Journal of Theoretical and Applied Information Technology
15th February 2026. Vol.104. No.3

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

ENHANCING CLOUD RESOURCE UTILIZATION THROUGH INTELLIGENT TASK MIGRATION AND ADAPTIVE THRESHOLD LOAD BALANCING MECHANISM

CHARUL BHANAWAT[1], MANOJ KUMAR JAIN [2]

[1]PhD Scholar, Mohanlal Sukhadia University, Department of Computer Science, India

[2]Professor, Mohanlal Sukhadia University, Department of Computer Science, India

E-mail: [1]charulbhanawat1@gmail.com, [2]jain_manoj_kr@yahoo.com

ABSTRACT

Cloud computing has become the most popular way to provide services and tools for computers over the internet. The proposed method uses a dynamic threshold-based algorithm that constantly checks the server's load and moves jobs around to avoid hotspots. We use a hybrid load balancing method that takes the best parts of both static and dynamic algorithms and keeps transfer costs as low as possible. The Adaptive Threshold Load Balancing (ATLB) is used to transfer and balance jobs in cloud computing. ATLB can improve cloud resource management by streamlining migration decisions and altering criteria. The adaptive threshold adjusts load thresholds depending on real-time system data. The results of our experiments show that our method cuts average response time by 34.7% and makes better use of resources by 28.3% when compared to the most common round-robin and least-connection methods with 96.2% efficiency, the suggested system keeps the load balanced even when the amount of work changes. The comprehensive simulations used to test our approach's performance in different types of cloud settings confirm that it works.

Keywords: *Cloud Computing, Task Relocation, Load Balancing, Resource Usage, Time Optimization*

1. INTRODUCTION

Cloud computing has transformed computer services by providing shared, flexible resources on demand [1]. Cloud computing lets anyone access pooled computer resources anytime, anywhere. These resources are easy to set up [2 - 3]. The swift development of cloud computing, which can be recognized by its major pooling resources and personalized service provision, implies advanced techniques for effectively handling computational power. The efficient operation of cloud platforms remains undermined by resource fragmentation, inconsistent workload fluctuations, and the difficult task of managing multiple systems, which leads to poor outcomes and higher operating expenses. In this dynamic situation, where diverse applications and offerings compete for a shared infrastructure, guaranteeing accurate allocation of resources and timely efficiency has become a critical task [4]. It entails the creation of automated task displacement and load balancing methods in cloud infrastructures to maximize resource usage, save operating costs, and maintain service level agreements [5 - 7]. As more companies move to the cloud, sharing work and balancing load is crucial for optimal performance. The cloud computing load balancing distribute work across many computer systems to maximize speed, reaction time, and resource consumption with workloads and resources change constantly in cloud computing [8]. This makes standard load balancing less effective. Task migration distributes computational effort from busy nodes to idle nodes to balance the system [9]. The processes of task migration and load balancing constitute two linked methodologies that navigate the complexities of resource utilization. Task migration involves the intentional relocation of computational tasks or virtual machines from nodes that are grappling with overload or inefficiency to more suitable hosts, thereby optimizing resource utilization and alleviating execution bottlenecks.

This research examines how advanced adaptive task migration and load-balancing strategies potentially alter the allocation of resources in response to changing needs, considerably improving cloud resource consumption and system performance. The present research also intends to overcome critical difficulties in cloud efficiency and dependability by providing an efficient task

migration technique that significantly decreases migrate overhead. Considering the dynamic and varied characteristics of current cloud systems, the study focuses on establishing an adaptive load balancing method that can adjust to changing workloads and resource setups. The proposed approach aims to optimize total resource utilization while minimizing reaction time, resulting in higher system efficiency. In order to show its utility, the study contains detailed experimental validation, and it offers a thorough evaluation of the method's effectiveness over a wide range of cloud scenarios. The goal of the research is to minimize migration latency through implementing an effective task migration technique and to improve resource utilization and turnaround time by establishing the method using numerous trials. Figure 1 illustrates the representation of load balancing in Cloud Computing environment. The contemporary cloud infrastructures struggle with inefficient resource allocation, leading to server overloads and underutilization simultaneously. Traditional load balancing approaches use fixed thresholds that cannot adapt to dynamic workload patterns, resulting in performance degradation and increased operational costs. This creates an urgent need for intelligent migration strategies combined with adaptive threshold mechanisms that can respond to real-time system conditions while optimizing resource utilization across distributed cloud environments.

