International Journal for Modern Trends in Science and Technology
Volume 10, Issue 12, pages 122-131.
ISSN: 2455-3778 online
Available online at: http://www.ijmtst.com/vol10issue12.html
DOI: https://doi.org/10.46501/ijmtst.v10.i12.pp122-131

A Two-Layered Ensemble Framework for Secure and Resilient Parallel Cloud Load Balancing

V.Anantha Lakshmi[1] | Dr R. Satya Prasad[2]

[1]Research Scholar, Acharya Nagarjuna University, Guntur, AP, India.
[2]Professor and Dean R & D, Department of Computer Science & Engineering, Dhanekula Institute of Engineering & Technology, Ganguru, Vijayawada, A.P., India.

To Cite this Article
V.Anantha Lakshmi and Dr R. Satya Prasad, "A Two-Layered Ensemble Framework for Secure and Resilient Parallel Cloud Load Balancing", International Journal for Modern Trends in Science and Technology, 2024, 10(12), pages. 122-131. https://doi.org/10.46501/ijmtst.v10.i12.pp122-131

Article Info
Received: 14 December 2024; Accepted: 28 December 2024.; Published: 03 January 2025

ABSTRACT

Load balancing is a sub-domain of cloud computing that distributes workloads across multiple servers and virtual machines (VMs) and is cost-effective. The load balancer primarily determines whether a server can handle each request on the cloud platform. Nowadays, load balancers help prevent high traffic, manage routing, and handle faults in data transmission, resulting in high performance. The existing load-balancing models face several issues, such as latency, autoscaling, bottlenecks, and Cold starts. In this paper, a two-layered ensemble load-balancing (TLELB) framework is introduced to address various issues in cloud platforms. In TLELB, the first layer is also a fast layer that executes heterogeneous schedulers in an ensemble to generate low latency and assign high-quality tasks. The second layer, named BottleneckBreachQuell, focuses on solving issues such as bottlenecks and Cold starts. Finally, the Secure and Resilient Parallel algorithm monitors transmission data and splits it into parts for reliable processing. The researchers conducted experiments on two different sub-datasets, such as the Google Cluster Workload Traces 2019 dataset. Results showed that the proposed approach achieves high performance in Response Time (RT), Execution Time (ET), Throughput (TP), Task Migration Rate (TMR), and CPU Utilization (CU).

KEYWORDS: Cloud Computing, Virtual Machines (VMS), Latency, Autoscaling, Bottlenecks, Two-Layered Ensemble Load-Balancing (TLELB).

1.INTRODUCTION

Cloud computing is a paradigm shift concerning the way in which computing resources are provision and consumed. It offers ubiquitous, convenient access through network models for rapid provisioning of a shared pool of configurable resources (e.g., networks,servers storage applications) that can be rapidly utilized with minimal management effort or service provider interaction. With organizations moving more of their workloads to the cloud, provisioning and

managing those resources effectively has become a key consideration for delivering the performance, cost and user experience customers expect. One of its enabler to achieve such high efficiency is through load balancing. The process of distributing the incoming requests, which consists of computational workloads among available VMs to ensure that no single resource is overwhelmed, is known as cloud load balancing. It avoids any idle computing node whilst several are overloaded, thus enhancing the throughput and decreasing the response time, meanwhile system stability is guaranteed. A good load distribution policy that also leads to the fault tolerance, and scalability and overall reliability of a cloud. There are conventional load balancing methods like Round Robin (RR), Weighted Least Connection (WLC) as well as Randomized wherein these algorithms support the simple strategies for distributing and sharing of computation among multiple resources but fail to adjust themselves with dynamism aspects of emerging cloud environment. Intelligent and adaptive load balancing techniques are introduced in view of explosive increase of heterogeneous resources, variability of work loads as well as complexity requirements from user. These models come in a variety of forms, as rule-of-thumb heuristics, metaheuristic or machine learning based algorithms that can assign resources dynamically live based on the system behavior and performance characteristics.

Efficient utilization of resources, minimal latency and uninterrupted service availability are crucial in large scale cloud computing era. Load balancing is a critical part of distributing the work between several virtual machines or servers in order to prevent any server from being overwhelmed and help achieve optimal performance and stability. However, classical load balancing techniques have been limited in two principal aspects - dealing with security attacks and contending under a changing workload. To mitigate these challenges, we propose Secure and Resilient Parallel Cloud Load Balancing (SRPCLB) which combines modern parallel processing techniques with strong security capabilities. This method facilitates the sharing of computational task assignment taking into consideration not only system performance parameters but also with the knowledge of possible malicious behavior, node failure or network attack. Have the same balancing decision made simultaneously on distributed nodes, such that SRPCLB can achieve scalability, fault-tolerance and fast-recovery speed yet reduce latency. And the introduction of blockchain-inspired trust management, machine learning-based anomaly detection and reinforcement learning-based task scheduling enhance reliability and adaptability of the system. With the ever-increasing size, and growing complexity of cloud infrastructures, secure parallel load balancing plays a key enabling role in sustainable high performance trustworthy cloud operations.

