


CA-MLBS: content-aware machine learning based load balancing scheduler in the cloud environment

Muhammad Adil¹ | Said Nabi¹ | Muhammad Aleem²  | Vicente Garcia Diaz³  | Jerry Chun-Wei Lin⁴ 

¹Department of Computer Science and Information Technology, Virtual University of Pakistan, Lahore, Pakistan

²Department of Computer Science National University of Computer & Emerging Sciences, Islamabad, Pakistan

³University of Oviedo, Oviedo, Spain

⁴Western Norway University of Applied Sciences, Bergen, Norway

Correspondence

Jerry Chun-Wei Lin, Western Norway University of Applied Sciences, Bergen, Norway.

Email: jerrylin@ieee.org

Abstract

Cloud computing is the on-demand provision of computing resources over the Internet, such as cloud storage, computing power, network, and so on. Cloud computing has several advantages, including high speed, cost reduction, data security, and scalability. The main challenge in cloud environment is to balance the workloads and network traffic among the available resources to achieve maximum performance. Several methods have been proposed in the literature for effective load balancing, including heuristic, meta-heuristic, and hybrid algorithms. The performance of these techniques has been improved by combining machine learning based Artificial Intelligence (AI) techniques and meta-heuristic algorithms. Most of the existing load balancing techniques are not aware of the content type of user tasks. However, from the literature, the content type of the tasks can be very effective to design a balanced workload distribution system in the cloud. In this work, a novel AI-assisted hybrid approach called *Content-aware Machine Learning based Load Balancing Scheduler* (CA-MLBS) is proposed. The scheduling system CA-MLBS combines machine learning and meta-heuristic algorithms to perform classification based on file type. To achieve this, a Support Vector Machine (SVM) based classifier is used to classify user tasks into different content types such as video, audio, image, and text. A meta-heuristic algorithm based on Particle Swarm Optimization (PSO) is used to map users' tasks in the cloud. The proposed approach was implemented and evaluated using a renowned Cloudsim simulation kit and compared with Ant Colony Optimization File Type Format (ACOFTF) and Data Files Type Formatting (DFTF) heuristics. The results of the proposed study show that the proposed CA-MLBS technique achieved improvements of up to 29%, 29%, and 44% in terms of makespan, response time, and throughput, respectively.

KEYWORDS

cloud, content-aware, machine learning, PSO scheduler, task scheduling

This is an open access article under the terms of the [Creative Commons Attribution](https://creativecommons.org/licenses/by/4.0/) License, which permits use, distribution and reproduction in any medium, provided the original work is properly cited.

© 2022 The Authors. *Expert Systems* published by John Wiley & Sons Ltd.

1 | INTRODUCTION

Cloud computing is the provision of computing resources over the Internet. These resources include data storage, databases, software, networks, and servers. There are three types of cloud computing services: *Software-as-a-Service* (SaaS) (Cusumano, 2010), *Infrastructure-as-a-Service* (IaaS) (Serrano et al., 2015), and *Platform-as-a-Service* (PaaS) (Boniface et al., 2010). Software as a service is a way of delivering applications as service over the internet. Where you can access an application on the internet without any installation, maintenance, and infrastructure cost. Infrastructure-as-a-Service (IaaS) is way to delivering infrastructure such as dedicated servers, network, storage, and virtual machines over the internet. In IaaS, application, operating system and other application runtime requirements are managed by user. Platform-as-a-Service model provides infrastructure, runtime, middleware, and OS. However, applications and data are managed by user.

The most popular public cloud providers are Amazon Web Services (Amazon EC, 2015), Microsoft Azure (Chappell, 2010), Google Cloud Platform (Krishnan & Gonzalez, 2015), and IBM Cloud (Boniface et al., 2010). Cloud computing offers several benefits, such as more reliable infrastructure, agility in business processes, ease of collaboration between teams, and cost savings. In addition, the cloud makes data available from anywhere and on any device on a scalable and secure platform (Sunyaev, 2020).

Cloud computing includes hosts (i.e., off-site physical servers) and virtual machines (virtual environment of computing resources such as CPU, RAM, storage, etc). Usually, a host computer contains multiple virtual machines (VM). In the cloud computing environment, the process of effectively distributing the workload across multiple VMs is referred to as *load balancing*. The load balancing aspect is important to achieve scalable performance (Mishra et al., 2020). The benefits of cloud load balancing is twofold. On one side, the cloud user wants to execute their tasks in lesser time with minimized overall cost which lead to higher user satisfaction. On the other side, the cloud service provider needs minimized execution cost and high utilization of cloud resource which lead to high Return on Investment (ROI). To attain higher user's satisfaction and improved resource utilization, the workload need to distributed in balanced way (Nabi, Ahmad, et al., 2022; Nabi & Ahmed, 2021a; Nabi, Aleem, et al., 2022). Efficient utilization of cloud resource maximize the profit of cloud service providers and reduce energy consumption. As a system's computing needs increase, you can scale your infrastructure by adding more hosts and VMs. However, it is difficult to estimate the size and nature of the workload and determine the exact number of resources required (with their computing power) to be created at runtime. In this regard, AI-powered algorithms are more helpful to predict the workload, the type of workload, and the resource requirements with their computing capability.

To achieve maximum load balancing, task scheduling algorithms play an important role. Cloud task scheduling can be classified into two broad categories, that is, *heuristic* and *metaheuristic* algorithms. The heuristics-based algorithms such as round-robin, max-min (Ibrahim, Nabi, Baz, Alhakami, et al., 2020), and min-min (Ibrahim, Nabi, Baz, Naveed, & Alhakami, 2020) are mostly problem-dependent. The Round-robin is the simplest model of load balancing. It forwards a client request to each VM in turn and is easy to implement and has low scheduling overhead. The Max-min load balancing model allocates larger tasks with the highest priority and smaller tasks with the lowest priority. This algorithm favours larger tasks and penalizes smaller tasks. Whereas the Min-Min algorithm calculates the completion time of the tasks and allocates tasks based on their minimum execution time. These algorithms are fast and suitable for short-term solutions related to scheduling. Several heuristic approaches have been presented in the literature, such as the Zero Imbalance Mechanism (Kong et al., 2020), which is based on using the transfer time of tasks to the network. Dynamic Resource Aware Load Balancing Approach (DRALBA) (Nabi et al., 2021), that allocates a group of independent and compute intensive tasks to available virtual machines in a balanced way. DRALBA map tasks on VMs based on computation share of VMs and dynamically updates the VM computing share. Another heuristic-based load balancing method (Adhikari & Amgoth, 2018), which uses task size as one of the main considerations. Although the algorithms based on heuristics optimize the makespan and the *quality of service* (QoS) metric, however, the heuristics-based algorithms have limitations, such as the inability to find an optimal solution (Kaur & Kaur, 2022) for load balancing problems with conflicting parameters such as execution time and execution cost.