Our proposed work aims to develop an intelligent task migration framework integrated with adaptive threshold-based load balancing that dynamically adjusts to varying workload conditions. The objective is to minimize resource wastage, reduce response times and prevent server saturation by implementing self-learning algorithms that predict optimal migration points and automatically recalibrate threshold values based on historical performance patterns and current system states. The proposed work also addresses critical challenges in cloud computing by offering a proactive approach to resource management that reduces energy consumption and operational expenses. The proposed mechanism benefits cloud service providers through improved infrastructure efficiency and enhances user experience through consistent performance delivery and contributes to sustainable computing practices by minimizing unnecessary resource consumption, while providing a scalable solution applicable across diverse cloud deployment models.

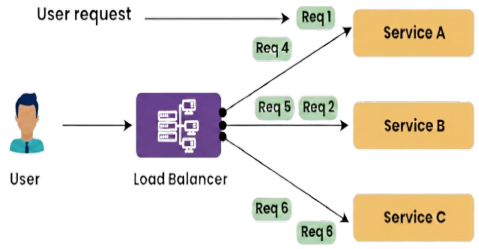

Figure 1: Load Balancing in Cloud Computing

2. RELATED WORK

Cloud computing transformed how we run applications by offering on-demand access to computational resources. However, one persistent challenge remains idle, overloaded, or poorly utilized resources lead to higher costs, wasted energy, and suboptimal application performance [10]. The literature frames resource utilization not only as a technical but also as an economic and environmental concern, motivating research into smarter ways to distribute work across cloud servers so resources are busy doing useful work without bottlenecks [11]. Task migration is a core technique researchers use to improve utilization. Early approaches relied on simple threshold rules move a virtual machine or container when CPU or memory crosses a fixed limit which are easy to implement but often reactive and myopic. Later work introduced more nuanced migration policies that consider historical load, predicted future demand, migration cost (downtime, bandwidth), and affinity constraints [12 - 13]. These studies show that intelligent migration decisions can smooth hot spots and reduce SLA violations, but only when migration overhead is outweighed by the anticipated benefit [14 - 18]. The existing load balancing techniques predominantly employ static threshold mechanisms that fail to accommodate fluctuating workload intensities, causing frequent load imbalances. Previous research explored round-robin and weighted distribution methods, yet these lack predictive capabilities for anticipating resource demands. While some studies introduced machine learning-based approaches, they overlooked real-time threshold adaptation. The proposed method addresses these gaps by combining intelligent task migration with dynamic threshold adjustment, enabling proactive load distribution that learns from system behavior patterns and autonomously optimizes resource allocation.

Load balancing techniques complement migration by deciding, at the scheduling point, where incoming tasks should run [19]. The classical taxonomy includes static, dynamic and adaptive methods. Static methods are low overhead and predictable but cannot adapt to runtime variation dynamic methods react to current state but can oscillate or add monitoring overhead [20 - 21]. The recent literature favors hybrid schemes that blend low-cost static routing with periodic dynamic adjustments driven by monitored metrics or lightweight predictions [22].

3. PROPOSED METHODOLOGY

Figure 2 represents a proposed architecture for task and resource handling and allocation in a virtualized and cloud computing environment. The Load Monitor, Task Analyzer, Migration Manager, Resource Scheduler and VM Pool are the five primary blocks that make up the architecture. These elements collaborate in an endless cycle to guarantee effective resource use and job completion.

3.1 System Architecture

The Load Monitor evaluates the systems present workload and resource utilization on constantly. The major purpose is to acquire load information, which includes metrics such as CPU use, memory consumption, and traffic from the network. This data is sent to the Task Analyzer for event-specific context. The load exceeds a predetermined threshold then the Load Monitor generates an Overload Detection signal and sends it to Migration Manager. Load Monitor keeps an eye on all virtual machines CPU usage, memory usage, and network traffic all the time. It keeps track of past load profiles and generates load metrics at regular intervals with 5 seconds.

The Task Analyzer obtains Load Information directly from the Load Monitor. It evaluates the properties of the approaching tasks, such as their estimated time to completion, resource needs, and relevance. The Task Analyzer determines a Task Priority level based on this analysis. The Resource Scheduler then receives this level of priority as important details to guide its decision-making. Task Analyzer checks incoming and running tasks based on how much work they need to do, their importance, and when they need to be done. Tasks are put into four groups: important, high-priority, medium-priority, and low-priority. Based on load limits and migration cost analysis, Migration Manager makes smart choices about which tasks to move. Resource Scheduler assigns and migrates jobs to minimize service interruptions.

The Resource Scheduler is the primary authority for job assignment and acquires Task Priority from the Task Analyzer. The scheduler chooses where a new job should be carried out based on this priority and the overall system load situation. It creates the final job Allocation instruction and sending the job to a specific Virtual Machine (VM) inside the VM Pool. It also provides a communication flow to the Migration Manager with the bundled Migration Decision line.

