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

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

A HYBRID MULTIPLE ENSEMBLE LOAD BALANCING FRAMEWORK FOR DECISION-MAKING IN REAL-TIME MULTI-SERVER, MULTI-TASK SCHEDULING IN CLOUD ENVIRONMENTS

GUTTA SRIDEVI[1], TUMMA SRINIVASA RAO[2], GUNTAPALLI MINNI[3], HARISH VUNDAVALLI[4], RAGAVAMSI DAVULURI[5], SAMEENA B[6]

[1]Professor, Dept. of CSE, Seshadri Rao Gudlavalleru Engineering College, Gudlavalleru, Andhra Pradesh, India
[2]Professor, Dept. of CSE, Seshadri Rao Gudlavalleru Engineering College, Gudlavalleru, Andhra Pradesh, India
[3]Professor, Dept. of CSE, Lakireddy Bali Reddy College of Engineering, Mylavaram, Andhra Pradesh, India
[4]Sr.Technical Architect Strategic Education INC, Dept. of IT, Minnesota, USA.
[5]Associate Professor, Dept. of CSE, Seshadri Rao Gudlavalleru Engineering College, Gudlavalleru, Andhra Pradesh, India.
[6]Student, Department of CSE Honors through Research, Koneru Lakshmaiah Education Foundation, Guntur, Andhra Pradesh.

E-mail: guttasridevi144@gmail.com, srinivas123fast@gmail.com, minni.guntapalli@gmail.com, harish.vundavalli@strategiced.com, raaga.vamsi@gmail.com, sameena311205@gmail.com

ABSTRACT

As the number of tasks and virtual machines increases for task scheduling in cloud computing environments, predicting an efficient load balancing model for multi-server, multi-task scheduling becomes challenging. Many conventional models rely on static multi-task scheduling algorithms for decision-making. Additionally, these models utilize traditional classification algorithms for generating multi-task patterns, often working with limited data sizes. This work presents a hybrid load balancing framework designed to enhance decision pattern mining in the multi-server, multi-task scheduling process. The framework integrates Particle Swarm Optimization (PSO) and Ant Colony Optimization (ACO) into the ensemble load balancing approach within a real-time cloud computing environment. Furthermore, a hybrid decision tree classifier is proposed to facilitate decision-making in the multi-task scheduling process. Experimental results demonstrate that the proposed model outperforms conventional models in terms of multi-task, multi-server scheduling in real-time cloud computing environments.

Keywords: *Multi-task, Multi-server, Load balancing, Cloud computing, Classification Algorithm.*

1. INTRODUCTION

As organizations progressively migrate into the cloud environment, data and resource monitoring and data management framework generated have increased exponentially. The cloud service provider is responsible for resolving and overcoming issues that arise dynamically within the public cloud model. [1] defines the Public Cloud as an infrastructure that can be used by or owned by any organization or institution by the public. It has a cloud service provider and is managed by it. Public cloud is popular with very large customers for easy access and simplicity. Load balancing requires efficient resource management and data management for dedicated purposes. The load balance can be defined as a processor's ability to schedule processors to ensure that all processors and resources are engaged during or until the command stream is active. The customer requests are calculated for the specific resources through load balancing and the demands are performed. In the static balanced method, the available resources and parameters (metrics, network and CPU bandwidth, server capacity, etc) are first considered and later the programming of the operating

workload will be completed. Figure 1 shows the performance of the load balancer in public cloud model. Some of the most popular algorithms for static scheduling include a ring-robin algorithm, central manager algorithm, threshold algorithm [2], etc., whereas the load balancing device chooses the appropriate and optimum resources to perform the tasks, based on client requests and the size of the jobs. Because of its powerful architecture for performing large-scale and complex computing, cloud computing has become an inspiring technology. It makes it easier to get on to network access to an infinite number of computing resources like CPUs, memory, and storage [3]. With the least amount of management effort, these tools are provisioned and published easily. Cloud computing's most important services are reliability, scalability, elasticity, and high availability, making it a highly dynamic and wide distributed system. Cloud computing services are classified as infrastructure, platform, and software as a service to provide these facilities for processing massive amounts of data. Customers can access these services on a pay-per-use basis [4]. Task scheduling methods are critical in cloud computing environments because a large number of users post their computing tasks on the cloud system.

The problem of execution, finishing time, and precedence order in a parallel processing system were solved in the conventional method by using the principle of top-level or bottom-level[5]. The question of handling a large number of activities must be discussed. The problem of dynamic workload assignment plagues search-based approaches, resulting in a very wide search space. The event-based scheduler (EBS) and ACO methods do not deal with ambiguity, which can arise at any time during the task planning process. The task guided acyclic graph (DAG)[6] can be used to run a parallel system in any static scheduling process. A node in a DAG represents a task, which is a set of interconnected tasks that must run in order without pre-empting their dependent tasks. Without violating the RCMPSP limitations, [7] combined a scientific genetic algorithm with a local searching technique, with the ideal feature being total task delay and/or resource output time. [8] proposed a simple genetic algorithm for multi-task scheduling. This algorithm could simultaneously reduce any delay, quickness, and time violation. [9] defined a hierarchical two-level system that included programming and scheduling procedures. The aim of the knapsack problem is to find a combination of tasks that contributes the most to shortening the planned task period while scheduling them at the same time does not breach resource constraints. The solution to the problem can be solved using heuristic methods, and the solution decides the task(s) to be scheduled in time [10].