Methods based on meta-heuristics do not depend on a particular problem. They are generic algorithms that can be applied to a large number of related problems. Meta-heuristic algorithms address the limitations of heuristic algorithms and provide optimal results for scheduling problems. Several meta-heuristic algorithms have been presented in the literature, such as the deadline and resource constrained particle swarm optimization algorithm (PSORDAL) (Nabi & Ahmed, 2021a), Simulated Annealing (Hanine & Benlahmar, 2020), the ant colony optimization based dynamic and elastic algorithm (D-ACOELB) (Naik, 2020), and another swarm based meta-heuristic technique (Megharaj & Kabadi, 2019). Each meta-heuristic algorithm has its strengths and limitations. Therefore, combining two or more algorithms could complement the advantages and lead to more suitable solutions.

The concept of combining multiple algorithms to solve certain problems, such as scheduling, is called a hybrid scheduling algorithm. These algorithms can be a combination of (1) heuristic and meta-heuristic algorithms, (2) two meta-heuristic algorithms, and (3) AI-powered machine learning and meta-heuristic algorithms. As compared to meta-heuristic algorithms, the hybrid meta-heuristic algorithms use strengths of multiple algorithms to solve a problem. Several hybrid algorithms for load balancing optimization in the cloud environment are presented in the literature, such as a combination of Firefly and genetic algorithms (Rajagopalan et al., 2020) and a hybrid of Cuckoo and Firefly (Kumar et al., 2020). In addition, several literature reviews (Abrol et al., 2020; Pradhan et al., 2021) show the use of various meta-heuristic based algorithms to optimize load balancing in the cloud computing environment.

1.1 | Motivation

The load balancing in the cloud environment makes the system scale-able, improves reliability, increases performance, and boosts cost-effectiveness. The literature survey shows that most modern approaches to task scheduling consider parameters such as task length, task priority, and task file size to evaluate their proposed task scheduling schemes. However, most of these scheduling schemes do not consider the content type of the input workload. Moreover, the literature study shows that the content type of the workload can play a crucial role in balancing the workload (Cervantes et al., 2020).

The adaptation of cloud computing has grown rapidly along with other core services such as cloud storage. Various studies presented in the literature have applied machine learning algorithms (ML) to address the growing data challenges. There are three common types of ML techniques: *Unsupervised Learning* (Celebi & Aydin, 2016), *Supervised learning* (Liu, 2011), and *Reinforcement Learning* (Mirjalili et al., 2020). Supervised learning is a methodology of machine learning where a machine learning algorithm builds its intelligence based on input data and classified for a specific kind of output. Whereas, unsupervised learning is another methodology of machine learning where an algorithm identify patterns in data points of dataset that not classified. Reinforcement learning is a methodology of machine learning that takes suitable actions to maximize output in a specific scenario. To classify data based on certain features: ML provides classification algorithms, which are mostly part of supervised learning (Reddy & Varma, 2020) techniques. The classification algorithms analyse the available data to extract features and build the corresponding training model. Then, the training model is used with a test dataset to classify the data (Sahli, 2020; Sen et al., 2020).

Some hybrid techniques of AI-based machine learning algorithms and meta-heuristics are discussed. These hybrid algorithms show significant improvements (Pinho et al., 2020; Rabbani et al., 2020) and demonstrate that machine learning with meta-heuristic algorithms can significantly improve the solutions to various problems. Existing classification methods (Liu, 2011; Mirjalili et al., 2020; Pinho et al., 2020; Rabbani et al., 2020) also use PostgreSQL and Amazon Web Services (AWS) (AWSACC Services, n.d.) to develop techniques to solve the content classification problem. PostgreSQL is an open source and highly expandable database management system that supports both relational and non-relational queries. As compared to other database management systems, PostgreSQL supports wide range of data types such as geometric, network address, JSON, XML, HSTORE, arrays, ranges, and composite. However, these platforms themselves do not provide content classification mechanisms such as text-, image-, audio-, and video-based resource management.

Support Vector Machine (SVM) is considered a preferred choice for input workload classification based on its content (Cervantes et al., 2020). The existing content-based workload distribution optimization studies in the cloud environment such as Ant Colony Optimization File Type Format (ACOFTF) (Junaid, Sohail, Ahmed, et al., 2020) and Data Files Type Formatting (DFTF) (Junaid, Sohail, Rais, et al., 2020) can be improved by a refined dataset, improved kernel method, and simpler hybrid meta-heuristic algorithms.

The SVM classification algorithm as compared to other ML algorithms such as Artificial Neural Network (ANN) has the following advantages: (1) shorter training time, (2) better capacity to converge, and (3) better interpret-ability. Based on all the above, there is a need for a content-aware load balancing model using machine learning classification. That can classify the cloud data into appropriate categories and the classified tasks can be scheduled to the best-suited type of VMs using the load balancing scheduler. In the next section, we present a Content-Aware Load Balancing model to overcome the limitations of existing approaches.

1.2 | Major contributions

This paper presents an AI-enabled meta-heuristic based hybrid and content-based method that uses a combination of machine learning and particle swarm optimization techniques to improve load balancing in cloud computing. The proposed *Content-Aware Machine-Learning Based Load Balancer* (CA-MLBS) classifies users' tasks based on content type. For task classification, CA-MLBS uses SVM classification algorithm. SVM is one of the most popular classification methods for cloud tasks (Cervantes et al., 2020). Moreover, SVM is a robust supervised learning algorithm that can be used to classify data and regression assignments. The proposed CA-MLBS technique uses the particle swarm optimization (PSO) algorithm to assign the classified tasks to the appropriate group of VMs. The reason for using the PSO algorithm for load balancing is that PSO-based algorithms can improve load balancing with optimal memory utilization and in a fast manner. The literature study shows that PSO based tasks scheduling and optimization algorithms have better performance and have simple implementation as compared to other optimization algorithms like Genetic Algorithm(GA), Ant Colony Optimization (ACO) among other (Fadlallah et al., 2021; Nabi & Ahmed, 2021a; Nabi, Aleem, et al., 2022). The proposed scheduling scheme performs user cloud task classification based on the type of user tasks (video, audio, image, and text). Moreover, CA-MLBS uses file fragments to build its training model and assign classification labels. For each file type, there is a corresponding set of VMs with their respective configurations. The classified tasks were distributed to appropriate VMs using PSO to effectively distribute the workload. The multi-objective model can be used for load balancing, however, multi-objective model is more useful for optimizing multiple objectives with conflicting parameters where attaining one objective (i.e., performance improvement of a certain parameter) may result in degraded performance of the other aspect, for example, time, cost, and so on. Moreover, the proposed research evaluates multiple non-conflicting parameters like

makespan, throughput, and response time (Nabi & Ahmed, 2021a; Nabi, Aleem, et al., 2022). In future, multi-objective model will be used for optimizing conflicting parameters like time and cost. The main contributions of this work are summarized below:

- This research proposes a content-aware, machine learning-based load balancer (CA-MLBS) scheme based on a hybrid load balancing model. The CA-MLBS uses *File Fragment Type* (FFT) (Mittal et al., 2019) dataset based on file fragments of different content types to improve classification results and workload distribution;
- The proposed algorithm is an easily applicable and simplified approach compared to the existing content-type algorithms;
- The proposed scheme classifies users' cloud tasks based on content types such as video, audio, image, and text;
- The proposed algorithm uses a PSO-based scheduling mechanism to optimize workload distribution;
- Extensive analysis and comparison of response time, throughput and makespan have been performed.