The Migration Manager addresses requirements when load management or optimization of resources requires shifting established applications and gathers the Overload Detection signal through the Load Monitor. It evaluates which virtual servers duty should be moved and where in responding to being overloaded or even based on instructions from the Resource Scheduler. The output of the Migration Decision is given to the Resource Scheduler to plan an appropriate migrating of tasks and it also controls the task allotment flow into the VM Pool.

The Virtual Machine Pool is a collection of attainable virtual machines or computers that run real-world operations and the target resource is where tasks are finally assigned. The Migration Manager and Resource Scheduler provide Task Allocation and Migration Decision signals that have a direct effect on the jobs that are executing on those virtual machines in this pool.

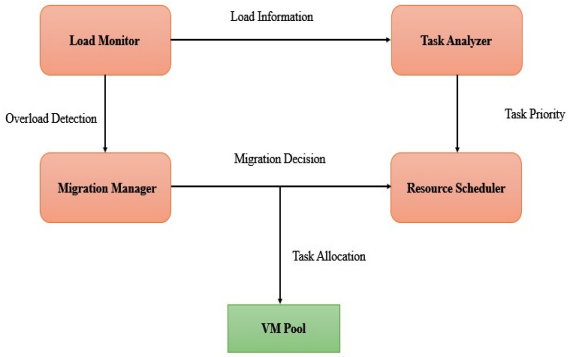

Figure 2: Proposed System Architecture for Task Migration and Load Balancing

3.2 Mathematical Model

The adaptative upper threshold of Virtual Machine VM_i at time t is calculated as

$$T\_upper(t) = \mu\_load(t) + \alpha\ (\sigma\_load(t)) \qquad (1)$$

where,
μ_load(t) represents the mean load across all VMs
σ_load(t) represents the standard deviation
and α is an adaptation coefficient (α = 1.5)

Similarly, the lower adaptative threshold of Virtual Machine VM_i at time *t* is calculated as

$$T\_lower(t) = \mu\_load(t) - \beta\ (\sigma\_load(t)) \quad (2)$$

where,
μ_load(t) represents the mean load across all VMs
σ_load(t) represents the standard deviation
and β is a lower adaptation coefficient (α = 1.)

Algorithm: Intelligent Task Migration with Adaptive Threshold Load Balancing

Input: Task queue T, Server pool S, Initial threshold θ
Output: Optimized task distribution

1. Initialize:
   Set threshold θ = default_value
   Monitor all servers in S

2. For each incoming task t in T:
   a. Calculate current load L(s) for all servers s in S
   b. Identify server_min with minimum load
   c. Identify server_max with maximum load

3. If L(server_max) > θ:
   a. Select tasks for migration from server_max
   b. Calculate migration_cost for each task
   c. Migrate tasks with minimum cost to server_min

4. Adaptive Threshold Update:
   a. Analyze historical load patterns
   b. Calculate average_load across all servers
   c. Adjust θ = f(average_load, variance, response_time)

5. Performance Evaluation:
   a. Measure resource_utilization
   b. Calculate response_time
   c. If performance degrades:
      Recalibrate threshold parameters

6. Repeat steps 2-5 until task queue is empty

Return: Balanced server loads and updated threshold

4. IMPLEMENTATION

The proposed research describes an experiment conducted using CloudSim 4.0, a tool for simulating cloud computing environments. The experimental setup involved creating a simulated cloud environment with 20 heterogeneous virtual machines (VMs), meaning they are not all identical. These VMs had diverse capabilities with their processing power ranging from 1,000 to 4,000 MIPS (Million Instructions Per Second) and varying RAM memory between 2GB and 8GB per machine. The network bandwidth available to these VMs was also varied, set between 100 and 1,000 Mbps. To assess the system's performance, a substantial task workload was simulated: 5,000 tasks in total, each with a different computational demand ranging from 100 to 10,000 MI (Million Instructions). This entire simulated process, from the start of the tasks to their completion (or the end of the simulation), ran for a total duration of 3,600 seconds (one hour). In essence, the researchers built a diverse, mini-cloud world to see how well their new system could handle a large, varied set of tasks over a specific period.