A multi-objective problem like software task scheduling does not have a single optimal solution in the single objective optimization function. That is, it is not possible to minimize the task's total expense without optimizing its resources using a single solution to the SPS. The need for a reliable, scalable and flexible environment has led to a new era of cloud computing. NIST[11] defines cloud computing as a model to allow a shared pool of configurable dedicated resources to be conveniently used on demand and that can be delivered quickly with minimal management or provider interaction. It is an agile, innovative and efficient system that uses mainstream virtualization technology. Available algorithms include a round robin improvised by the CLBDM, the Low Balancing Min Max techniques (LBMM), and the optimization algorithm for Ant colony. It offers mainly load balancing algorithms and cloud techniques for the efficient use of resources. [12] Describes several proposed algorithms to resolve problems related to cloud load balance and task scheduling. The comparison shows that static load balance algorithms are more constant than dynamic algorithms, but that dynamic load balancing is chosen over static charge balance algorithms due to the ability to perform correctly in distributed systems. This analysis can also contribute to the development of new algorithms for load balance. [13] Discuss computer, system and message-oriented model load balancing. [14] A scenario of the hospitals management system examines the availability features of cloud services and the data recruitment system and also manages the use of resources is presented with an overview of transaction metrics and activities. [15] Provide problems and limitations of traditional cloud load balancing technology, and also provide a generalized framework to create the cloud environment for a scalable architecture for the cloud. In order to support the sizeable load balance architectures, an overview of the best practices of the Amazon web service is available. Examination of issues relating to the development of the distributed system dynamic load balancing algorithm. The Guide provides a guide to critical issues that must be tackled when it is a question of developing and studying the dynamic load balance algorithm. In public cloud computing, load balance is the main problem. The public cloud computing

infrastructure includes hardware, software and platform for public demand. The process uses job planning and task scheduling for handling multiple requests of the user cloud computing process[16]. The full load here is assumed to be arbitrarily partitionable in a way that could be divided into load fractions to be assigned to all the cloud master and slave computers. In this case, each master computer assigns to each corresponding N slave computer a load share that is to be measured, and then gets measuring data from each slave. After the measuring instructions from the respective master are received by each slave each slave begins to measure its load part. We also take for granted that time compared to communication and measurement is negligible[17]. Cloud computing provides much more centralized memory, processing, bandwidth and storage computing. It must ensure that tasks are not heavily loaded on a single VM and that certain VMs do not remain idle and/or under loaded. Data and apps are kept via the Internet and central remote servers in cloud computing technology. The first type is Batch mode heuristic algorithms of programming (BMHA) and the second type is heuristic algorithms of mode online. A global load balancing algorithm through local Server Action. Load balancing[18] inspired by Honey Bee Behavior is a technique used to maximize the performance of even load balances across virtual machines. Honey bee's load balancing algorithm ensures that virtual machines achieve good equilibrium by optimizing their performance and reducing response time[16].When they arrive in the system, jobs are combined in BMHA. After a set period of time, the BMHA programming algorithm begins. Examples include: First Come First Served Scheduling (FCFS), Round Robset Scheduling (RR) algorithms, the Min Min and Max Min Algorithms, etc. Examples of BMHA-based algorithms are: All jobs are scheduled when they arrive in the system in the heuristic planning algorithm. The cloud is a heterogeneous system, which varies quickly and easily in each processor's speed.

Despite the availability of several traditional and heuristic load balancing algorithms such as Round Robin, Min-Min, Max-Min, Honey Bee behavior-based methods, Ant Colony Optimization, and Genetic Algorithms, many existing approaches suffer from limitations related to scalability, dynamic workload handling, response time optimization, and efficient resource utilization in highly distributed cloud environments. Most of the existing techniques focus either on static scheduling or single-objective optimization, which reduces their effectiveness in real-time public cloud infrastructures where workload conditions change dynamically. Existing load balancing methods such as Round Robin, Min-Min, and PSO mainly focus on single-objective optimization and often fail to efficiently handle dynamic workloads in heterogeneous cloud environments. These approaches may result in increased execution delay, poor resource utilization, and scalability issues during multi-server multi-task scheduling. To overcome these limitations, the proposed hybrid IPSO-IACO framework introduces intelligent task scheduling with improved QoS, adaptive resource allocation, and efficient load balancing for real-time cloud computing environments.

The primary contribution of this research is the development of an enhanced load balancing and task scheduling framework for public cloud environments that improves resource utilization, minimizes execution time, and reduces response delay under dynamic workloads. Unlike existing approaches, the proposed framework integrates intelligent scheduling strategies with optimized resource allocation mechanisms to achieve better scalability and performance efficiency. The proposed work also emphasizes effective virtual machine utilization and balanced task distribution to avoid overloading and underutilization of cloud resources. Furthermore, this study provides a comparative analysis of existing load balancing techniques and demonstrates how the proposed approach achieves improved Quality of Service (QoS) parameters such as throughput, response time, and system efficiency in cloud computing environments.

2. RELATED WORKS

[19], In 2018, a new model for replication-based load balancing was proposed, where a cloud service provider (CSP) distributes the entire job queue across a cloud environment, as well as a private resource, to maximize profits. This model utilizes a load balancing technique based on job replication. The job queue is divided among several sub-clouds, with each sub- cloud being assigned to a virtual machine (VM) to execute the tasks. The VMs communicate with each other—if one completes its task early, it notifies the others, prompting them to stop processing and remove tasks from their respective queues. However, this approach leads to significant energy waste, as many tasks are unnecessarily handled by multiple VMs. Additionally, the overall execution time is longer

compared to other algorithms. The first sub-cloud to finish the task handles the remainder of the workload, increasing execution inefficiency.

[20], The 2019 study titled "Dynamic Load Balancing and Job Replication in a Global-Scale Grid Environment: A Comparison" introduces a Dynamic Resource Allocator (DRA) for cloud service providers. The proposed strategy, known as skewness, helps prevent server overloading by managing varying workloads. This method improves resource scheduling by analyzing differences in resource utilization across servers. Since server resources can be unevenly consumed, the skewness strategy calculates these disparities to facilitate load balancing. By determining skew values for all active virtual machines (VMs), it is possible to forecast future workloads. The strategy also incorporates VM migration to address system overload conditions and employs load prediction techniques to further optimize resource distribution.