The remainder of the article is organized as follows: Section 2 presents the literature review. Section 3 contains the system architecture, system model, RADL algorithm, complexity analysis, and scheduling overhead. Section 4 presents the experimental setup, dataset configurations, performance comparison, and evaluation results. Section 5 provides the conclusion and a roadmap for the future.

2 | RELATED WORK

This section discusses the relevant literature by highlighting pros and cons of state-of-the-art and has been summarized in the Table 1. Resource-Aware Load-Balancing Algorithm (RALBA) (Hussain et al., 2018) is a heuristic load-balancing algorithm that performs workload distribution according to the processing capacity of VMs. It applies the load balancing algorithm in the following two phases: (1) identifying the processing requirements of the task and (2) identifying the processing capacity of the available VMs. Based on the task's requirements, RALBA allocates to

TABLE 1 Summary of load balancing models in cloud computing

References	Approach	Pros	Cons	CA
Nabi et al. (2021)	Heuristic	Reduced response time and resource utilization	No makespan and classification	×
(Nabi and Ahmed (2021b)	Heuristic	Improved resource utilization and makespan	lacks tasks' classification, sequence-based support and response time	×
Muthusamy and Chandran (2021)	Heuristic	Reduced makespan and better execution time	Lacks degree of Imbalance	×
Nabi and Ahmed (2021a)	Meta-heuristic	Improves makespan, resource consumption, and execution cost	Lacks response time and tasks classification	×
Semmoud et al. (2020)	Meta-heuristic	Reduced makespan and better response time	throughput of imbalance can be improved	×
Agarwal et al. (2020)	Meta-heuristic	Reduced Makespan	Lacks response time, no pseudo implementation	×
Muthsamy and Ravi (2020)	Meta-heuristic	Reduced makespan and better execution time	Experiments results may vary with state of the arts dataset	×
Lilhore et al. (2020)	Hybrid	Improved Resource utilization and Makespan	Limited Test Environment, Lacks proper dataset, No task classification	×
Mishra and Majhi (2021)	Hybrid	Improved response time and Makespan	lacks sophisticated testing environment, state of the arts dataset, and categorization	×
Neelima and Reddy (2020)	Hybrid	Reduced Execution time and cost	larger dataset and testing environment	×
Junaid, Sohail, Rais, et al. (2020)	Hybrid	Better throughput, overhead Time, and energy consumption	Lack unified dataset, painless classification, and Kernel method needs optimization	✓
Rafieyan et al. (2020)	Hybrid	Reduced makespan, Less waiting time	Results can be improved refined test and dataset	×
Attiya et al. (2020)	Hybrid	Reduced makespan	No state of the arts dataset	×
Sharma and Garg, (2020)	Hybrid	Reduced Makespan, Better energy consumption	More QoS Metrics, Smaller dataset	×
Jena et al. (2022)	Hybrid	Reduced makespan and better Throughput	lacks state of the art dataset and adequate environment	×

the appropriate VM. The RALBA algorithm is based on two sub-schedulers (fill and split schedulers). RALBA provides improvements in makespan, resource utilization, and execution time. However, RALBA does not support task response time and content type classification based on the cloud's task type.

The Dynamic and Resource Aware Loading Balancing Algorithm (DRALBA) is a heuristic load balancing algorithm that balances loads based on processing performance and maintains VMs' workload. It assigns a set of independent tasks to a preconfigured group of VMs. It calculates the processing capacity of a group of VMs for a set of independent tasks. Based on these calculations, DRALBA selects the most appropriate VM with the highest processing capacity. DRALBA provides improvements in resource allocation and response time. However, it lacks a mechanism for distributing the workload based on content type classification. Overall Gain-based Resource-aware Dynamic Load-balancer (OG-RADL) (Nabi & Ahmed, 2021b) is a heuristic algorithm for dynamic load balancing. The OG-RADL algorithm improves workload distribution with enhanced resource utilization, better task deadline management, and delivers improved load to enhance overall cloud performance. OG-RADL presents a novel technique to normalize the values of evaluation parameters such as average resource utilization ratio, response time, makespan, and task deadline. Moreover, the OG-RADL load balancing technique provides improvements in the overall gain of the cloud. However, the proposed algorithm requires further improvements for task sequence and content-based workload distribution.

Muthusamy and Chandran (2021) have proposed a cluster-based task scheduling framework (CBTS) that uses K-Means clustering and considers task length and VM limit. In CBTS, tasks are distributed based on task length and VMs are clustered based on computational power. The single task in each cluster is assigned to the appropriate VM in the VM group of that cluster. This technique has shown improvements in execution time and makespan. However, the CBTS technique does not support classification of tasks based on their content.

Particle swarm optimization based resource and deadline aware dynamic load balancers (PSO-RDAL) (Nabi & Ahmed, 2021a) have been proposed. The PSO-RDAL is a meta-heuristic load balancing algorithm that delivers lower processing cost and time for large-scale independent cloud tasks. This research work evaluates the PSO-RDAL mechanism using multiple performance aspects such as resource consumption, makespan, task deadline compliance, response-time, penalty-cost, and execution-cost. The PSO-RDAL does not perform classification by task content type.

Semmoud et al. (2020) propose a Starvation Threshold-based algorithm for scheduling problems. Each VM maintains its workload state and performs load balancing without considering the state of the other VMs. Experiments are conducted to evaluate the timeout and response time based on quality of service (QoS) metrics. The performance evaluation of the proposed method was performed with up to 800 tasks on 100 VMs. The proposed study showed improvements in response time and makespan. However, the proposed technique was tested with a smaller dataset. Moreover, this technique does not show content-based classification support for cloud tasks. The mutation-based PSO (Agarwal et al., 2020) updates the fitness function of each particle until the maximum iteration is reached. The makespan QoS performance metric is used to analyse the experiment results. For the proposed method, 20 data centers and up to 200 tasks were used for simulation. The proposed method achieves an improvement in the makespan. However, the pseudo-algorithm was not properly explained. Moreover, this technique does not support the classification of tasks based on the content type. Muthusamy and Ravi (2020) have presented a load-balancing technique based on an Artificial Bee Algorithm (ABC). The ABC-based scheduling algorithm performs optimal search based on the honey bee's approach to find the best from the available sources. The presented study has shown improvements in makespan and execution time. However, the dataset used for the evaluation is primitive and the technique is not capable to deal with the content-aware Cloud tasks scheduling. A hybrid approach of PSO and Firefly Algorithm (FA) optimization was proposed (Lilhore et al., 2020), which assigns the shortest job to the fastest processor-based machine and applies the shortest job next approach to PSO. The proposed technique considers the makespan and task migration as core scheduling objectives. However, the experiments were conducted in a limited environment and lack a comprehensive evaluation with state-of-the-art datasets. It also does not support classification of tasks by task type.