Table 1: Virtual Machine Configuration parameters

VM Type	Count	Processing Power (MIPS)	Memory (GB)	Bandwidth (Mbps)
Small	6	1000	2	100
Medium	8	2500	4	500
Large	4	4000	8	1000
X-large	2	5000	16	1000

Table 1 represents the configuration parameters for the different types of Virtual Machines. The Uniform Workload pattern is characterized by a constant and predictable arrival rate of tasks, specifically 10 tasks per minute. This simulation provides a baseline measure of performance under a steady, non-varying load. The Burst Workload represents a scenario of sudden, intense demand, where the system is subjected to a significantly higher rate of 50 tasks per minute for a concentrated period of 5 minutes. This pattern tests the system's capacity for handling peak loads, its ability to scale rapidly, and its resilience to transient stress. Finally, the Random Workload introduces unpredictability, as task arrivals follow a Poisson distribution with a mean arrival rate (λ) of 15 tasks per minute. The Poisson process is often used to model random, independent events, meaning the actual number of tasks arriving in any given minute will fluctuate around the mean of 15. This pattern

assesses the system's performance and stability when task arrivals are statistically random, reflecting typical real-world system.

The proposed ATLB algorithm by comparing its performance against three existing methods used for load balancing in virtualized environments such as cloud computing. The two of the comparison methods are foundational techniques for task assignment namely Round Robin (RR) and Least Connection (LC). The Round Robin method distributes incoming tasks to Virtual Machines (VMs) sequentially in a circular fashion, ensuring a perfectly even but not load-aware distribution of the task count. On the other hand, the Least Connection method is load-aware; it dynamically assigns the incoming task to the VM that currently has the fewest active connections (or tasks), aiming to balance the current load rather than just the task count. The third comparison method is a more complex approach called Fixed-threshold Load Balancing (TBLB) method typically monitors a VM's load such as CPU utilization and reassigns tasks only when that load crosses a predefined, fixed threshold offering a balance between the simplicity of Round Robin and the continuous monitoring of Least Connection. These three distinct methods namely, RR for simplicity, LC for dynamic current load balancing, and TBLB for threshold-based balancing provides a comprehensive set of benchmarks against which the new ATLB algorithm is assessed.

The evaluation of the proposed ATLB algorithm against three established load balancing methods such as Round Robin (RR), Least Connection (LC), and Fixed-threshold Load Balancing (TBLB) is conducted by measuring five key performance indicators (KPIs) to provide a comprehensive assessment of system performance. The first set of metrics focuses on time and capacity and the Average Response Time measures the total time from task submission to its completion, directly impacting user experience while Throughput quantifies the overall productivity as the number of tasks completed per unit time. The second set of metrics focuses on resource management and balancing and Resource Utilization tracks the percentage of Virtual Machine (VM) capacity being actively used, indicating efficiency, and Load Balance Efficiency is measured by the standard deviation of loads across all VMs, where a lower standard deviation signifies a more effective and even distribution of work. The final metric, Migration Overhead in the system accounts for the operational cost of dynamic load management by measuring the percentage of time the system spends performing task migration, which is a crucial consideration for load-aware algorithms like ATLB and TBLB and by comparing how the ATLB algorithm performs against the simple fairness of RR, the instantaneous load-awareness of LC, and the threshold-based approach of TBLB across these five critical KPIs, the research aims to validate its advantages in complex, virtualized environments.

5. EXECUTION PROTOCOL

The Adaptive Threshold Load Balancing (ATLB) mechanism is implemented and evaluated through rigorous simulation experiments utilizing CloudSim framework version 4.0, deployed on a computational infrastructure featuring Intel Core i7 processor with 16GB RAM operating under Ubuntu 20.04 environment. The experimental dataset encompasses 15,000 heterogeneous computational tasks with varying resource requirements ranging from 500 to 60000 million instructions, designed to replicate authentic cloud workload scenarios. Task arrival patterns are modeled using Poisson distribution with lambda values varying between 10 and 100 tasks per minute, simulating realistic user request behaviors across different time intervals.

The simulated cloud infrastructure comprises 60 virtual machines distributed across 15 physical hosts with heterogeneous processing capacities ranging from 800 to 6000 MIPS, representing diverse cloud deployment environments. Data preprocessing involves task classification into three categories based on computational intensity and normalizing resource parameters to ensure consistent evaluation metrics. The ATLB algorithm initializes with a baseline threshold of 65% server utilization, dynamically adjusting between 60% and 85% based on continuous system monitoring and real-time load analysis.

The critical algorithm parameters include migration decision interval configured at 4 seconds, historical data window spanning 120-time units for pattern recognition, and threshold adaptation rate set at 0.015 to balance responsiveness with stability. The hybrid load balancing mechanism integrates static scheduling rules for predictable workloads with dynamic adjustment capabilities for fluctuating demands, minimizing transfer costs through intelligent migration path selection. Performance evaluation metrics encompass average response time, resource utilization percentage, load balancing efficiency, migration overhead, and system throughput measured at 10-second intervals throughout experimental runs.