[21], The 2018 study, "Offloading Interrupt Load Balancing from SMP Virtual Machines to Hypervisor," presents a strategy that operates with a diverse configuration of virtual machines (VMs). Tasks are assigned to VMs based on their priority, with the processing speed in memory being used to determine these priorities. The load balancer allocates tasks to the highest-priority VM, while a dedicated panel keeps track of the number of active VMs and updates them as tasks are assigned and distributed. In this setup, resources are unevenly distributed among VMs, meaning low-priority VMs might remain idle for extended periods, while more critical VMs handle higher workloads. To improve this, the strategy suggests maintaining the least-priority VM active to achieve a better balance. The proposed approach uses a non-preemptive scheduling method via the Nephele scheduler, where the critical slot required for each task is calculated. Tasks are assigned based on their criticality, with the highest-priority VM handling the most time-sensitive tasks, which are then removed from the task queue. The scheduler evaluates task efficiency after each cycle. This algorithm assumes a heterogeneous environment, with various server types and numbers forming a computational cluster for problem-solving. New jobs are grouped under this strategy, aiding the service manager in creating clusters of tasks requiring similar resources. Dedicated VMs are then assigned to these clusters for sequential execution, ultimately aiming to reduce task completion time while maximizing resource utilization.

[22]. This paper focuses on minimizing the makespan and enhancing throughput using Particle Swarm Optimization (PSO) alongside the Adaptive Load Balancing (ALB) technique [23]. By integrating adaptive scheduling with PSO, the approach effectively handles both overloaded and under loaded conditions simultaneously. The load balancing mechanism ensures even distribution of system workloads by evaluating the load on each virtual machine (VM) and reallocating tasks based on the VM's status and task deadlines. In [24], an improved PSO strategy is presented, which adapts the load as iterations progress and incorporates random weights at a later stage. This adjustment helps prevent the particle swarm from getting stuck in local optima, allowing for better performance in advanced stages. The proposed strategy was applied to task scheduling, aiming to optimize cloud computing scheduling. Experimental results using the CloudSim demonstrate that the proposed algorithm surpasses the traditional PSO in terms of overall performance. The algorithm assigns tasks to VMs based on the length of the submitted tasks and the VM's processing capacity. The implementation results show that the Bin-LB-PSOGSA method improves load distribution over time, while reducing VM processing speed, indicating better efficiency in maintaining load balance. Additionally, [25] introduces a PSO-based approach where tasks are independent and non-preemptive. By employing load balancing, this method enhances the performance of the standard PSO algorithm. When compared to the conventional PSO method, the proposed one reduces the makespan by 33% and improves resource utilization by 22%. Moreover, [26] suggests an adaptive, multi-objective task scheduling algorithm utilizing PSO techniques to reduce both processing and transmission time.

Existing load balancing and task scheduling methods improve cloud resource management, but they still face limitations such as poor adaptability to dynamic workloads, high computational overhead, local optima problems, inefficient resource utilization, and increased execution delay in large-scale cloud environments. Additionally, many approaches focus only on single-objective optimization and fail to effectively manage multiple QoS parameters and heterogeneous virtual machine environments simultaneously. To overcome these limitations, the proposed work introduces a hybrid ensemble load balancing framework that integrates Improved Particle Swarm Optimization (IPSO), Improved Ant Colony Optimization (IACO),

statistical resource collection mechanisms, and a hybrid decision tree classifier for intelligent task scheduling and decision-making. Unlike conventional methods, the proposed approach supports dynamic multi-server and multi-task scheduling with improved scalability and adaptive resource allocation. The integration of statistical collectors with hybrid meta-heuristic optimization enables efficient workload distribution, minimizes execution delay, improves virtual machine utilization, and enhances overall Quality of Service (QoS) performance in real-time cloud computing environments.

3. MATERIALS AND METHODS

This paper presents a hybrid ensemble load balancing model with a variable statistical collector, implemented on real-time cloud servers. The model utilizes various cloud virtual machines (VM 1, VM 2, ... VMn) and a set of tasks (Tk-1, Tk-2, ... Tk-m) alongside cloud resources (R 1, R 2, ... Rm) as input for effective load balancing. The statistical collector is responsible for gathering all necessary cloud resources for task scheduling. A meta-heuristic approach is introduced to enhance the overall load balancing process. As illustrated in Figure 1, two meta-heuristic algorithms—hybrid ACO (Ant Colony Optimization) and hybrid PSO (Particle Swarm Optimization)—are incorporated into the framework to improve the load balancing of multiple large tasks in a cloud computing environment. Lastly, a decision tree classification technique is proposed to make dynamic decisions on multi-task and multi-server patterns.

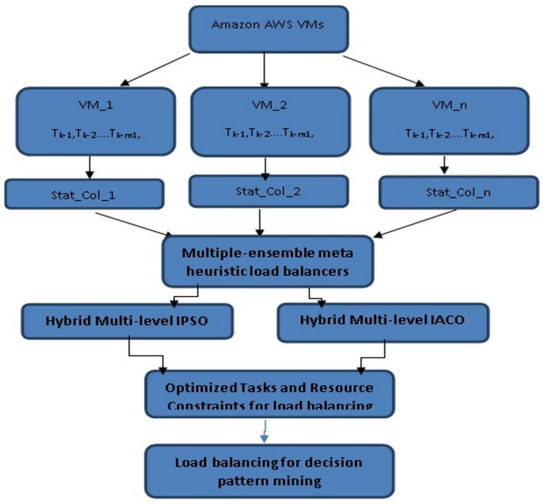

Figure 1. *Multiple Load Balancing Framework for Multi-Task Cloud Server Management*

3.1 Hybrid Ensemble Meta heuristic Multi-Cloud Load balance estimation

1. Initializing of cloud virtual machines VM_i for n tasks, m resources and k load balancers.
2. To each virtual machine VM_i
3. do
4. Assign n tasks to each virtual machine.
5. Determine the optimal cloud services and their metrics by applying integrated ensemble IPSO (Improved Particle Swarm Optimization) and IACO (Improved Ant Colony Optimization) optimization techniques. The Quality of Service (QoS) metrics, along with the initialization of cloud servers and tasks, are assessed using the specified measures.

$$MV(f_i+1,i)=\phi.[e^{\log(f_i)}.MV(f_i,i).\xi_{c1}(pB_i-MV(f_i,i)+\xi_{c2}(gB_i-MV(f_i,i)]$$