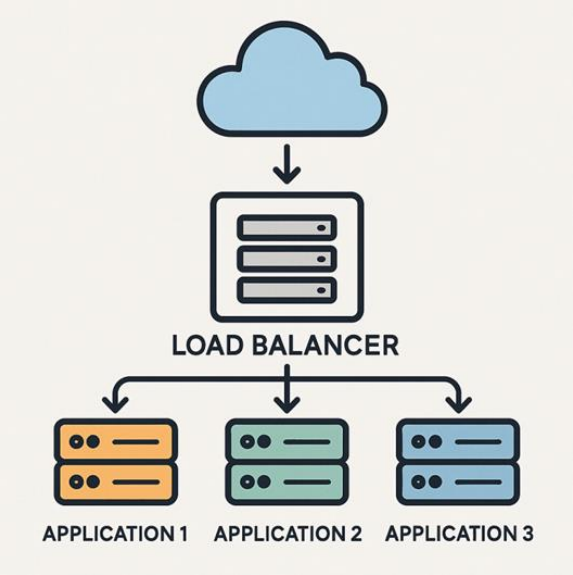

Figure 1: Basic Load balancing in Cloud Platforms

2. LITERATURE SURVEY

Dornala et al. [9] presents an Advanced Multi-Model Cloud Service Framework (AmmcSF) that combines different load balancing algorithms, heuristic/metaheuristic as well as AI driven ones, for scalable and fault tolerant services with low energy consumption in distributed cloud systems. Based on runtime workload and the service-level metrics, the framework makes a dynamic decision in order to choose the most suitable load balancing. Experimental results with benchmark cloud datasets and CloudSim simulation show that our new model improves drastically the average response time, throughput, and resource usage when compared with conventional single-model multi-systems. The combination of different algorithms needs more computational power during decision making. Ponnapalli et al. [10] introduces Triple-Tap Hybrid Load Balancing (TTHLB) System, which is tailored to health surveillance systems. The TTHLB combines the static, dynamic and AI based predictive balancing policies to control fluctuating loads

from PMDs. First, a static analyser distributing base-level loads through the servers; then, a dynamic scheduler that adapts to changes in data traffic; finally, a reinforcement learning-like module that forecasts and reallocates future workloads before saturation decreases without need of stopping for adding additional capacity. Experimental results based on simulated medical workloads show that TTHLB can achieve better performance than conventional algorithms in response time, throughput, energy consumption and fault tolerance to guarantee the reliable real-time health data transmitting and analyzing. However, the model's efficacy crucially depends upon accurate workload forecast and may deteriorate under conditions of heavy patient data spikes or emergency.

Dornala et al. [11] actual system uses coordination over quantum superposition and entanglement to have optimal resource utilization with real-time the task allocation, parallel decision making, and minimal latency. By using QECC and fault-tolerant quantum gates, the system is designed to be resilient against computational and hardware errors. Besides, the QFTLB model employs quantum-inspired reinforcement learning for dynamic scheduling, which is beneficial to the adaptive resource allocation under different workloads. Building and operating a quantum-capable cloud infrastructure is resource-consuming and expensive, which makes it inaccessible for most users. Sudhir Ponnapalli et al. [12] proposed a lightweight virtualization-based framework with data compression and content delivery optimization works for reducing the resource overhead and increasing the performance. In this context, the microservices-oriented design and smart cache engines, the platform is able to perform more efficient data fetching with better latency management and less energy cost. Besides, adaptive security policies for encryption and access control are designed to guarantee secure and efficient data transactions against multi-tenancy cloud infrastructures. It might not be optimized enough to deal with really, really big data sets or large physics streams of data in real time analysis.

Shukri et al. [13] presents the Enhanced Multi-Verse Optimizer (EMVO) to conduct an efficient and adaptive task scheduling in heterogeneous cloud infrastructures. The proposed EMVO extends from the classical Multi-Verse Optimizer (MVO) algorithm that utilizes dynamic adaptive weights combined with chaotic initialization and a hybrid mutation operator to ensure exploitation exploration trade-off. The algorithm adopts dynamic weighting of scheduling priorities according to realtime load variations and the capabilities of virtual machines (VMs). Articulating adaptive control parameters and hybrid mutation operation results in more computationally intensive algorithms than simple MVO or heuristic methods. Arunarani et al. [14] thoroughly reviews the literature of the stateof-the-art task scheduling algorithms in cloud computing including heuristic, metaheuristic hybrid, and machine learning methods. In the survey, the scheduling algorithms are classified by their goals; i.e., minimizing makespan, load balancing, and reducing energy consumption and ensuring Quality of Service (QoS). It also discusses recent progress in intelligent and adaptive scheduling methods based on deep learning, reinforcement learning, and bio-inspired optimization algorithms. A number of the scheduling algorithms surveyed are evaluated in only static or semi-static scenarios that do not capture the dynamic nature of real cloud workloads.