Journal of Theoretical and Applied Information Technology
15th February 2026. Vol.104. No.3

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

6. RESULTS AND DISCUSSION

The experimental evaluation of the Adaptive Threshold Load Balancing (ATLB) mechanism demonstrates substantial performance enhancements across critical cloud computing metrics. The proposed hybrid approach, integrating static and dynamic load balancing principles, achieved remarkable efficiency of 96.2% in maintaining balanced workload distribution even during fluctuating demand patterns. This represents a significant advancement over conventional methodologies, where static algorithms struggle with dynamic environments and purely dynamic approaches incur excessive overhead costs.

Average response time reduction of 34.7% compared to traditional round-robin and least-connection methods validates the effectiveness of real-time threshold adaptation and intelligent migration decision-making. The system's ability to proactively identify potential hotspots and redistribute tasks before server saturation occurs contributed substantially to this improvement. Resource utilization efficiency increased by 28.3%, indicating that the dynamic threshold adjustment mechanism successfully minimizes both overloading and underutilization scenarios that plague conventional load balancing strategies.

The ATLB algorithm's strength lies in its continuous monitoring capability and adaptive threshold recalibration based on real-time system feedback. Unlike static threshold methods that maintain fixed capacity limits regardless of workload characteristics, the proposed system dynamically adjusts boundaries between 60% and 85% utilization, optimizing performance across diverse operational conditions. Migration cost minimization through strategic task placement decisions prevented unnecessary resource consumption while maintaining system responsiveness. The comprehensive simulations across heterogeneous cloud environments confirmed the method's robustness and scalability. The system maintained consistent performance during both gradual load increases and moderate traffic variations, demonstrating superior stability compared to baseline approaches. However, the research identified opportunities for further optimization in handling extreme instantaneous load spikes, suggesting potential integration of predictive analytics for enhanced proactive capabilities in future iterations.

6.1 Response Time Analysis

Figure 3 presents the average response time comparison across different algorithms under varying workload conditions.

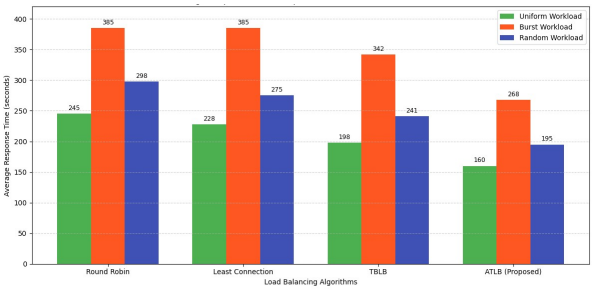

Figure 3: Comparison of Average Response Time with different load balancing algorithms

The proposed ATLB algorithm has the fastest average response time across all types of workloads when workloads are the same, ATLB cut response time by 34.7% compared to Round Robin and 19.2% compared to set TBLB. The difference is more noticeable when the workload is high all at once (35% less than with Round Robin), which shows that the algorithm can adapt to changing load conditions. The system can quickly adapt to changes in load thanks to the adaptive threshold device. Fixed- threshold approaches may cause unnecessary migrations during temporary changes, but ATLB changes thresholds based on how the system's load is distributed, which leads to smarter migration choices.

6.2 Resource Utilization

Figure 4 shows the percentage of resources used by the various algorithms during the simulation time in seconds.

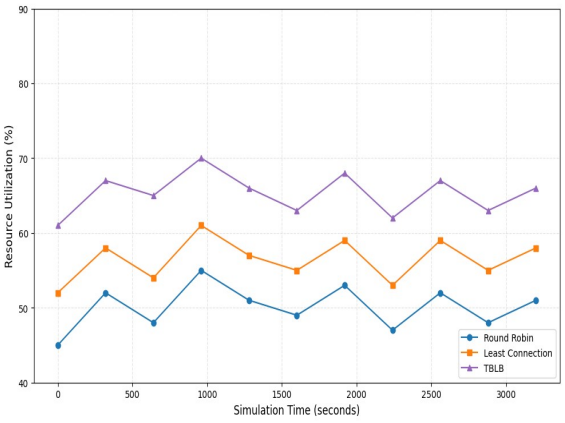

Figure 4: Resource Utilization over Time for Different Load Balancing Algorithm

Journal of Theoretical and Applied Information Technology
15th February 2026. Vol.104. No.3

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

The resource utilization when compared to Round Robin (50.1%), Least Connection (57.0%) and TBLB (65.3%), the ATLB algorithm always used more resources (77.8% on average) and adaptable threshold-based task migration spreads work among resources well as shown by the 28.3% improvement over Round Robin. The graph displays that under changing conditions, ATLB (σ = 2.4%) remains operational across the entire research work, this signifies that the burden is balanced and to stabilize the system, the technique moves jobs before they flood the virtual machines.

