balancing in cloud using file type formatting. IEEE Access 2020, 8, 118135–118155.

[6] Shahid, M.A.; Islam, N.; Alam, M.M.; Mazliham, M.S.; Musa, S. Towards Resilient Method: An exhaustive survey of fault tolerance methods in the cloud computing environment. Comput. Sci. Rev. 2021, 40, 100398.

[7] A. Kishor, R. Niyogi, A. T. Chronopoulos and A. Y. Zomaya, "Latency and Energy-Aware Load Balancing in Cloud Data Centers: A Bargaining Game Based Approach," in IEEE Transactions on Cloud Computing, vol. 11, no. 1, pp. 927-941, 1 Jan.-March 2023, doi: 10.1109/TCC.2021.3121481.

[8] X. Wei and Y. Wang, "Popularity-Based Data Placement With Load Balancing in Edge Computing," in IEEE Transactions on Cloud Computing, vol. 11, no. 1, pp. 397-411, 1 Jan.-March 2023, doi: 10.1109/TCC.2021.3096467.

[9] T. Liu, S. Ni, X. Li, Y. Zhu, L. Kong and Y. Yang, "Deep Reinforcement Learning Based Approach for Online Service Placement and Computation Resource Allocation in Edge Computing," in IEEE Transactions on Mobile Computing, vol. 22, no. 7, pp. 3870-3881, 1 July 2023, doi: 10.1109/TMC.2022.3148254.

[10] S. Nath and J. Wu, "Deep reinforcement learning for dynamic computation offloading and resource allocation in cache-assisted mobile edge computing systems", Intell. Converged Netw., vol. 1, no. 2, pp. 181-198, 2020.

[11] J. Zhang, B. Chen, Y. Zhao, X. Cheng and F. Hu, "Data security and privacy-preserving in edge computing paradigm: Survey and open issues", IEEE Access, vol. 6, pp. 18209-18237, 2018.

[12] T. Li et al., "Privacy-preserving participant grouping for mobile social sensing over edge clouds", IEEE Trans. Netw. Sci. Eng., vol. 8, no. 2, pp. 865-880, 2021.

[13] B. Pourghebleh, A. A. Anvigh, A. R. Ramtin, and B. Mohammadi, "The importance of nature-inspired meta-heuristic algorithms for solving virtual machine consolidation problem in cloud environments," Cluster Comput., vol. 24, pp. 2673–2696, May 2021.

[14] E. H. Houssein, A. G. Gad, Y. M. Wazery, and P. N. Suganthan, "Task scheduling in cloud computing based on meta-heuristics: Review, taxonomy, open challenges, and future trends," Swarm Evol. Comput., vol. 62, Apr. 2021, Art. no. 100841.

[15] S. K. Mishra, D. Puthal, B. Sahoo, S. Sharma, Z. Xue, and A. Y. Zomaya, "Energy-efficient deployment of edge dataenters for mobile clouds in sustainable IoT," IEEE Access, vol. 6, pp. 56587–56597, 2018.

[16] B. Alankar, G. Sharma, H. Kaur, R. Valverde, and V. Chang, "Experimental setup for investigating the efficient load balancing algorithms on virtual cloud," Sensors, vol. 20, no. 24, p. 7342, Dec. 2020.

[17] A. A. Abdelltif, E. Ahmed, A. T. Fong, A. Gani, and M. Imran, "SDN-based load balancing service for cloud servers," IEEE Commun.Mag., vol. 56, no. 8, pp. 106–111, Aug. 2018.

[18] M. A. Mukwevho and T. Celik, "Toward a smart cloud: A review of fault-tolerance methods in cloud systems," IEEE Trans. Services Comput., vol. 14, no. 2, pp. 589–605, Mar. 2021.

[19] B. Cao, S. Fan, J. Zhao, S. Tian, Z. Zheng, Y. Yan, and P. Yang, "Largescale many-objective deployment optimization of edge servers," IEEE Trans. Intell. Transp. Syst., vol. 22, no. 6, pp. 3841–3849, Jun. 2021.

[20] B. Lin et al., "A time-driven data placement strategy for a scientific workflow combining edge computing and cloud computing", IEEE Trans. Ind. Informat., vol. 15, no. 7, pp. 4254-4265, Jul. 2019.

[21] S. Yang, F. Li, S. Trajanovski, X. Chen, Y. Wang and X. Fu, "Delay-aware virtual network function placement and routing in edge clouds", IEEE Trans. Mobile Comput., vol. 20, no. 2, pp. 445-459, Feb. 2021.

[22] J. Zhang, H. Guo, J. Liu and Y. Zhang, "Task offloading in vehicular edge computing networks: A load-balancing solution", IEEE Trans. Veh. Technol, vol. 69, no. 2, pp. 2092-2104, Feb. 2020.