Another load balancing approach is proposed (Mishra & Majhi, 2021) and is inspired by the improved PSO and Honey-Bee Optimization (HBO) algorithms. The proposed technique distributes the workload among the VMs while the birds search for the food sources. The proposed method showed improvements in makespan, response time, and throughput. However, the experiments were conducted in a limited setting with fewer orders. The results could vary drastically if the experiments are conducted with large datasets. In addition, the proposed method lacks classification of tasks based on task types. An adaptive dragonfly algorithm (ADA) based on the dragonfly algorithm (DA) and the firefly algorithm (FA) was presented (Neelima & Reddy, 2020). The main goal of the proposed approach is to allocate the workload to the VM using the ADA. This limits the absolute execution time and cost. The proposed method shows improvements in execution time. However, the task dataset is smaller for the configured number of VMs. Moreover, this approach is not content-aware model.

Rafieyan et al. (2020) presented a ranking model called VIKOR, which is based on the best-worst method (BWM). The VIKOR algorithm acts as a manager to specify the task requirements. The identification of important decision criteria is intended to reflect the importance of cloud tasks in scheduling by specialists. The proposed approach shows improvements in makespan, throughput, and VM utilization. However, the experiments were conducted in a limited environment and lack content-based task classification. The Harris Hawks optimization algorithm (HHO) combined with a Simulated Annealing (SA) based model has been proposed (Attiya et al., 2020) for a balanced distribution of cloud tasks. The proposed method uses SA for local search with an improved HHO algorithm. The proposed method shows an improvement in the makespan. However, the experiments are conducted with a relatively small dataset. Moreover, the classification of tasks by task type is missing.

Sharma and Garg (2020) presented a hybrid meta-heuristic approach, which is called Harmony-Inspired Genetic Algorithm (HIGA). The HIGA technique uses the exploration and exploitation features of genetic and harmony search algorithms, respectively. The HIGA identifies the local and global optimum and provides a fast combination. The proposed technique has shown improvement in makespan and energy consumption. However, the experiments were conducted in a limited environment with a limited dataset. Moreover, the tasks in the proposed study are not classified by task types. Data Files Type Formatting (DFTF), which is based on Cat Swarm Optimization (CSO) and SVM (Junaid, Sohail, Rais, et al., 2020). The DFTF classifies the cloud tasks into different types, that is, text, images, video, and audio with SVM using a polynomial kernel. The classified tasks are input to CSO to perform load balancing. However, DFTF does not use a state-of-the-art dataset. In addition, DFTF classifies videos and audio into subcategories. The Radial Basis Function (RBF) kernel provides higher accuracy for the fine-tuned dataset compared to the polynomial kernel.

The QMPSO is a hybrid approach of modified Q-learning and particle swarm optimization (Jena et al., 2022). Three objective functions were formulated, which include (1) the workload difference between hosts and the average load on the cloud network, (2) the total energy consumption, and (3) the number of submitted tasks. The proposed technique considers the evaluation parameters such as throughput, energy usage, and makespan. However, it lacks state-of-the-art, refined experimental environment, and content-based classification. The proposed task scheduling technique is a mixture of Firefly and PSO techniques (FIMPSON) (Devaraj et al., 2020). The FIMPSON algorithm improves load balancing based on the resource utilization of the cloud tasks. The proposed model showed improvements in makespan and throughput. However, the experiments were not performed with the state-of-the-art simulation environment. The analysis of the results may be different with a proper test environment. The proposed model categorizes tasks based on their size, but it lacks classification of tasks based on task types.

The study proposes the Honey Bee behaviour-based load balancing method, which attempts to minimize load redundancy by assigning tasks to matching or suitable VMs. After task assignment, it computes the state of VMs. The proposed method showed improvements in the following QoS performance matrices: makespan and degree of load balancing. However, the proposed technique does not show such improvements in response time. Moreover, the proposed study lacks the classification approach to categorize the tasks by content type. The summary of the literature review can be found in Table 1.

3 | PROPOSED CA-MLBS FRAMEWORK

In the cloud computing environment, balancing the workload across VMs is a challenging task. Balanced workload distribution helps to achieve optimal utilization of cloud computing resources. Optimal load balancing also improves cloud task execution and response time. Load balancing methods use various task scheduling algorithms based on cloud task characteristics such as task length, file size, and task content type to calculate the available workload of cloud resources. An optimal load balancing method receives incoming cloud task requests, intercepts their processing needs, estimates the existing workload, and selects the best VM to process the request based on this information.

Our proposed cloud load balancing model called CA-MLBS is based on a hybrid technique that combines PSO and SVM machine learning algorithms. The detailed system-architecture of our proposed hybrid load balancing model is shown in Figure 1. As shown in Figure 1, the cloud computing infrastructure has a lowest layer, the physical layer (Kaur, 2020), which includes hosts (a host is a physical dedicated server with processing, memory, data storage, and data transfer resources). The physical layer of cloud computing also includes cloud data storage resources.

The virtualization layer of cloud computing infrastructure is on the top of the physical layer that includes Virtual Machines (VMs). A VM uses the resources of a host machine and acts as a separate computer with all the essential software including the operating system, web server, and database applications, and so on. A host machine can have multiple virtual machines, and a virtual machine receives incoming cloud task requests, processes these tasks, and sends the execution responses back to the user.

The topmost layer is the load balancing layer that receives the user's cloud task, parses the cloud task parameters, and sends tasks to VMs. The load balancing layer is responsible for effective workload distribution among VMs. This is the layer where we introduce our proposed scheduling algorithm. Our proposed load balancing model classifies user tasks based on content type and selects the best VM to map a specific cloud task.

The working semantics of the proposed load balancing model is presented in Figure 2. The proposed model comprises two phases: (1) content classification phase, and (2) the load balancing phase as shown in Figure 2. The content classification phase performs the classification of user tasks into four categories based on their content type. These categories include video, audio, image, and text type. The task classification is performed using the Support Vector Machine (SVM) based machine learning technique. Before task classification, training of the SVM model is performed using the training dataset. For the training of the SVM model, Radial Basis Function (RBF) kernel method has been used.

SVM Kernel functions are a set of mathematical functions, that take input data and transform it into a specific form. SVM kernel functions are categorized as a linear, nonlinear, polynomial, radial basis function, and sigmoid. The radial basis function is the most adopted kernel function as it has a finite and localized response with the entire x -axis. The classified tasks have been stored in four different collections (video, audio, image, and text-based) and forwarded to the Load-balancing phase. The load balancing process begins with receiving incoming collections of classified cloud tasks and the classified tasks are distributed among the VMs in a balanced way.