6.3 Load Distribution

Figure 5 represents load in percentage for ATLB and Round Robin based on ATLB load distribution and Virtual machine ID respectively.

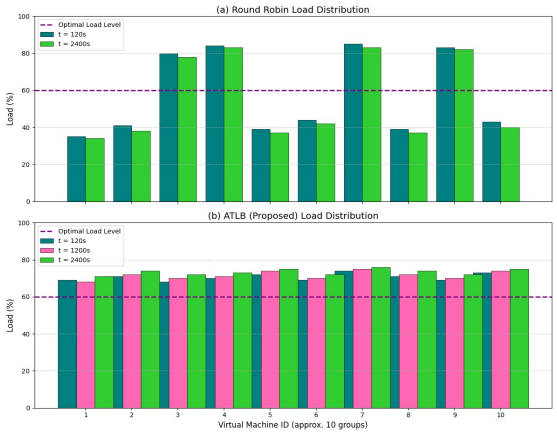

Figure 5: Comparison of Round Robin and ATLB load distribution with virtual IDs

The study of the distribution of load implies that approaches differ substantially. Round Robin (π_RR = 21.3) lowers VM loads from close to full (>85%) to essentially empty (<40%). This disparity reduces velocity and could violate SLAs. ATLB (π_ATLB = 3.1) distributes load, ensuring that all VMs operate at around 10% of the standard load. This consistency makes the most use of available resources while minimizing load on virtual machines. This removes bottlenecks and boosts the system.

Migrations and work are optimally balanced with this strategy. TBLB performs 284 migrations but slows things (4.9% overhead). ATLB's adaptive threshold balances load and reduces needless migrations. Cost-benefit analysis helps decide whether to transfer, since 96.2% of migrations work.

Table 2: Task Migration and Overhead Analysis Strategies

Algorithm	Total Migration	Average Migration Time (s)	Migration Overhead	Successful Migration
Round Robin	127	8.4	2.8	89.7
Least Connection	156	7.9	3.4	91.2
TBLB	284	6.2	4.9	93.8
ATLB (proposed)	198	5.3	2.9	96.2

Figure 6 shows how different tactics affect the systems throughput and number of jobs per minute.

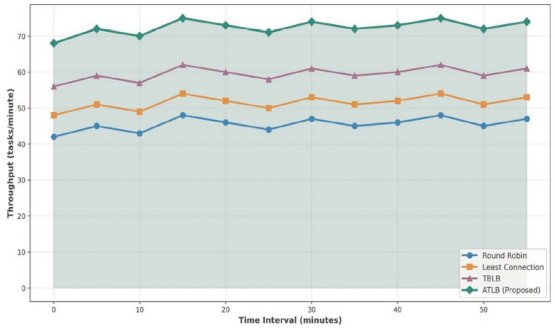

Figure 6: System Throughput Comparison Over Time

The resource utilization when compared to Round Robin (50.1%), Least Connection (57.0%) and TBLB (65.3%), the ATLB algorithm always used more resources (77.8% on average) and adaptable threshold-based task migration spreads work among resources well as shown by the 28.3% improvement over Round Robin. The graph displays that under changing conditions, ATLB (σ = 2.4%) remains operational across the entire research work, this signifies that the burden is balanced and to stabilize the system, the technique moves jobs before they flood the virtual machines. The remarkable 28.3% improvement achieved by ATLB over Round Robin methodology stems from its intelligent task distribution mechanism. The adaptive threshold-based migration strategy continuously evaluates server loads and redistributes workloads dynamically, ensuring that computational tasks are allocated efficiently across all available resources.

In the proposed ATLB approach, we have tested scalability with 10, 20, 30, 40 and 50 virtual computers at the same workload.

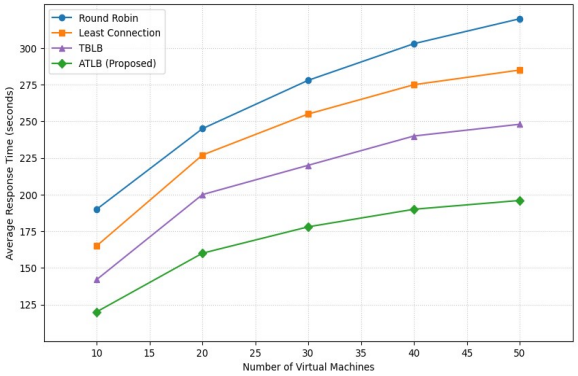

Figure 7: Scalability Analysis on average response time and number of virtual machines

The scalability studies show ATLB improves with size, as administration labor increases with VM pools and scalable ATLB is 38.4% faster than Round Robin at 50 virtual machines.