Rjoub et al. [15], they presented the BigTrustScheduling, a trust-aware big data task scheduling strategy focusing on improving reliability, security and performance like stated trust in cloud computing. In the model, trust evaluation mechanism and task scheduling algorithm are combined in order to guarantee that tasks of computing will be allocated to the most reliable nodes with honesty. Trust is measured by attributes such as historical performance, security compliance, service availability and user rate. By taking trust into account into the scheduling decision, BigTrustScheduling decreases task failures, optimizes resource utilization, andamp; increases QoS (Quality of Service) for large-scale data processing. The ongoing trust assessment increases computational and communication workload, challenging the scalability of the system. Kavitha et al. [16] and some related work on security threats, data protection and scheduling algorithms in cloud computing is reviewed. Security mechanisms are classified into authentication, encryption, and intrusion detection approaches, and trust management whereas scheduling schemes are divided into resource-aware, energy-efficient, deadline-based, and hybrid optimization techniques.

The survey emphasizes the impact of machine learning, block chain and heuristics based algorithms on the improvement in secure task assignment to resources and optimized resource. It also points out some cutting-edge research directions including AI-oriented secure scheduling, quantum-motivated optimization and lightweight encryption for edge-cloud fusion. Research indicates that such a hybrid AI–metaheuristic scheduling models decrease the task completions times up to 20–35% as compared to conventional ones.

3. PROPOSED METHODOLOGY

Two-layer ensemble load balancing (TLELB) is a two-level system intended to improve the Quality of Service (QoS) performance and reduce cold start, high latency, and bottleneck issues for cloud environments. In the first layer also called the Fast Layer, several diverse types of scheduling algorithm including RL-based schedulers- are presented in parallel as an ensemble. Such ensemble strategy helps in making decisions on-the-fly, and the task allocation can be distributed to different schedulers for better performance. Weighted voting mechanism or adaptive fusion is used by the ensemble to find optimized task-to-node assignment that minimizes the latency, maximizes throughput and balances resource use between cloud nodes.

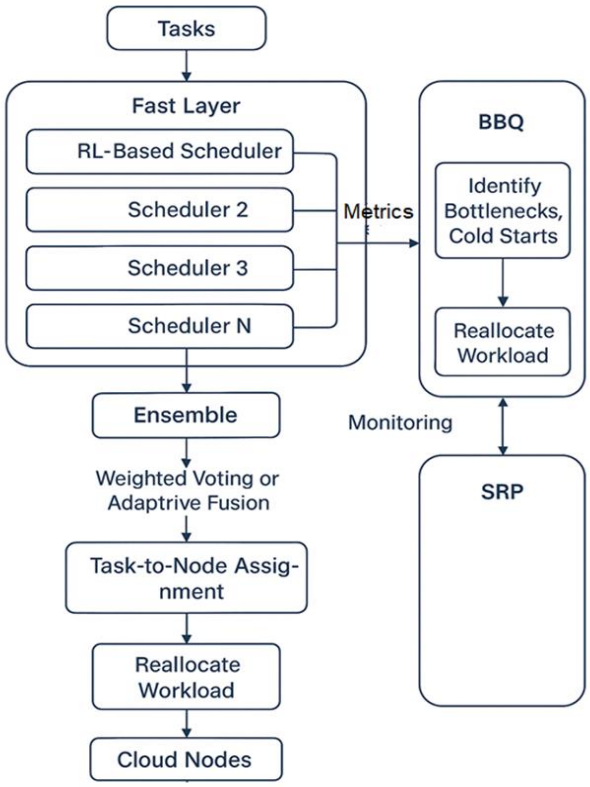

Figure 2: System Architecture for TLELB

The next layer operates deeper inefficiencies of the more politically implemented by GPS scheduling. This layer analyzes system metrics (such as task wait time, CPU/memory usage and depth of the queue) to identify bottlenecks or cold start. Once the possible overload or under load is detected, BBQ can dynamically reallocate the workload by approximating resources performance through a time-series analysis or regression models. It uses adaptive scaling and migration policies to redistribute the workload, still provide good performance in case of quick surges or outages. Thereby it ensures that the system remains resilient and stable over time.

The only exception is SRP (i.e., Secure and Resilient Parallel) that works as a transmission monitoring tool implemented over two layers. It robustly splits transmission data into multiple fragments, which can be treated in parallel to achieve throughput as well as fault-tolerance and confidentiality properties. All the fragments are hash-verified prior to aggregation, which may help the data transmission against tampering and losing. By combining SRP with the 2-layer architecture, TLELB improves scheduling accuracy and fault-tolerance, an possesses secure, scalable and high-quality task execution features for modern heterogeneous cloud systems.