ϕ is the multi-cloud server based task assignment decision variable.

$$\phi=\frac{e^{-m^2}\left(\log(\alpha_i-\alpha_j)\right)}{|m*f_i-(\xi_{c1}+\xi_{c2})-\max\{\xi_{c1},\xi_2\}.\sqrt{(\xi_{c1}+\xi_{c2})^2-4(\xi_{c1}+\xi_{c2})}|};m<=2;$$

where ξ_{c1},ξ_{c2} are the randomized cyclic group elements.

$$\xi_{ci}=\frac{1}{\sqrt{2\pi}}e^{-\max\{\square c_i-c_j\square\},\frac{(c_i-\mu_i)^2}{\sigma_i^2}}$$

$i=1,2...iterations$

6. Initialize each ant with the default properties, repeat

7. Initialize all ant solutions as true, Construct neighbor nodes and its paths until best solutions. Compute

heuristic rule for node pheromone as

$\varphi : \rho_w$: Pheromone weight

$P_\varphi : \rho_{cw}$: Current pheromone value

$PN_\varphi : \rho_n$ Neighbor pheromone value

$\phi : H_w$: Heuristic weight

$H_\phi : H_{cw}$: Current heuristic weight

$HN_\phi : H_n$: Neighbor heuristic value

Load balacing Probability of each serveer is computed as

Overall load time taken by each server is given as

$$TP(N_c) = 1 - \left(\exp(\rho_n^{\sqrt{\rho_w}}) \times \frac{\rho t_i - \rho t_{\min}}{\rho t_{\max} - \rho t_{\min}} + H_n^{\sqrt{H_w}} \times \log\left(\frac{H_n c_i - H_n c_{\min}}{H_n c_{\max} - H_n c_{\min}}\right) \right)$$

8. Update optimize ant heuristic rule probability, local and global best computations

$$RP(\varphi,\phi) = \min\{\upsilon^2\sigma^2 + \exp\left(\log(\sigma * \upsilon) * \frac{1}{\sigma}\right),$$

$$\frac{(m-p)^{\varphi-1}(q-x)^{\phi-1}}{B(\varphi,\phi)(q-p)^{\varphi+\phi-1}}\} \text{ for } p \le m \le q, \text{ and } 0 \text{ elsewhere}$$

$B(\varphi,\phi)$: Beta distribution

$$\upsilon = \text{Variance} = V(RP(\varphi,\phi)) = \frac{\exp(\varphi.\phi)}{(\varphi+\phi)^2(\varphi+\phi+1)}$$

$$\sigma = SD = SD(RP(\varphi,\phi)) = \sqrt{\frac{\exp(\varphi.\phi)}{(\varphi+\phi)^2(\varphi+\phi+1)}}$$

$$\text{Randomized Rule probability} := RP = PN_\varphi^{\upsilon^2} * \log(HN_\phi^{\sigma^2})$$

9. Repeat to each node in the cloud servers.

Local best positioning update

$$ED(\chi) := f(\chi) = \chi e^{-\chi x} \quad \text{for } x \ge 0$$

$$\omega = \varphi * (1 + (1 - ED(\chi)) * (1 - \varphi) * \phi)$$

$$L = (1 - ED(\chi)) * \varphi + ED(\chi) * \omega$$

Global best positioning update:

r1,r2 node initialization parameters

$\eta = (1 + \varphi * \phi) * \varphi;$

$$CD(r1,r2) := f(x) = \frac{r2}{\pi[(x - r1)^2 + r2^2]}$$

$$GU = (1 - CD(r1,r2)) * \varphi + CD(r1,r2) * \eta$$

10. Computing the fitness function to each metric in the selected metrics space.

Let $R_{t\min}$ and $R_{t\max}$ are the minimum and maximum response time of the task in each instance VMi.

$$F(c) = MV(cf_i+1,i)*(R_{t\min} \rho + R_{t\max}(1-P_{cf_i}).Wq)$$

$$W_q = \max\{ VM(R_t,i)\}.T_\mu / \lambda_R \quad \text{Where}$$

$|VM|$= Total number of virtual machines.

$|T|$=Total number of tasks.

$$\lambda_R = VM(\mu,i).(1-P_{cf_i})$$

$$P_{cf_i} = (\lambda_R{}^n / |VM|!\ |T|.\mu_R{}^n)$$

11. Repeating the steps 5 and 11 to each dynamic response time and the cloud metrics.

3.2 Statistical Tasks and Meta heuristic Resource Constraints for load balancing

i) Apply Improved PSO task scheduling algorithm as follows.

1. Initialize IPSO features in the cloud computing environment.
2. Compute QoS predicted on demand cost of each VM using V_{time} as metric.
3. Let W_u and W_t be the cloud price rates per hour charges for allocating various resources and the final
4. cost function is shown below
5. $F(c) = V_{time}(W_u s \mu + W_t (1-p_k)\lambda e Wq)$
6. $W_q = L_q / \lambda_e$
7. $L_q = \sum_{n=s}^{k}(n-s).P_n$ where Effective Arrival Rate λ_e:
8. $\lambda_e = \lambda (1-P_k)$
9. $P_n = (\lambda^n / S!\ S^{n-s}\ \mu^n)\ P_o$, when $s \le n \le k$
   λ = the arrival rate for the system,
   μ = the service rate for each server
   s = Number of Servers
   K = System size (Finite); Total number in system ≤ k
   Let P_n be the probability that there are n customers in the system in the steady-state. n = 0, 1, 2, 3… k.

   $$\text{GaussianValue}: gv = \frac{1}{2.\pi} e^{\sum \log(gbest-pbest)}$$

   $$\text{UpdatedVelocity}:$$

10. $$V_{i+1} = \log(w * v_i) + C.(V_{time}) * (pbest_i - x_i) + (C * gv / PS)) * (\exp(gbest_i) - x_i)$$
    $w = 0.9$
    $PS :=$ ParticleSize
11. Repeat until all tasks are scheduled

$$\text{Min} \sum_{i=1}^{N} B_i.\log\{\psi_{VMj}\}/|R_i| \text{ and Max} \sum_{j=1}^{M} B_j.\exp(\psi_{VMj})/|R_i|$$