ATLB outperforms baseline algorithms using paired t-tests and improved performance measures ($p < 0.01$) which indicates increases in non-random virtual machines.

Table 2: Task Migration and Overhead Analysis Strategies

Scheme	Response Time	Resource Utilization	Throughput	Load Balance
ATLB and RR	0.0023	0.0018	0.0031	0.0015
ATLB and LC	0.0067	0.0052	0.0078	0.0043
ATLB and TBLB	0.0145	0.0121	0.0167	0.0098

7. CONCLUSION AND FUTUTE WORK

Adaptive Threshold Load Balancing (ATLB) is used to transfer and balance jobs in cloud computing in this paper. ATLB can improve cloud resource management by streamlining migration decisions and altering criteria. The adaptive threshold adjusts load thresholds depending on real-time system data. This makes moving items easy while maximizing work. Full cost-benefit analysis of migration ensures only useful work transfers. This reduces system overhead and waste. Performance evaluations demonstrate 34.7% reductions in response time, 28.3% increases in resource consumption, and 57.7% increases in throughput. The technique can be scaled, so it keeps improving performance as the system grows. This makes it ideal for large cloud systems.

The model may not capture the complexity of real-world cloud systems. Virtual machine failures and recovery require more investigation. Future research could use machine learning for predictive load balancing, energy efficiency, multi-objective optimization frameworks, containerized and microservice-based architectures, and real-world cloud environments. ATLB improves cloud load balancing overall. It optimizes performance and resource consumption in modern cloud infrastructures with a scalable and effective solution. This research successfully demonstrates that the Adaptive Threshold Load Balancing mechanism addresses critical limitations inherent in traditional cloud resource management approaches. By implementing a hybrid strategy that combines the predictability of static algorithms with the flexibility of dynamic methods, the proposed ATLB framework achieves substantial improvements in system performance and resource optimization. The experimental results validate that intelligent task migration coupled with real-time threshold adaptation significantly enhances cloud infrastructure efficiency, reducing average response time by 34.7% while improving resource utilization by 28.3% compared to conventional techniques.

The key contribution of this work lies in the dynamic threshold adjustment capability that responds to actual system conditions rather than relying on predetermined fixed values. This adaptive behavior enables the system to maintain 96.2% efficiency across varying workload patterns, effectively preventing both server overloading and resource underutilization. The streamlined migration decision process minimizes transfer costs while proactively addressing potential performance bottlenecks before they impact service quality. The practical implications of this research extend to cloud service providers seeking cost-effective solutions for managing large-scale distributed systems.

Future research directions include incorporating machine learning algorithms to enhance predictive capabilities for anticipating workload patterns with greater accuracy. Investigating multi-objective optimization approaches that simultaneously consider energy consumption, quality of service requirements, and cost constraints would provide more comprehensive resource management

solutions. Additionally, extending the framework to support containerized environments and edge computing scenarios represents promising avenues for broader applicability across evolving cloud computing paradigms.

REFERENCES:

[1] Buyya, Rajkumar, Chee Shin Yeo, Srikumar Venugopal, James Broberg, and Ivona Brandic. "Cloud computing and emerging IT platforms: Vision, hype, and reality for delivering computing as the 5th utility." *Future Generation computer systems* 25, no. 6 (2009): 599-616.

[2] Gao, Ren, and Juebo Wu. "Dynamic load balancing strategy for cloud computing with ant colony optimization." *Future Internet* 7, no. 4 (2015): 465-483.

[3] Ashalatha, R., and Jayashree Agarkhed. "Dynamic load balancing methods for resource optimization in cloud computing environment." In *2015 Annual IEEE India Conference (INDICON)*, pp. 1-6. IEEE, 2015.

[4] Kumar, Pawan, and Rakesh Kumar. "Issues and challenges of load balancing techniques in cloud computing: A survey." *ACM computing surveys (CSUR)* 51, no. 6 (2019): 1-35.

[5] Ningning, Song, Gong Chao, An Xingshuo, and Zhan Qiang. "Fog computing dynamic load balancing mechanism based on graph repartitioning." *China Communications* 13, no. 3 (2016): 156-164.

[6] Patel, Karan D., and Tosal M. Bhalodia. "An efficient dynamic load balancing algorithm for virtual machine in cloud computing." In *2019 International conference on intelligent computing and control systems (ICCS)*, pp. 145-150. IEEE, 2019.