3.1 Reinforcement Learning (RL)-based schedulers

RL-based schedulers in the field of cloud load balancing learn optimal schedule allocation by interacting with the cloud environment at whilst reinforcement learning (RL). Different from typical schedulers that rely on predefined rules or heuristics, RL-based schedulers formulate the scheduling decisions as a Markov Decision Process (MDP) defined by the tuple (S,A,P,R,γ), where S are system states (e.g., task queue length, CPU load and bandwidth usage), A is actions in scheduling transition, P is the state transition probability R interpret as reward of making an action, and γ is discount factor of rewards at different time periods. The schedulers take a current state s_t as input, select an action a_t and get a resultant reward r_t indicating the QoS impact (e.g., latency degradation or utilization equilibrium). Where G_tan infinite horizon is expected cumulative reward with:

$$G_t = \mathbb{E}\left[\sum_{k=0}^{\infty} \gamma^k r_{t+k+1}\right]$$

The RL scheduler learns the optimal allocation strategy by iteratively updating a policy π (a | s) which specifies the probability of taking action a given state s.

In Deep Q-Networks (DQN) are frequently used to approximate the expected value of state-action pairs. In Q-learning, the action-value function Q(s, a) is updated as per Bellman equation:

$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha \left[r_t + \gamma \max_{a'} Q(s_{t+1}, a') - Q(s_t, a_t) \right]$$

Where α is the learning rate that modulates the impact of new knowledge. In today's RL-based schedulers, Q(s,a;θ) is approximated using deep neural networks - a representation that enables the scheduler to generalize over large, continuous state spaces and constantly evolving workloads. The reward function should be constructed to promote low response time, well-balanced consumption of CPU and least migration cost; the reward function may include but not limited to:

$$r_t = -(\omega_1 \times \text{Latency} + \omega_2 \times \text{Imbalance} + \omega_3 \times \text{Energy})$$

Where are tunable weights, by refining its policy, the RL-based scheduler can automatically adjust to diverse load patterns and achieves near-optimal scheduling efficiency under the Two-Layered Ensemble Load Balancing (TLELB) scheme.

3.2 Weighted Voting

Weighted voting is an advanced weight-based decision-making algorithm for multi-scheduler systems, such as the TLELB model. As opposed to historical majority decision voting, where all schedulers have the same weight, in weighted voting, each participating scheduler is given some precedence, or "weight," depending on its past success rate, response time, etc., or its fitness for a particular workload pattern. A separate task-to-node recommendation is provided by each scheduler in the ensemble, and their recommendations are weighted and summed. The ultimate decision is made by selecting the scheduler that achieves the maximum cumulative weighted score, and stronger or more stable algorithms will have a greater impact on the final scheduling. The approach enables the ensemble to leverage the benefits of fast heuristics, predictive metaheuristics, and adaptive learning algorithms for optimal load regulation in dynamic clouds. When applied to cloud load balancing, weighted voting enhances robustness and flexibility in scheduling decisions by accounting for the disparate performance levels of diverse schedulers. For example, a Reinforcement Learning-based scheduler may benefit from a higher weight in dynamic workloads. Still, in stable situations, heuristic schedulers, such as Round Robin or Min-Min, can be favored for their predictability. The voting-with-weights model can be formulated as follows.

$$D = \arg\max_j \sum_{i=1}^{n} w_i \times v_{ij}$$

Where D is the ultimate decision (e.g., task-to-node mapping) selected, wi represents the weight of the *i*th scheduler, and vij is the vote (or confidence score) assigned to decision j by scheduler *i*. Through adaptively adjusting these weights according to performance feedback metrics like delay, energy consumption and throughput, weighted voting guarantees the online self-optimization. It is therefore the key fusion engine in Fast Layer of TLELB, which supports low latency, high throughput and balanced utilization of resource among the nodes in cloud.

3.3 The BottleneckBreachQuell (BBQ) Layer

The BottleneckBreachQuell (BBQ) consider as the refined layer in TLELB framework. The Fast Layer is concerned with quickly dispatching work onto multiple heterogeneous (or reinforcement learning based) schedulers, while BBQ works after this phase to constantly monitor, predict and fix imbalance in the system. Its main objective is to avoid performance drop by overloading nodes, idle nodes, and traffic surges. The layer does so by monitoring system metrics that reflect resource usage, task queue length, response time and throughput. It triggers the dynamic re-allocation of tasks to keep the static load on balance when it observes abnormal Constitution Load or performance degraded. This prevents any single machine from being a throughput bottleneck, and keeps idle machines well-integrated into the systemís workload. BBQ is shown to substantially enhancing a scaling, stability and QoS in complex clouds by keeping equal load distribution.

In this context, the BBQ layer provides ability to predict and adapt over time, as well as self-healing properties for long-term performance sustainability. It uses feedback loops driven by learning that anticipate

bottlenecks considering they are about to happen so resources can be adjusted proactively. You then use the target tasks from each domain to import some prior information into BBQ, lowering how much work it has to do since you now have some idea of how those policy gradients should look. This is data that helps BBQ predict which potential system states are coming in fast-more than once giving it predictive intelligence rather than just a bunch of dials and alarm bells. The system activities have been learned so well by BBQ in anticipation that it uses them a heads-up about future situations-and often taking preemptive action as opposed to letting an object crash and then having to react. BBQ also helps resolve the cold start, when a virtual machine that is just brought up or left idle, starts to run slowly initially—by softening nodewarmup according to historical performance and readiness of nodes. This controlled activation results in the absence of latency spikes, and permits to easily add new resources into the load-balancing environment. In conclusion, the BBQ layer turns the TLELB framework into a self-regulatory, well-behaved and latency-conscious system that can respond to real-time variations in load without sacrificing stability even under high or erratic workloads.