The average response time of the server is given as

$$T_{resp} = \frac{\sum_{j=1}^{n} r_j}{n}$$

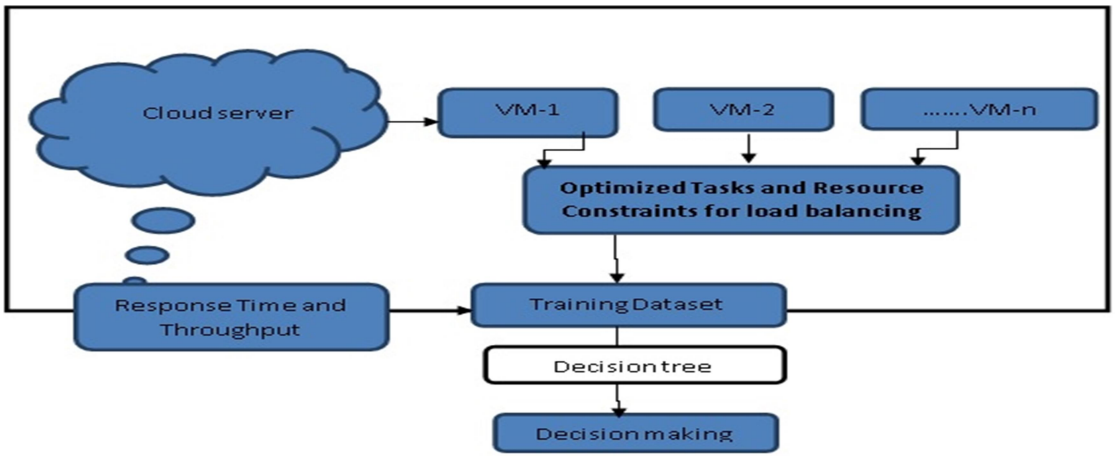

Figure 2. Cloud load balancing based decision-making framework

The variables considered in this work were selected based on their direct influence on cloud load balancing and task scheduling performance in real-time environments. Parameters such as runtime, CPU utilization, memory utilization, response time, job idle time, and resource request rate were chosen because they significantly affect Quality of Service (QoS), task execution efficiency, and virtual machine performance. Runtime and response time help evaluate scheduling efficiency, while CPU and memory utilization measure resource consumption and workload distribution among cloud servers. Job idle time and request metrics are included to identify under loaded and overloaded virtual machines during dynamic scheduling. These variables collectively support effective decision-making and improve adaptive resource allocation in the proposed hybrid IPSO-IACO framework.

The input data is first prepared according to various class attributes. Nominal attributes are transformed into binary attributes, and missing values are imputed with mean values. This processed data is then used for classification to enhance task prediction accuracy. Different nominal attributes serve as class labels in the decision-making process, as illustrated in Figure 2.

ii) Apply Random Forest algorithm as follows.

Step 1: Input file (Filtered data)

Step 2: Preprocess anomaly data for missing values.

Step 3: Data transformation using the normalization approach.

Step 4: For each randomized sample S_i do

$$B_f = \text{uniqueCV}(D);$$

$$DB_f = \text{Histobins}[] = \text{discretizationbins}(D)$$

$$\text{GaussianlogKernel}: GLK(\phi,\theta) = e^{-Log(\theta^2)} / (2 * \exp(\phi^2))$$

$$\psi l = gkv = GLK(\sum DB_f, \sum B_f);$$

$$\text{Kernel Probability} = KP(D) = | DB_f / (\sum \psi l * DB_f) |$$

$$\text{GaussianLogEntropy}: GLE(d_i) = -GLK(\sum_i \exp(d_i).\log(d_i), \mu_d)$$

In the equations mentioned above, Gaussian entropy is utilized to evaluate the feature entropy value using a Gaussian estimator.

Proposed entropy formula:

$$PE = e^{-D^2} / (2 * D_1^2) * | DB_{D_1} / (\sum \psi l * DB_{D_1}) |$$
$$+ e^{-D^2} / (2 * D_2^2) * | DB_{D_2} / (\sum \psi l * DB_{D_2}) |$$

In the load balancing pattern detection model described above, each attribute is evaluated against the data distribution. If an attribute is found to be non-uniformly distributed, it is transformed into a uniform format. Once the dataset is uniformly distributed, the instances are divided into sub-partitions based on their respective classes. Similarity calculations are then performed on these sub-partitions to identify relevant relational anomaly features. High anomaly patterns in each partition are detected using true probability and false probability measures.

4. EXPERIMENTAL RESULTS

The experimental results were simulated using Amazon AWS servers and cloud service training

data. These results were developed in a NetBeans and JDK environment, utilizing various third-party libraries. The experiments were conducted in real-time on an Amazon AWS cloud server, employing different types of virtual machines—such as micro, medium, and large instances—as inputs for the task scheduling process. The AWS Java API was used in this study to manage balancing parameters. Novel statistical collection-based load balancing algorithms were tested in a real-time cloud environment. The experiments evaluated cloud servers for memory usage, CPU utilization, and runtime computation. This section details the outcomes, where tasks were executed across virtual machines of varying sizes (small, medium, and large). Dynamic Load Balancing (DLB), Load Bayes Clustering (LB-BC), and the Firefly Algorithm yielded comparable traditional results.