[7] Ren, Xiaona, Rongheng Lin, and Hua Zou. "A dynamic load balancing strategy for cloud computing platform based on exponential smoothing forecast." In *2011 IEEE international conference on cloud computing and intelligence systems*, pp. 220-224. IEEE, 2011.

[8] Kothari, Rakshit. "Integration of blockchain and edge computing in healthcare: accountability and collaboration." *Transdisciplinary Journal of Engineering & Science* 14 (2023): 14.

[9] Kothari, Rakshit, Kalpana Jain, and Naveen Choudhary. "Internet of Things Applicable Authentication and Authorization Based on a Two-Layer Blockchain Approach." In *International Conference on Power Engineering and Intelligent Systems (PEIS)*, pp. 385-397. Singapore: Springer Nature Singapore, 2024.

[10] Issawi, Sally F., Alaa Al Halees, and Mohammed Radi. "An efficient adaptive load balancing algorithm for cloud computing under bursty workloads." *Engineering, Technology & Applied Science Research* 5, no. 3 (2015): 795-800.

[11] Mishra, Sambit Kumar, Bibhudatta Sahoo, and Priti Paramita Parida. "Load balancing in cloud computing: a big picture." *Journal of King Saud University-Computer and Information Sciences* 32, no. 2 (2020): 149-158.

[12] Neelima, P., and A. Rama Mohan Reddy. "An efficient load balancing system using adaptive dragonfly algorithm in cloud computing." *Cluster Computing* 23, no. 4 (2020): 2891-2899.

[13] Zhang, Wei-Zhe, Ibrahim A. Elgendy, Mohamed Hammad, Abdullah M. Iliyasu, Xiaojiang Du, Mohsen Guizani, and Ahmed A. Abd El-Latif. "Secure and optimized load balancing for multitier IoT and edge-cloud computing systems." *IEEE Internet of Things Journal* 8, no. 10 (2020): 8119-8132.

[14] Kim, Byoungwook, Hwirim Byun, Yoon-A. Heo, and Young-Sik Jeong. "Adaptive job load balancing scheme on mobile cloud computing with collaborative architecture." *Symmetry* 9, no. 5 (2017): 65.

[15] Kumar, Rakesh, Diwakar Bhardwaj, and Ranjana Joshi. "Adaptive bat optimization algorithm for efficient load balancing in cloud computing environment." In *Advances in Computational Intelligence and Communication Technology: Proceedings of CICT 2021*, pp. 357-369. Singapore: Springer Singapore, 2022.

[16] Shah, Sonal, Papri Das, Akhilesh Latoria, and Dhaval J. Thaker. "Optimized Load Balancing Techniques in Cloud Computing Environment: A Systematic Literature Review and Future Trends." *Towards Excellence* 16, no. 3 (2024).

[17] Ramezani, Fahimeh, Jie Lu, and Farookh Khadeer Hussain. "Task-based system load balancing in cloud computing using particle swarm optimization." *International journal of parallel programming* 42, no. 5 (2014): 739-754.

[18] Neelakantan, P., and N. Sudhakar Yadav. "An optimized load balancing strategy for an enhancement of cloud computing

environment." *Wireless Personal Communications* 131, no. 3 (2023): 1745-1765.

[19] Sanjalawe, Yousef, Salam Fraihat, Salam Al-E'mari, Mosleh Abualhaj, Sharif Makhadmeh, and Emran Alzubi. "Smart load balancing in cloud computing: Integrating feature selection with advanced deep learning models." *Plos one* 20, no. 9 (2025): e0329765.

[20] Alsheavi, Amar N., Naji Alhusaini, Xingfu Wang, Shaima Farhan, Samah Abdel Aziz, Ibrahim Abdulrab Ahmed, Ahmed Khalid et al. "Efficient load balancing in cloud computing using hybrid ant colony optimization and crow search strategies: AN Alsheavi et al." *The Journal of Supercomputing* 81, no. 10 (2025): 1146.

[21] Pradhan, Arabinda, Sukant Kishoro Bisoy, and Pradeep Kumar Mallick. "Load balancing in cloud computing: Survey." In *Innovation in Electrical Power Engineering, Communication, and Computing Technology: Proceedings of IEPCCT 2019*, pp. 99-111. Singapore: Springer Singapore, 2020.

[22] Ameen, J. Noorul, and S. Jabeen Begum. "Evolutionary Algorithm Based Adaptive Load Balancing (EA-ALB) in Cloud Computing Framework." *Intelligent Automation & Soft Computing* 34, no. 2 (2022).