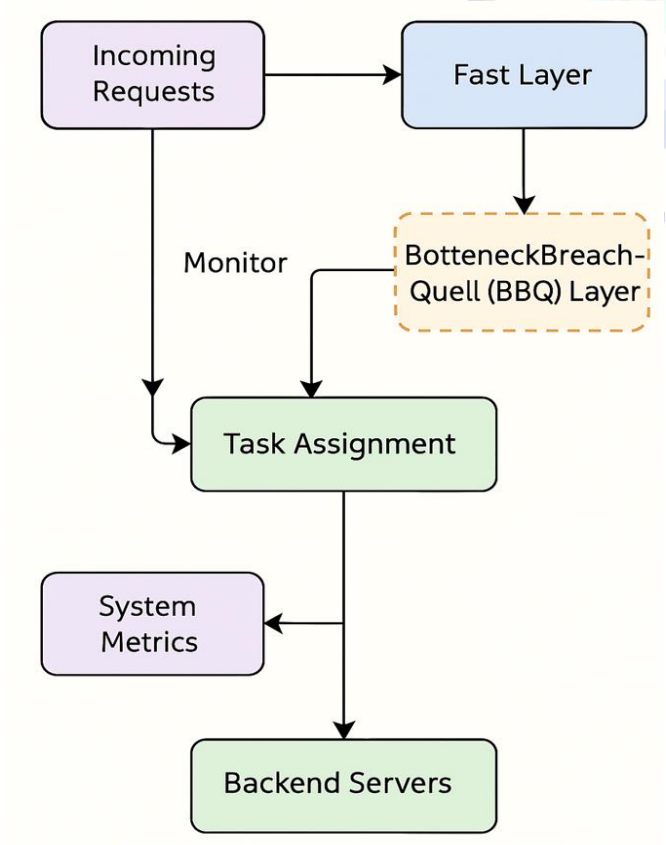

Figure 3: Architecture Diagram for BBQ

3.4 Secure and Resilient Parallel (SRP) Algorithm

The secure and resilient parallel (SRP) algorithm is an increasing layer of cloud system architecture in order to improve the quality-of-service, security, and fault tolerance in data transmission for distributed processing. This method first collects and examines real-time transmission data to ascertain any unusual phenomena of the network, such as congestion, packet loss or node failure. After scanning the data stream, SRP divides the input into multiple fragments or blocks that can be processed separately and possibly in parallel across several nodes. Each fragment is self-integrity-protected (by means of a hash, a MAC, or a digital signature) in order to avoid modification and allow its verification at the time of reconstitution. This fragmentation enables an efficient employment of several nodes, it also introduces fault isolation – if a node crashes or a transmission link breaks and goes dead only part of the data must be sent again. In addition, SRP embeds a control feedback loop to monitor transmission latency and packet drop rate, as well as system throughput when adapting chunk size and redundancy level dynamically. The key objective is secure fast and reliable processing of data even for failures or when there exist malicious nodes.

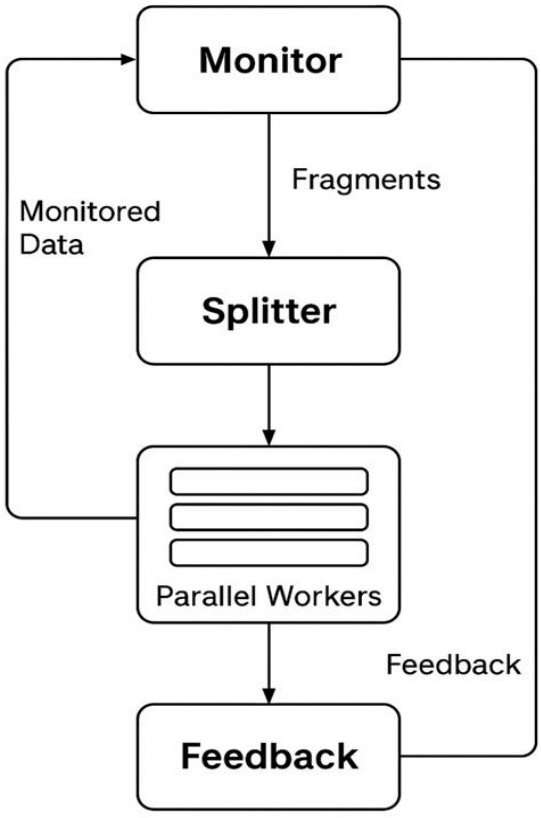

Figure 4: Architecture Diagram for SRP

Mathematically, data reliability of the SRP algorithm can be provided as the probability that all k original fragments can be recovered after transmission. Let each fragment be replicated m times and have a probability of failure p, then the probability of successful recovery overall can be expressed as:

$$P_{success} = (1 - p^m)^k$$

The (1- p^m) probability of success for each individual fragment (at least one copy must survive), and we exponentiate to k because all fragments must be recoverable to reconstruct the message. This equation shows that redundancy (m) improves reliability exponentially, and the number of fragments (k) determines how much nodes share the load. The SRP latency perspective considers the minimum total response time and fault tolerance. For parallel execution of the snippets, the overall query latency is dominated by the time required for the slowest necessary snippet to finish its computation. If the response time at every worker node is with mean μ_w and variance σ_w^2 we have: The desired guaranteed completion threshold that we expect for reliable can be quantified as follows.

$$T_{out} = \mu_w + k_\sigma \sigma_w + \Delta$$

Where is a safety margin factor (typically between 2 and 3) and Δ provides the ability to cope with network jitter and processing delay. This dynamic timeout value allows SRP to recover from mildly delayed networks without retransmitting too early and at the same time try to cater for slow or non-responsive nodes. Using this timing feedback, SRP adjusts the tradeoff between latency and reliability on-the-fly to ensure secure and efficient distributed processing in cloud load-balancing scenarios.

4. DATASET DESCRIPTION

The ClusterData2019/v3 (download) dataset consists of anonymized trace data for production workloads of Google's Borg cluster management system, at a large scale. These datasets collected from detailed records about submitted jobs, extracted tasks running these jobs as well as resource demands from several data center clusters over 30 days. In the former, we focused on two concepts, Job Events and Task Events, as two fundamental building blocks for studying workload dynamics and scheduling efficiency. The Job Events data contains job-level events like submission, modification and completion; whereas the Task Events data contains lifecycles of individual tasks such as scheduling, execution and failure. All these clusters taken together lay down a strong basis to evaluate load balancing and resource management in realistic cloud settings.

5. PERFORMANCE METRICS

To evaluate the efficiency and stability of the proposed cloud-based scheduling and load balancing system, five essential performance metrics that are given below were computed using Python.

Response Time (RT): It is one of the critical parameters that measures the total time taken by the proposed approach to respond to the task. It mainly represents the responsiveness and scheduling efficiency.

$$RT_i = Startingtime_i - Arrialtime_i$$

Note: The performance of the algorithm improves when the response time is low, primarily due to its scheduling performance.

Execution Time (ET): It represents the overall execution time of the tasks, from the start time to the end time.

$$ET_i = Endingtime_i - Startingtime_S$$

Note: If the execution time is low, then the proposed algorithm demonstrates rapid computation and high efficiency in task scheduling.

Throughput reflects the quantity of work completed per unit of time. That describes the runtime performance and scalability of the system.

$$TP = \frac{N_c}{T_{total}}$$

Note: The higher the throughput, the more tasks can be processed efficiently in a certain time period.

The Task Migration Rate (TMR) represents the frequency of task migration between VMs or nodes. It is indicative of the stability and balance in the scheduling algorithm.

$$TMR = \frac{N_m}{N_t} \times 100$$

Note: The lower the TMR value, the less migration overhead occurs, and thus it contributes to a minimum communication cost and better system stability.

CP Utilization measures how much of all possible CPU capacity is actually used to execute tasks. It measures how the CPU's resources are utilized.

$$CU = \frac{\sum_{i=1}^{N} CPU_{used}(i)}{N \times CPU_{total}} \times 100$$

Note: At one extreme, high CPU usage does mean your computational resources are being utilized to their full

potential, but it can also be an indicator of an overloaded system.

6. RESULTS AND DISCUSSIONS

In this section, the results obtained from the implementation of proposed approach using Python programming language. Table 1 explains a quantitative performance comparison for the five latest scheduling and load-balancing algorithms using with Google Cluster Workload Traces 2019 (ClusterData2019/v3) Job Events cluster dataset. The performance metrics used in this section are: RT, ET, TP, TMR and CPU Utilization which are the measures of effectiveness, responsiveness and stability of the scheduling policy. The IMPROVED MULTI-VERSE OPTIMIZER (IMVO) [17] algorithm of 2023t has the highest response time (430 ms) and execution time (1240ms), exhibiting bad task responsiveness for little computation overhead. Nevertheless, it does possess a moderate number of completed tasks per-second rate of 61 (and reflects inefficient resource usage as its CPU utilization is only 78.2%.) The Quantum Fault-Tolerant Load Balancing (QFTLB) 13 protocol, proposed in 2024, boasts an RT of 392 ms and an ET of 1184 ms with a similar throughput as the IMVO algorithm but less CPU usage thanks to fault recovery through quantum. The Reinforcement Learning Scheduler (RL-Sched) has an excellent improvement, with a response time of 358 ms and throughput of 71 tasks/sec as it can use its learning-based decision-making to adapt task allocation. Recently in 2025, the BBQ–TLELB [13] is introduced as a model that reduces RT down to 310 ms and ET down to 1068 ms with high throughput of 77 task/sec and CPU utilization of %86.1%, it greatly reduces cold start and bottleneck issues. It is lastly observed that the Proposed Secure and Resilient Parallel (SRP) algorithm exhibits better performance in comparison to all factors reporting minimum response time (268 ms) and execution time (973 ms), but with maximum CPU utilization rate 89.3% as well as throughput of 84 tasks/sec. Furthermore, SRP achieves the least number of task migration count (1.7%), showing its ability in balancing loads and reducing overhead for unnecessary task switches. These results clearly show that the SRP excels regarding scheduling intelligence, resource consumption and system stability and performs better than all the other compared algorithms in terms of efficiency as well as resilience on the Job Events cluster especially.