Table 1. Sample data collected using the statistical collector in cloud computing environment

Instance No	Backet name	Region	Load balancer ID	Data req	Data res	Load time	Req time	Job idle time	CPU utilization	Mem req	Job runtime
CloudVM#1	Buckt-112.0	USEas(Ohio),USWest(N.California)	LoadBal_ID197.0	21	269	24	4578	14	3	22	742
CloudVM#2	Buckt-194.0	SouthAmerica(SãoPaulo)	LoadBal_ID143.0	39	292	26	4050	11	2	10	373
CloudVM#3	Buckt-133.0	USEast(N.Virginia)	LoadBal_ID136.0	44	258	26	2771	8	5	32	622
CloudVM#4	Buckt-185.0	AsiaPacific(Mumbai)	LoadBal_ID167.0	44	258	26	2771	8	5	32	622
CloudVM#5	Buckt-132.0	USEas(Ohio),USWest(N.California)	LoadBal_ID162.0	48	300	44	2724	21	4	24	692
CloudVM#6	Buckt-121.0	USEast(N.Virginia)	LoadBal_ID114.0	21	317	26	2354	23	3	50	303
CloudVM#7	Buckt-121.0	AsiaPacific(Mumbai)	LoadBal_ID148.0	34	254	29	3101	20	5	35	651
CloudVM#8	Buckt-144.0	USEas(Ohio),USWest(N.California)	LoadBal_ID154.0	27	238	31	3609	20	3	32	440
CloudVM#9	Buckt-189.0	USEas(Ohio),USWest(N.California)	LoadBal_ID220.0	34	299	44	1858	19	3	22	603

Table 2: Sample training data of the cloud metrics

AVAILABILITY	MAX_RT	MIN_RT	AVG_RT	THROUGH	SERVICE_ID	IDX	COST	JOB_ID	TASK_INDEX	TASK_USAGE_IDX
363	3758	1146.36	120	1.96	41	1012	3.432794	39308692	0	752165
252	1945	806.03	166	2.71	41	1014	5.328627	39308692	1	898663
322	6817	1463.86	123	2.02	41	1017	1.293431	39308692	1	95
95	689	255.51	269	4.45	41	1021	3.432794	39308692	0	752165
222	2105	639.1	199	3.3	41	1023	3.432794	39308692	0	752165
109	1461	385.64	223	3.69	41	1027	1.293431	39308692	1	95
108	1436	361.83	227	3.71	41	1028	3.313186	39308692	1	752166
167	1733	449.48	209	3.44	41	1030	1.928611	39308692	1	132267
182	6915	863.31	156	2.54	41	1032	1.704608	39308692	0	613661
4565	9430	6759.34	32	0.52	42	1037	6.958529	46542951	1	898665
5309	8361	6679.89	37	0.61	42	1039	1.856176	46542951	0	132268
3031	9045	5113.02	39	0.62	42	1042	1.265173	46542951	0	421856
3132	8797	6008.46	30	0.49	42	1043	1.649461	46542951	1	97
3151	8521	6196.94	37	0.59	42	1047	6.502647	46542951	1	614260
95	577	223.55	279	4.63	42	1049	6.958529	46542951	1	898665
4506	9410	6297.02	34	0.55	42	1052	20.68987	46542951	0	277529
113	1725	286.63	233	3.86	42	1058	6.958529	46542951	1	898665
92	6957	696.28	169	2.74	42	1060	6.958529	46542951	1	898665

Table 1 and Table 2 are the sample datasets used to estimate the decision of the load balancer for the task scheduling and resource constraints. The decision patterns generated using the Table 2 are represented in Table 3.

Table 3. Patterns for load balancing

```
JOB_ID = '(-inf-64002520.2]'
|  IDX = '(-inf-278.1]'
|  |  TASKUSAGEIDX = '(-inf-90235.3]'
|  |  |  COST = '(-inf-11.899706]'
|  |  |  |  CLOUDS_ID = '(-inf-10.9]'
|  |  |  |  |  CLOUDAVGRT = '(-inf-37]' : 0 (0/0)
|  |  |  |  |  CLOUDAVGRT = '(37-73]' : 0 (0/0)
|  |  |  |  |  CLOUDAVGRT = '(73-109]' : 0 (0/0)
|  |  |  |  |  CLOUDAVGRT = '(109-145]' : 0 (0/0)
|  |  |  |  |  CLOUDAVGRT = '(145-181]' : 0 (0/0)
|  |  |  |  |  CLOUDAVGRT = '(181-217]'
|  |  |  |  |  |  CLOUDVMMAXRT= '(-inf-4942.5]' : 2 (1/0)
|  |  |  |  |  |  CLOUDVMMAXRT= '(4942.5-9885]' : 0 (1/0)
|  |  |  |  |  |  CLOUDVMMAXRT= '(9885-14827.5]' : 0 (0/0)
|  |  |  |  |  |  CLOUDVMMAXRT= '(14827.5-19770]' : 0 (0/0)
|  |  |  |  |  |  CLOUDVMMAXRT= '(19770-24712.5]' : 0 (0/0)
|  |  |  |  |  |  CLOUDVMMAXRT= '(24712.5-29655]' : 0 (0/0)
|  |  |  |  |  |  CLOUDVMMAXRT= '(29655-34597.5]' : 0 (0/0)

|  |  |  |  |  |  CLOUDVMMAXRT= '(34597.5-39540]' : 0 (0/0)
|  |  |  |  |  |  CLOUDVMMAXRT= '(39540-44482.5]' : 0 (0/0)
|  |  |  |  |  |  CLOUDVMMAXRT= '(44482.5-inf)' : 0 (0/0)
|  |  |  |  |  CLOUDAVGRT = '(217-253]' : 0.25 (4/0.19)
|  |  |  |  |  CLOUDAVGRT = '(253-289]' : 1 (4/0.5)
|  |  |  |  |  CLOUDAVGRT = '(289-325]' : 1 (1/0)
```

Runtime Analysis: Measures the time required to allocate tasks to cloud instances using the statistical collector.

Memory Utilization (%): Represents the percentage of memory consumed during the task scheduling process on cloud instances, using the statistical collector and load balancing method.

CPU Utilization (%): Indicates the percentage of CPU usage by the server when scheduling tasks to cloud instances through the statistical collector and load balancing technique.

Reliability Analysis (%): Assesses the reliability of the cloud server in successfully scheduling tasks to cloud instances using the statistical collector and load balancing method.