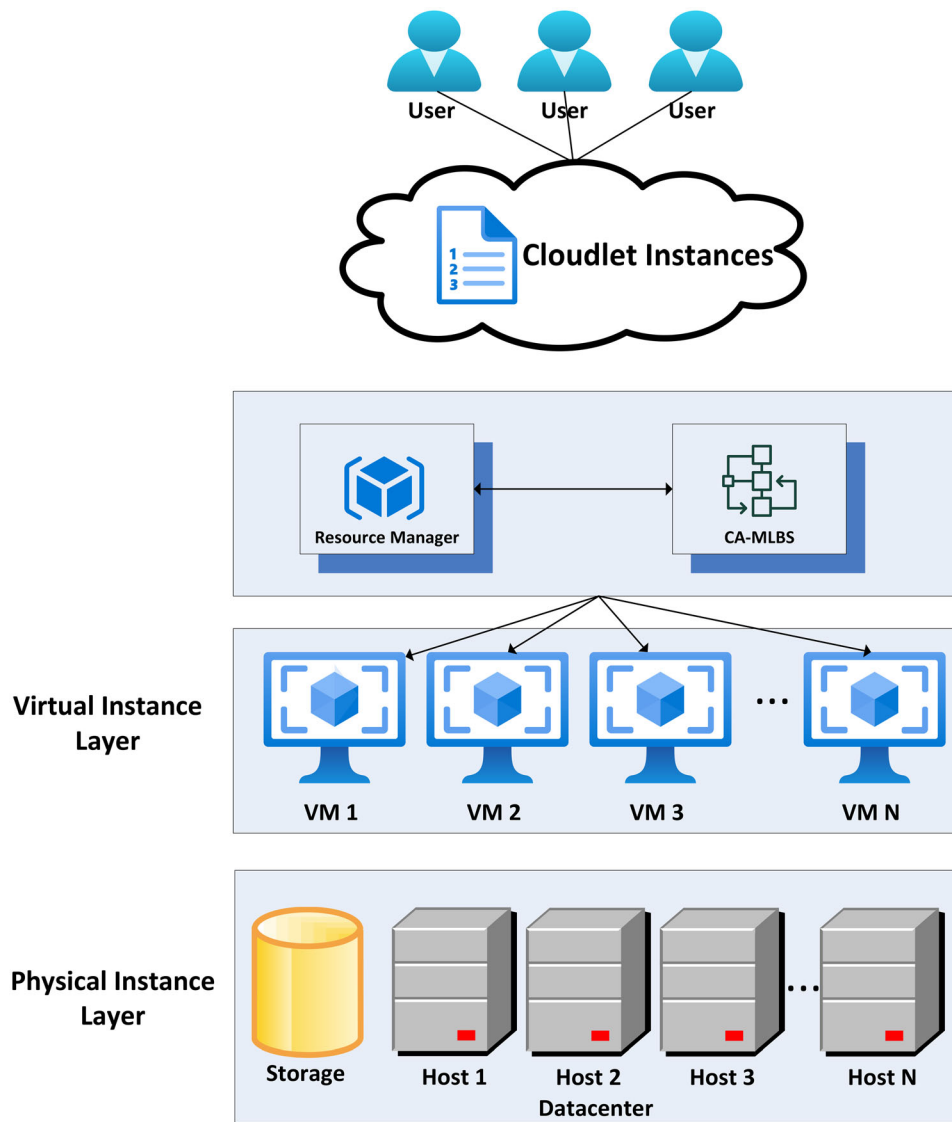


FIGURE 1 System architecture of CA-MLBS

The Particle Swarm-based Optimization (PSO) technique was used for load balancing. The reason is that PSO performs better than its counterparts (Nabi, Ahmad, et al., 2022; Nabi & Ahmed, 2021a). The PSO-based load balancing phase includes steps such as initializing the PSO scheduler along with its parameters, computing the particle position and velocity, evaluating the fitness function, updating all the particles, completeness criteria, and final mapping of the optimal solution. After receiving the final mapping from PSO, the tasks are executed according to the optimal mapping plan. When the execution of the tasks is completed, the value of the makespan, throughput, and response time of the tasks are calculated.

3.1 | CA-MLBS algorithm

The CA-MLBS for task scheduling starts by receiving incoming cloud task requests based on file fragments containing various file fragments of the task content, for example, text, image, audio, and video. The first phase is content type classification, which is performed using an SVM machine learning algorithm and prepares collections of classified tasks. The Key benefits of using SVM in the proposed model are that SVM is relatively memory efficient, more effective with high dimensional spaces, works well when there is a clear margin of separation between classes and is effective in cases where the number of dimensions is greater than the number of samples.

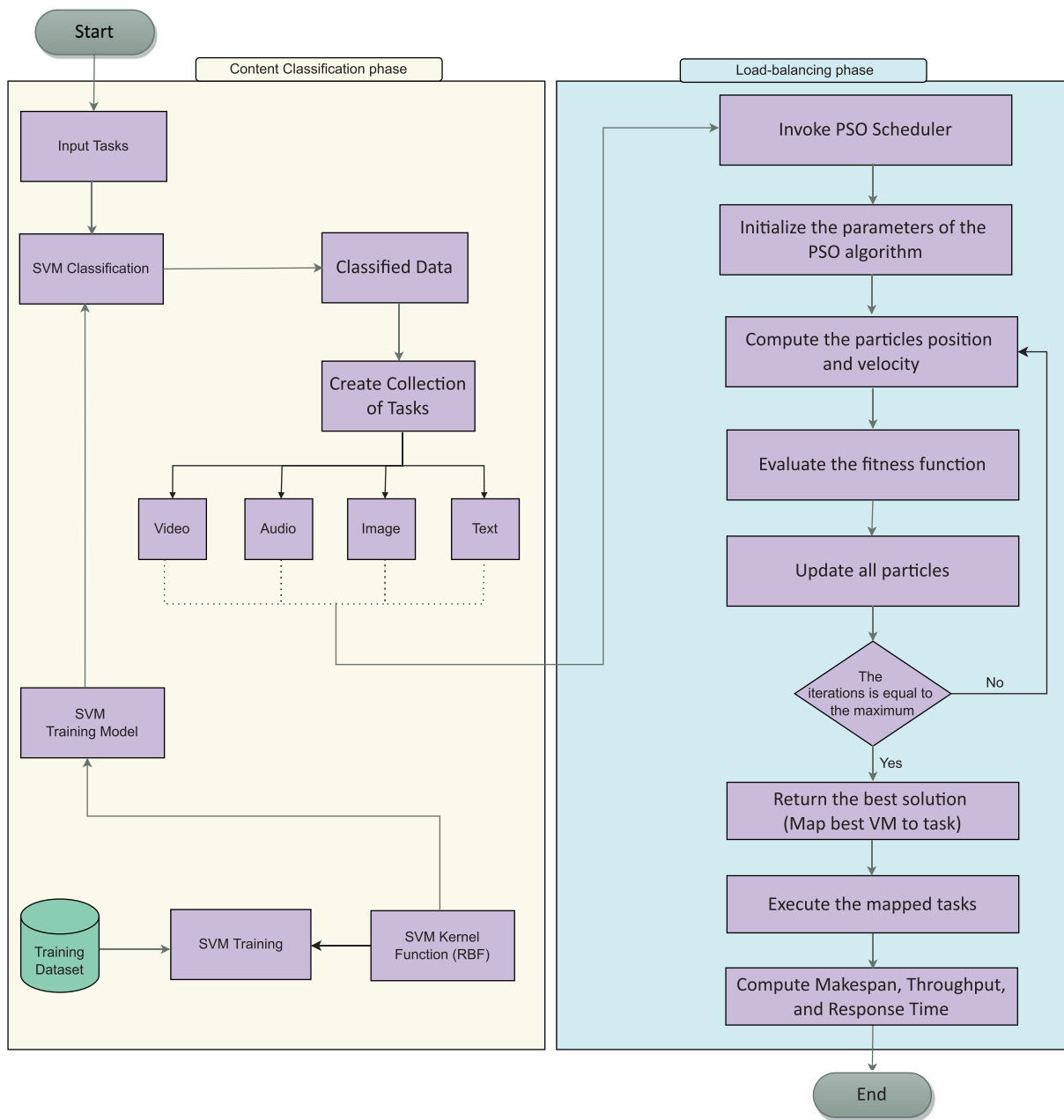


FIGURE 2 Working semantics of CA-MLBS

The content type classification process is based on the *Radial Basis Function* (RBF) kernel method using high-dimensional data (Algorithm 1, lines 1–8). Once the algorithm has collections of classified tasks, it proceeds with load balancing. The load balancing process takes two parameters: (1) first, the collections of classified tasks, and (2) the groups of VMs, where each group of VMs is dedicated to a particular content type. The load balancing process is based on the PSO scheduling algorithm explained earlier (Algorithm 1, lines 9–35).