Table 1: Quantitative Performance of Algorithms Applied on Job Events (Cluster)

Algorithm	RT (ms)	ET (ms)	Throughput (tasks/sec)	TMR (%)	CPU Utilization (%)
IMVO (2023)	430	1240	61	3.8	78.2
QFTLB (2024)	392	1184	66	3.2	81.4
RL-Sched (2024)	358	1125	71	2.9	83
BBQâ€“TLELB (2025)	310	1068	77	2.3	86.1
Proposed SRP (2025)	268	973	84	1.7	89.3

Table 2 presents quantitative comparisons of five advanced scheduling algorithms on the Google Cluster Workload Traces 2019 (ClusterData2019/v3) dataset, using the Task Events cluster. This study quantifies key performance indices, including RT, ET, TP, TMR, and CPU Utilization (CU), to assess the efficiency of algorithms for controlling fine-grained task-level operations in large-scale clouds. The IMVO algorithm could respond in 427 ms and complete execution in 1198 ms, with moderate efficiency but limited flexibility for dynamic task changes. Its task-level throughput (58 tasks/s) and CPU utilization (76.7%) suggest potential underutilization of the computing resources, in part due to static scheduling behavior. In contrast, the QFTLB algorithm in 2024 reports a slight performance gain, with an RT of 389 ms and an ET of 1201 ms, by leveraging quantum error correction to stabilize task execution and increase throughput to 63 tasks/sec.

The RL-Sched provides a significant boost, with an RT of 363ms and a throughput of 74 tasks/sec, demonstrating that real-time decisions are made effectively when

running on an adaptive workload manager with optimization models. The BBQ–TLELB algorithm achieves further improvement, with RT decreasing to 298 ms and ET to 1056 ms, while maintaining a high throughput of 79 tasks/sec at 87.8% CPU utilization, indicating significant bottleneck relief and strong inter-layer coordination. However, Table 2 shows that the proposed SRP algorithm performs best across all algorithms on all metrics. It achieves the minimum response time (259 ms), the minimum execution time (969 ms), the maximum throughput of 87 tasks/sec, and a maximum CPU usage of 91.2%. Furthermore, SRP has the smallest average task migration rate of 1.5%, indicating good workload stability and low communication overhead. The results validate that SRP not only improves the efficiency of task processing but also maintains consistent load distribution across parallel nodes, and it is the most responsive, efficient, and stable method for system-level scheduling of tasks in a large-scale cloud environment.

Table 2: Quantitative Performance of Algorithms Applied on Task Events

Algorithm	RT (ms)	ET (ms)	Throughput (tasks/sec)	TMR (%)	CPU Utilization (%)
IMVO (2023)	427	1198	58	3.4	76.7
QFTLB (2024)	389	1184	63	2.9	83.9
RL-Sched (2024)	363	1099	74	2.7	81.8
BBQ-TLELB (2025)	298	1056	79	2.1	87.8
Proposed SRP (2025)	259	969	87	1.5	91.2

7. CONCLUSION

TLELB architecture has been presented in this paper. The first layer, named Fast Layer, combines several heterogeneous scheduling algorithms in an ensemble to achieve low latency, dynamic task allocation, and QoS improvement. BBQ - a second layer that avoids bottlenecks, cold-starts, and load imbalance by dynamically redirecting workload between workers using real-time feedback. Additionally, the inclusion of the SRP algorithm contributes to a reliable system by enabling transport data monitoring, task parallelization, and fault-tolerant/security processing. Experiments on the Google Cluster Workload Traces (2019) show that the proposed method achieves significant improvements in response time, throughput, CPU utilization, and system stability compared with baselines. In summary, the TLELB model is a powerful and intelligent method to maintain high efficiency, resilience, and reliability in contemporary cloud system environments.

Conflict of interest statement

Authors declare that they do not have any conflict of interest.

REFERENCES

[1] L. Sliwko, "Cluster Workload Allocation: A Predictive Approach Leveraging Machine Learning Efficiency," IEEE Access, vol. 12, pp. 194091–194107, 2024, doi: https://doi.org/10.1109/access.2024.3520422.

[2] 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.