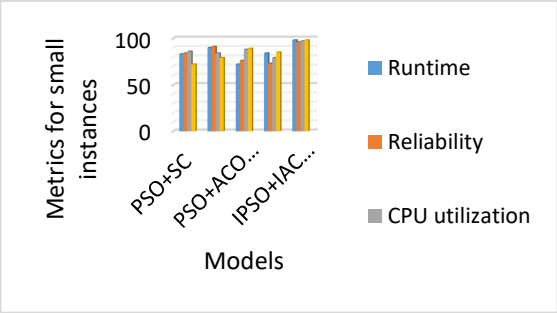

Figure 3. Comparative analysis of proposed model to the conventional models for multi-task multi-server load balancing using small instances.

Figure 3. represents the average runtime, reliability, CPU utilization and memory utilization for small instances.

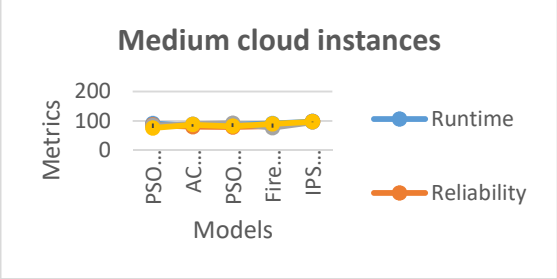

Figure 4. Comparative analysis of proposed model to the conventional models for multi-task multi-server load balancing using medium instances.

Figure 4. represents the average runtime, reliability, CPU utilization and memory utilization for medium instances. As represented in the figures, proposed model has greater efficiency than the conventional approaches on different cloud metrics.

Table 4. Comparative analysis of proposed model to the conventional models for multi-task multi-server load balancing using large instances.

Metrics	PSO+SC	ACO+SC	PSO+ACO+SC	Firefly+PSO	IPSO+IACO+SC
Runtime	76	83	82	80	96
Reliability	72	82	78	89	98
CPU Utilization	84	77	82	78	96
Memory Utilization	88	87	75	92	98

Table 4 represents the average runtime, reliability, CPU utilization and memory utilization for large instances. As represented in the table, proposed model has greater efficiency than the conventional approaches on different cloud metrics.

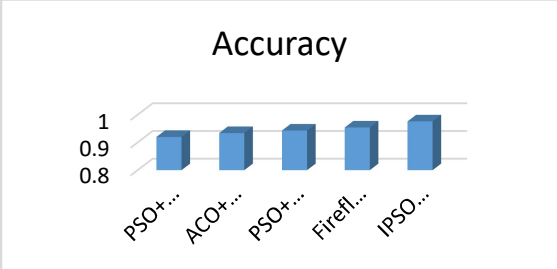

Figure 5. Classification Accuracy of Proposed and Conventional Models

Figure 5, presents the average accuracy of the present approach to the conventional models for load balancing decision patterns**.**

The experimental results demonstrate that the proposed IPSO+IACO+SC framework achieves superior performance compared to conventional approaches such as PSO+SC, ACO+SC, and Firefly+PSO across different cloud instance configurations. For example, in large cloud instances, the proposed model achieved 98% runtime efficiency and 98% memory utilization, whereas conventional methods showed comparatively lower performance. Similarly, the classification accuracy of the proposed framework reached 97.6%, indicating improved decision-making capability for multi-task scheduling. The enhanced performance is mainly due to the integration of hybrid meta-heuristic optimization with statistical resource collection, which enables balanced workload distribution and adaptive resource allocation. However, certain limitations were also observed during experimentation. In some scenarios involving extremely large and heterogeneous workloads, the optimization process required higher computational time due to repeated heuristic iterations. Additionally, performance variations were observed depending on the virtual machine configurations and workload arrival rates. These observations indicate that further optimization may be required for highly dynamic large-scale cloud environments. Future work can focus on reducing computational overhead and improving scalability using deep learning and adaptive predictive scheduling techniques.

The proposed hybrid IPSO-IACO framework demonstrates improved performance compared to conventional load balancing approaches in terms of runtime, reliability, CPU utilization, memory utilization, and classification accuracy. The integration of hybrid meta-heuristic optimization with statistical resource collection enables efficient task scheduling and adaptive resource allocation in dynamic cloud environments. These findings indicate that the proposed model can effectively support real-time multi-server multi-task scheduling applications such as cloud data centers, distributed computing systems, and large-scale service management platforms. The study advances existing research by combining intelligent optimization and classification techniques to improve Quality of Service (QoS) performance under heterogeneous workloads. However, slight computational overhead and performance variations were observed under highly dynamic large-scale workloads due to repeated optimization iterations. These observations suggest that further improvements can be achieved through adaptive deep learning-based scheduling and predictive resource management techniques in future work.

5. CONCLUSION

This paper introduces a novel classification model based on a multi-user, multi-server load balancing algorithm to enhance efficiency. Traditional models often struggle to accurately predict decision patterns across various cloud instances and typically rely on conventional classification algorithms for generating multi-task patterns with limited datasets. To address these issues, this work develops a hybrid ensemble load balancing framework aimed at improving decision pattern mining during the multi-server, multi-task scheduling process. The framework integrates hybrid Particle Swarm Optimization (PSO) and Ant Colony Optimization (ACO) into the ensemble load balancing approach within a real-time cloud computing environment. Additionally, a hybrid decision tree classifier is proposed to facilitate decision-making in the multi-task scheduling process. Experimental results demonstrate that the proposed model significantly outperforms conventional models in terms of multi-task, multi-server scheduling in a real-time cloud computing environment.

REFERENCES

[1] L. Zhang, Q. Deng, R. Lin, G. Gong, and W. Han, "A combinatorial evolutionary algorithm for unrelated parallel machine scheduling problem with sequence and machine-dependent setup times, limited worker resources and learning effect," *Expert Systems with Applications*, vol. 175, Aug. 2021, Art. no. 114843.

[2] J. L. Cremer and G. Strbac, "A machine-learning based probabilistic perspective on dynamic security assessment," *International Journal of Electrical Power & Energy Systems*, vol. 128, Jun. 2021, Art. no. 106571.