The PSO-based scheduling algorithm specifies its parameters such as the number of iterations and the population size (Algorithm 1, lines 11–12). The PSO mechanism is used, which first initiates the particles with random velocity and position (Algorithm 1, lines 13–18). After initializing the PSO parameters with defined configurations, the fitness function evaluation is updated at each iteration to estimate the global and local best values (Algorithm 1, lines 19–32). The objective of fitness function is to find best VM for appropriate task in appropriate set of VM(s). Further, the PSO-based scheduling algorithm determines the best VM for each task and returns the scheduled tasks data. The evaluation process is repeated until the exit criterion is met. Moreover, the PSO-based scheduling algorithm determines the best VM for each task and returns the data of the scheduled tasks (Algorithm 1, line 33). The main elements of the proposed model were presented in the previous section.

Algorithm 1 Ca-MLBS

```

1: procedure CA-MLBS(tasks,vms)
2:   procedure ClassifyTasks(tasks)
3:     classifiedTasks: empty;
4:     for i ← 1... size(tasks) do
5:       classifiedTask ← predict(tasks[i]);
6:       classifiedTasks.add(classifiedTask);
7:     end for
8:     return classifiedTasks      ▷ return classified tasks;
9:   end procedure
10:  procedure PSOScheduler(classifiedTasks,vms)
11:    iterations ← 900;
12:    population ← 80;
13:    for p ← 1... population do      ▷ Initialize particles
14:      for t ← 1... size(classifiedTasks) do
15:        positionpt ← random(Pmin,Pmax);
16:        velocitypt ← random(Vmin,Vmax);
17:      end for
18:    end for
19:    while k ≤ iterations do
20:      for each particle p do
21:        f ← Calculate fitness based on task type and task length;
22:        if f is better than pbestpt from history then
23:          pbestpt ← f;
24:        end if
25:        gbestt ← best particle's fitness value;
26:      end for
27:      for each particle p do
28:        for each task t do
29:          vpt ← W * vpt + C * r() * (pbestpt - ppt) + C * r() * (gbestpt - ppt);
30:        end for
31:      end for
32:    end while
33:    return Best VMs for tasks;
34:  end procedure
35:  return Scheduled Tasks
36: end procedure

```

3.1.1 | Content-based classification step

CA-MLBS begins with a collection of file fragments of various content types such as videos, images, audios, and texts. The training dataset contains feature sets, class labels, and values. We used the training datasets and the SVM classification algorithm to build the training model. In machine learning based classification, the training model is a training source for the classification algorithms. It analyses and understands the training model to build its intelligence.

We build the training model using the features of kernel function of the SVM. The kernel function is a set of mathematical functions that process the input data and convert it into the required form. It converts the low-dimensional data format into a high-dimensional data format that can be further used for classification. There are various SVM kernel functions such as linear, non-linear, polynomial, radial basis function (RBF) and sigmoid. We used RBF (Majdisova & Skala, 2017) kernel function in our model because there are less constraints in terms of data formats, versatile applicability, high accuracy and fast convergence. The RBF kernel model is shown in Equation (1):

TABLE 2 Sample dataset

Task type	Task length	File size
AUDIO	3010	903
TEXT	750	225
IMAGE	1620	486
TEXT	810	243
VIDEO	4560	1368
AUDIO	3110	933
TEXT	990	297
VIDEO	7240	2172
VIDEO	7110	2133
IMAGE	1290	387
IMAGE	1460	438
VIDEO	7980	2394
IMAGE	1070	321
AUDIO	3020	906
VIDEO	7720	2316

$$f(x1, x2) = \exp(-\gamma * \|X1 - X2\|^2). \quad (1)$$

In the above equation, gamma (γ) represents the points around a single training point. $X1 - X2$ is the product of the features used. We use a testing dataset (as shown in Table 9), which is also a collection of file fragments associated with different content types, as the input source. The content-based classification phase receives incoming queries and uses the training model to perform analysis and predict the class of the cloud user's tasks.

The File Fragment Type (FFT)–75 DATASET (Mittal et al., 2019) contains random samples of file fragments from 75 of the most commonly used content types. It is one of the most distinct and most harmonious datasets according to our research. The FFT dataset consists of 512 and 4096 byte file fragment blocks and is labelled with classification IDs. This is a well-organized dataset for machine learning classification. The FFT dataset is in NumPy (Harris et al., 2020) format, an important machine learning library in Python. It supports a robust and powerful multidimensional array. The dataset is further processed using Scikit-learn, an open source machine learning library for Python. It supports major machine learning algorithms such as SVM, k-means, and random forests. Using Scikit-learn, the dataset was first filtered for text, image, audio and video content (as shown in Figure 8). It is also converted into the desired dataset format (Chang & Lin, 2011).

The content-based classification process performs classification of tasks into four classes: video, audio, image, and text. Content-based classification reduces the processing overhead for the load balancer by preprocessing learning, feature extraction, and classification. After classification, the classified tasks are converted to the Comma-Separated Values (CSV) format. CSV format is one of the most commonly used formats. The CSV format is in CloudSim ready format that can easily used as input for CloudSim modelling and it can easily be read in any programming language. Each line in the CSV file describes the content type (i.e., text, image, audio or picture), the length of the task in millions of instructions (MIs) and the file size in MIs. Table 2 shows the sample dataset.

3.1.2 | Load balancing step

The classified tasks in CSV format are then the input to the load balancing process. For each task category, there are four different groups of VMs with corresponding resource configurations (i.e., processing power, memory, and storage capacity). For example, a video content type task typically requires more processing, memory, and storage resources. We configured video-type VMs with 1000 MIPS (million instructions per second), 16 GB of memory, and 360 GB of data storage. Audio-type VMs were configured with 900 MIPS operations, 12 GB of memory, and 250 GB of storage; image-type VMs were configured with 700 MIPS and 200 GB of storage and 8 GB of memory; text-type VMs were configured with 500 MIPS, 4 GB of memory, and 120 GB of storage.