[3] Perumal Geetha, S.J. Vivekanandan, R. Yogitha, and M.S. Jeyalakshmi, "Optimal load balancing in cloud: Introduction to hybrid optimization algorithm," Expert Systems with Applications, vol. 237, pp. 121450–121450, Sep. 2023, doi: https://doi.org/10.1016/j.eswa.2023.121450.

[4] Y. Y. Raghav, V. Vyas, and H. Rani, "Load balancing using dynamic algorithms for cloud environment: A survey," Materials Today: Proceedings, Sep. 2022, doi: https://doi.org/10.1016/j.matpr.2022.09.048.

[5] S. M. Moghaddam, M. O'Sullivan, C. P. Unsworth, S. F. Piraghaj, and C. Walker, "Metrics for improving the management of Cloud environments — Load balancing using measures of Quality of Service, Service Level Agreement Violations and energy consumption," Future Generation Computer Systems, vol. 123, pp. 142–155, Oct. 2021, doi: https://doi.org/10.1016/j.future.2021.04.010.

[6] J. N. Witanto, H. Lim, and M. Atiquzzaman, "Adaptive selection of dynamic VM consolidation algorithm using neural network for cloud resource management," Future Generation Computer Systems, vol. 87, pp. 35–42, Oct. 2018, doi: https://doi.org/10.1016/j.future.2018.04.075.

[7] S. Iftikhar et al., "HunterPlus: AI based energy-efficient task scheduling for cloud–fog computing environments," Internet of

Things, vol. 21, p. 100667, Apr. 2023, doi: https://doi.org/10.1016/j.iot.2022.100667.

[8] Ipsita Behera and Srichandan Sobhanayak, "Task scheduling optimization in heterogeneous cloud computing environments: A hybrid GA-GWO approach," Journal of parallel and distributed computing, vol. 183, pp. 104766–104766, Jan. 2024, doi: https://doi.org/10.1016/j.jpdc.2023.104766.

[9] R. R. Dornala, "An Advanced Multi-Model Cloud Services using Load Balancing Algorithms," 2023 5th International Conference on Inventive Research in Computing Applications (ICIRCA), Coimbatore, India, 2023, pp. 1065-1071, doi: 10.1109/ICIRCA57980.2023.10220892.

[10] S. Ponnapalli, R. R. Dornala and S. P. Vallabaneni, "A Triple-Tap Hybrid Load Balancing System (TTHLB) for Health Monitoring System," 2023 7th International Conference on I-SMAC (IoT in Social, Mobile, Analytics and Cloud) (I-SMAC), Kirtipur, Nepal, 2023, pp. 616-622, doi: 10.1109/I-SMAC58438.2023.10290416.

[11] R. R. Dornala, S. Ponnapalli, K. T. Sai, S. R. K. R. Koteru, R. R. Koteru and B. Koteru, "Quantum based Fault-Tolerant Load Balancing in Cloud Computing with Quantum Computing," 2023 3rd International Conference on Innovative Mechanisms for Industry Applications (ICIMIA), Bengaluru, India, 2023, pp. 1153-1160, doi: 10.1109/ICIMIA60377.2023.10426349.

[12] Sudhir Ponnapalli, Raghunadha Reddi Dornala, K. T. Sai, and K. Reddi, "A Light-Weight Data Storage and Delivery Platform in Cloud Computing," Algorithms for intelligent systems, pp. 199–209, Jan. 2024, doi: https://doi.org/10.1007/978-981-97-1488-9_16.

[13] S. E. Shukri, R. Al-Sayyed, A. Hudaib, and S. Mirjalili, "Enhanced multi-verse optimizer for task scheduling in cloud computing environments," Expert Systems with Applications, vol. 168, p. 114230, Apr. 2021, doi: https://doi.org/10.1016/j.eswa.2020.114230.

[14] AR. Arunarani, D. Manjula, and V. Sugumaran, "Task scheduling techniques in cloud computing: A literature survey," Future Generation Computer Systems, vol. 91, pp. 407–415, Feb. 2019, doi: https://doi.org/10.1016/j.future.2018.09.014.

[15] G. Rjoub, J. Bentahar, and O. A. Wahab, "BigTrustScheduling: Trust-aware big data task scheduling approach in cloud computing environments," Future Generation Computer Systems, vol. 110, pp. 1079–1097, Sep. 2020, doi: https://doi.org/10.1016/j.future.2019.11.019.

[16] T. Kavitha, S. Hemalatha, T. M. Saravanan, A. K. Singh, M. I. Alam and S. Warshi, "Survey on Cloud Computing Security and Scheduling," 2022 International Conference on Computer Communication and Informatics (ICCCI), Coimbatore, India, 2022, pp. 1-4, doi: 10.1109/ICCCI54379.2022.9740932.

[17] M. S. Sanaj and P. M. Joe Prathap, "An efficient approach to the mapreduce framework and genetic algorithm based whale optimization algorithm for task scheduling in cloud computing environment," Mater. Today, Process., vol. 37, pp. 3199–3208, Oct. 2021.