[3] X. Li and R. Yao, "A machine-learning-based approach to predict residential annual space heating and cooling loads considering occupant behaviour," *Energy*, vol. 212, Dec. 2020, Art. no. 118676.

[4] S. Ikeda and T. Nagai, "A novel optimization method combining metaheuristics and machine learning for daily optimal operations in building energy and storage systems," *Applied Energy*, vol. 289, May 2021, Art. no. 116716.

[5] M. Farhoumandi, Q. Zhou, and M. Shahidehpour, "A review of machine learning applications in IoT-integrated modern power systems," *The Electricity Journal*, vol. 34, no. 1, Jan. 2021, Art. no. 106879.

[6] B. Sabeena, S. Sivakumari, and P. Amudha, "A technical survey on various machine learning approaches for Parkinson's disease classification," *Materials Today: Proceedings*, Nov. 2020.

[7] A. Bellahsen and H. Dagdougui, "Aggregated short-term load forecasting for heterogeneous buildings using machine learning with peak estimation," *Energy and Buildings*, vol. 237, Apr. 2021, Art. no. 110742.

[8] J. H. Woo, B. Kim, S. Ju, and Y. I. Cho, "Automation of load balancing for Gantt planning using reinforcement learning," *Engineering Applications of Artificial Intelligence*, vol. 101, May 2021, Art. no. 104226.

[9] Y.-R. Shiue, G.-R. You, C.-T. Su, and H. Chen, "Balancing accuracy and diversity in ensemble learning using a two-phase artificial bee colony approach," *Applied Soft Computing*, vol. 105, Jul. 2021, Art. no. 107212.

[10] Z. Tan et al., "Combined electricity-heat-cooling-gas load forecasting model for integrated energy system based on multi-task learning and least square support vector machine," *Journal of Cleaner Production*, vol. 248, Mar. 2020, Art. no. 119252.

[11] H. ur Rehman, T. Korvola, R. Abdurafikov, T. Laakko, A. Hasan, and F. Reda, "Data analysis of a monitored building using machine learning and optimization of integrated photovoltaic panel, battery and electric vehicles in a Central European climatic condition," *Energy Conversion and Management*, vol. 221, Oct. 2020, Art. no. 113206.

[12] S. Thomas, H. Palahnuk, H. Amini, and I. Akseli, "Data-smart machine learning methods for predicting composition-dependent Young's modulus of pharmaceutical compacts," *International Journal of Pharmaceutics*, vol. 592, Jan. 2021, Art. no. 120049.

[13] Z. Tong, X. Deng, H. Chen, and J. Mei, "DDMTS: A novel dynamic load balancing scheduling scheme under SLA constraints in cloud computing," *Journal of Parallel and Distributed Computing*, vol. 149, Mar. 2021, pp. 138–148.

[14] N.-T. Ngo, "Early predicting cooling loads for energy-efficient design in office buildings by machine learning," *Energy and Buildings*, vol. 182, Jan. 2019, pp. 264–273.

[15] S. Mashhadi Moghaddam, M. O'Sullivan, C. Walker, S. Fotuhi Piraghaj, and C. P. Unsworth, "Embedding individualized machine learning prediction models for energy efficient VM consolidation within Cloud data centers," *Future Generation Computer Systems*, vol. 106, May 2020, pp. 221–233.

[16] H. Zeineddine, U. Braendle, and A. Farah, "Enhancing prediction of student success: Automated machine learning approach," *Computers & Electrical Engineering*, vol. 89, Jan. 2021, Art. no. 106903.

[17] L. Gan, X. Zhao, H. Wu, and Z. Zhong, "Estimation of remaining fatigue life under two-step loading based on kernel-extreme learning machine," *International Journal of Fatigue*, vol. 148, Jul. 2021, Art. no. 106190.

[18] S. Mangalathu, H. Shin, E. Choi, and J.-S. Jeon, "Explainable machine learning models for punching shear strength estimation of flat slabs without transverse reinforcement," *Journal of Building Engineering*, vol. 39, Jul. 2021, Art. no. 102300.

[19] E. U. Haq, X. Lyu, Y. Jia, M. Hua, and F. Ahmad, "Forecasting household electric appliances consumption and peak demand based on hybrid machine learning approach," *Energy Reports*, vol. 6, Dec. 2020, pp. 1099–1105.

[20] O. Avalos, "GSA for machine learning problems: A comprehensive overview," *Applied Mathematical Modelling*, vol. 92, Apr. 2021, pp. 261–280.

[21] U. K. Jena, P. K. Das, and M. R. Kabat, "Hybridization of meta-heuristic algorithm for load balancing in cloud computing environment," *Journal of King Saud University – Computer and Information Sciences*, Feb. 2020.

[22] C. Liu, L. Zhang, J. Niu, R. Yao, and C. Wu, "Intelligent prognostics of machining tools based on adaptive variational mode decomposition and deep learning method with attention mechanism," *Neurocomputing*, vol. 417, Dec. 2020, pp. 239–254.

[23] A. Dineva, B. Csomós, S. Kocsis Sz., and I. Vajda, "Investigation of the performance of direct forecasting strategy using machine learning in State-of-Charge prediction of Li-ion batteries exposed to dynamic loads," *Journal of Energy Storage*, vol. 36, Apr. 2021, Art. no. 102351.

[24] I. Ahmed et al., "Laser induced breakdown spectroscopy with machine learning reveals lithium-induced electrolyte imbalance in the kidneys," *Journal of Pharmaceutical and Biomedical Analysis*, vol. 194, Feb. 2021, Art. no. 113805.

[25] V. Srivastava and R. S. Pandey, "Machine intelligence approach: To solve load balancing problem with high quality of service performance for multi-controller-based Software Defined Network," *Sustainable Computing: Informatics and Systems*, vol. 30, Jun. 2021, Art. no. 100511.

[26] R. Kaur et al., "Machine learning and price-based load scheduling for an optimal IoT control in the smart and frugal home," *Energy and AI*, vol. 3, Mar. 2021, Art. no. 100042.