The load balancing step has groups of VMs and classified tasks as input. Each group of VMs is responsible for processing a specific type of task. We used a PSO-based scheduling algorithm, which is one of the most popular population-based optimization algorithms. The PSO concept is derived from social interaction to solve a problem. It has multiple agents (particles) that perform swarm movement around the search space and search for the best solution. After each iteration, the position and velocity of the particles are updated until the best solution is reached. PSO is one of the most robust, coherent, and manageable algorithms compared to other population-based algorithms. We used the CloudSim (Calheiros et al., 2011) simulation environment to evaluate our proposed model. Our implementation is based on JSwarm-PSO (Cingolani, 2011), a Java-based PSO framework (Cingolani, 2011).

The technologies used in the proposed models are followings. For Content Classification LibSVM (Chang & Lin, 2011), JSwarm-PSO (Cingolani, 2011) for PSO based load scheduling, and CloudSim (Calheiros et al., 2011) for the load balancer modelling and simulation.

Fitness evaluation

The fitness evaluation function is a special type of function that calculates the fitness value and indicates how close it is to achieving the specified goals. The fitness evaluation function serves as a guide to find the optimal solution. It is an important part of meta-heuristic algorithms. It facilitates the application of the required optimization to find the best solution. In load balancing models, it helps to find the best VM for a given task to achieve effective workload distribution. Our proposed model performs fitness evaluation at each iteration. It also performs the exit criteria to complete the processing of the algorithm. Our proposed model optimizes the response time, throughput, and makespan to improve the load balancing among VMs.

Makespan: To get the best performance from cloud computing resources, the makespan should be minimized. The term makespan is used for the finish time of all cloud tasks using the available cloud resources such as computing power, memory, and storage (Zheng et al., 2017). Makespan is represented in Equation (2).

$$\text{Makespan} = FT_{\text{Max}\{t,v\}} \quad t \in T, t = 1, \dots, n \quad l \in VM, v = 1, \dots, m. \quad (2)$$

In Equation (2), T_t represents number of tasks, VM_l represents number of VMs, and FT_{tv} represents finish time of t tasks with v VMs.

The processing time of a given type of task for a given type of VM needs to be estimated so that the load balancer can select the best VM from the group. We have presented the calculation for makespan in Equation (3).

$$ET_{\text{max}} = \sum_{t=1}^m \sum_{v=1}^n (t_{tv} \cdot X_{tv}). \quad (3)$$

FT_{max} is the maximum finish time for the t th task on the v th VM. The n is task count and m is VM count.

The total finish time of all the cloud tasks with the support of the objective function (Khorsand et al., 2019; Saeedi et al., 2020) have been computed using Equation (4).

$$F2(t) = \min \{ \text{Makespan}(t) \}. \quad (4)$$

Response time: The response time in cloud computing is the total time it takes to respond to a cloud task request (Phi et al., 2018). The response time of tasks has been computed using Equation (5).

$$\left[R_T = \sum_{t=1}^m P_{Tt} - (S_{Tt} - E_{Tt}) \right]. \quad (5)$$

In Equation (5), R_T represents the response time, P_{Tt} represents the t task's processing time, S_{Tt} presents t th task's start time, and E_{Tt} presents t th task's end time.

Throughput: Load balancing throughput in the cloud is the number of tasks processed in a given period of time (Ahmad & Khan, 2018). An efficient load balancing algorithm should maximize the completion of the number of tasks in a unit of time. We have represented the throughput in Equation (6).

$$J_{TP} = \sum_{h=1}^m \frac{J_h}{J_{th}}. \quad (6)$$

In Equation (6), J_{TP} represents the job throughput time, J_h represents h th job, and J_{th} the h -job's completion time.

4 | PERFORMANCE EVALUATION

4.1 | Experimental setup

The CloudSim (Calheiros et al., 2011) is used for verification and evaluation of proposed model. CloudSim is the most commonly used Java-based cloud computing simulation framework that supports the setup could computing resources such as hosts, VMs, Network pools, and storage capacities. It also helps to set up cloudlets (a cloudlet is a cloud task in the CloudSim environment), datacenter brokers, VM allocation policies, bandwidth, and RAM provisioning. It facilitates verifying and evaluating load balancing models cost-effectively. It allows users to concentrate on system modelling without gaining low-level details of cloud infrastructure and services. The advantages of CloudSim are it delivers an open-source, free, and customizable cloud simulation environment to assess the performance of the cloud computing models.

Further, it is an easy-to-use and robust cloud simulation software that allows testing a wide range of cloud applications in heterogeneous environments. Our proposed model has four groups of VM: Text-type VMs, Image-type VMs, Audio-type VMs, and Video-type VMs. Each group of VMs has distinct configurations of computing, memory, storage, and network resources. That is presented host configurations in Table 3. And, VM configurations for each group of VMs for the simulation environment presented in Table 7.

The state-of-art load balancing strategies that are used for comparative evaluation have evaluated for different parameter settings and parameters having the best results selected for comparison. Parameter settings used for proposed approach CA-MLBS, DFTF, and ACOFTF are presented in Tables 4–6, respectively.

TABLE 3 Host parameters

Hosts	PEs	Ram(GB)	Storage(TB)	Speed	Bandwidth
1–100	2	64	3	10,000	10,240
1–100	4	64	6	10,000	10,240
1–100	8	128	8	10,000	10,240

TABLE 4 Parameter configurations for CA-MLBS

Parameter	Value
Number of iterations	900
Population Size	80
Update particle's velocity	0.5
Particle's inertia	1
Acceleration coefficient	0.9

TABLE 5 Parameter configurations for DFTF

Parameter	Value
Cat size	100
Population size	500
SMP	5
CDC	80%
SRD	20%

TABLE 6 Parameter configurations for ACOFTF

Parameter	Value
Initial Pheromone	0.1
Alpha	3
Beta	1
RHO	12

TABLE 7 VMs parameters

VMs	PEs	Ram (GB)	Storage (TB)	Speed	Bandwidth
2-1000	1	4	128	1000	1024
2-1000	2	8	256	1000	1024
2-1000	2	16	512	1000	1024
2-1000	4	32	1024	1000	1024

TABLE 8 Dataset description

Category	Size in quantity
Text Files Dataset	16,000
Image Files Dataset	16,000
Audio Files Dataset	16,000
Video Files Dataset	16,000

TABLE 9 Training and test datasets statistics (Adil et al., 2022)

Samples	Size in quantity
Training Text File Fragments	15,000
Testing Text File Fragments	1000
Training Image File Fragments	15,000
Testing Image File Fragments	1000
Training Audio File Fragments	15,000
Testing Audio File Fragments	1000
Training Video File Fragments	15,000
Testing Video File Fragments	1000

We adapted the FFT dataset (Mittal et al., 2019) from IEEE Dataport (which is a refined dataset of file fragments of different content types) and produced classified tasks in CSV format. The format used includes the content type, file size, and length of the task. The customized dataset can be easily used in any simulation environment. Table 8 shows the refined FFT dataset containing 64,000 records. Moreover, statistics of training and testing datasets presented in Table 9.

We thoroughly tested our proposed model with 1500 different configurations of iterations (from 10 to 1500) and populations (from 1 to 100) to obtain the best configurations. We calculated the average statistics for each of the 100 configurations and selected the best configuration among these total configurations. The statistics of the best configurations are shown in Table 10. Using the best selected configurations, we presented statistics on the makespan in Figure 3, response time in Figure 4, throughput in Figure 5, and fitness value in Figure 6. So, based on these statistics, we selected the best configuration with 900 iterations and a population of 80. The selected configuration gives the best results in terms of makespan, response time and throughput. These statistics are shown in Table 10.

4.2 | Experimental analysis

In this section, a simulation-based comparison is made between the proposed model and the two existing models. The existing models are the following:

1. FTF (Junaid, Sohail, Rais, et al., 2020), which uses a modified version of Cat Swarm Optimization (CSO) along with SVM. First, the proposed system classifies data in the cloud from different sources into different types, such as text, images, video, and audio, by using one to several types of SVM classifiers. Then, the data is input to the modified load balancing algorithm CSO, which efficiently distributes the load among VMs. The proposed approach is divided into two main modules: 'Data Classification based on SVM' and 'Load Balancing using CSO'. The input

TABLE 10 Iterations and population's best statistics (Adil et al., 2022)

Iterations	Particles	Makespan	Response time	Throughput	Fitness value
100	100	22.9118	22.8118	21.82328997	10772.2
200	90	22.5813	22.4813	22.14570621	10586.6
300	90	22.4463	22.3463	22.27733907	10626.4
390	85	22.1558	22.0558	22.57119975	10444.6
510	50	22.0992	21.9992	22.62610312	10410.8
570	70	22.0624	21.9624	22.66382433	10398.8
680	60	22.0575	21.9575	22.66885002	10360.4
740	95	21.9373	21.8373	22.7923988	10348.8
900	80	21.5538	21.6538	23.17214887	10038.6
970	90	21.9371	21.8371	22.79375705	10333
1080	85	21.8988	21.7988	22.83272593	10298.2
1180	50	21.9448	21.8448	22.78506394	10311.4
1300	45	21.9357	21.8357	22.79474512	10302.6
1370	90	21.8955	21.7955	22.83684091	10282.8
1440	45	21.879	21.779	22.85469717	10,299

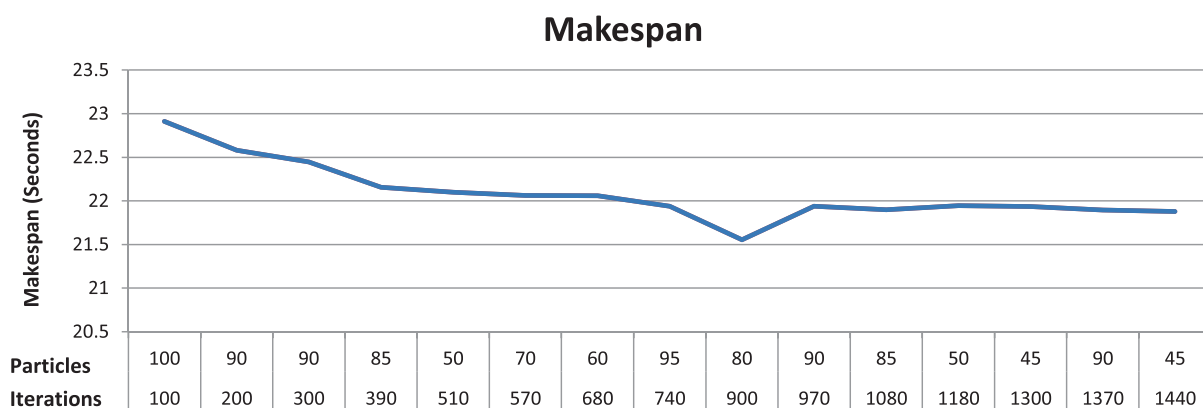


FIGURE 3 Makespan comparison by iterations and population

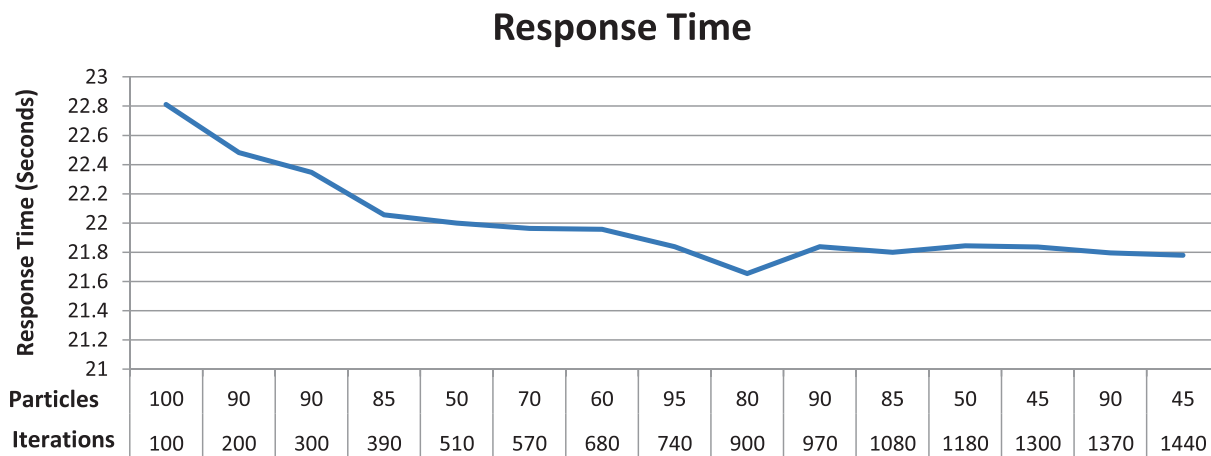


FIGURE 4 Response time comparison by iterations and population

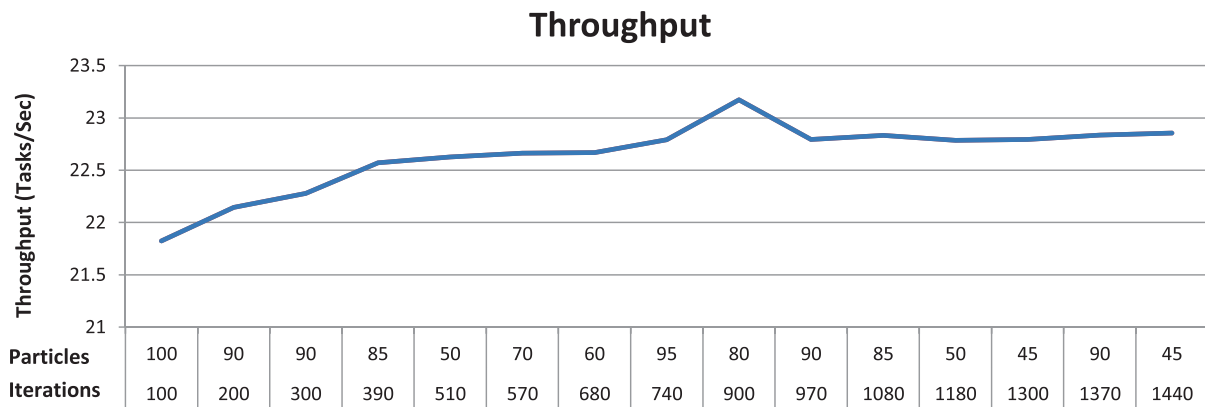


FIGURE 5 Throughput comparison by iterations and population

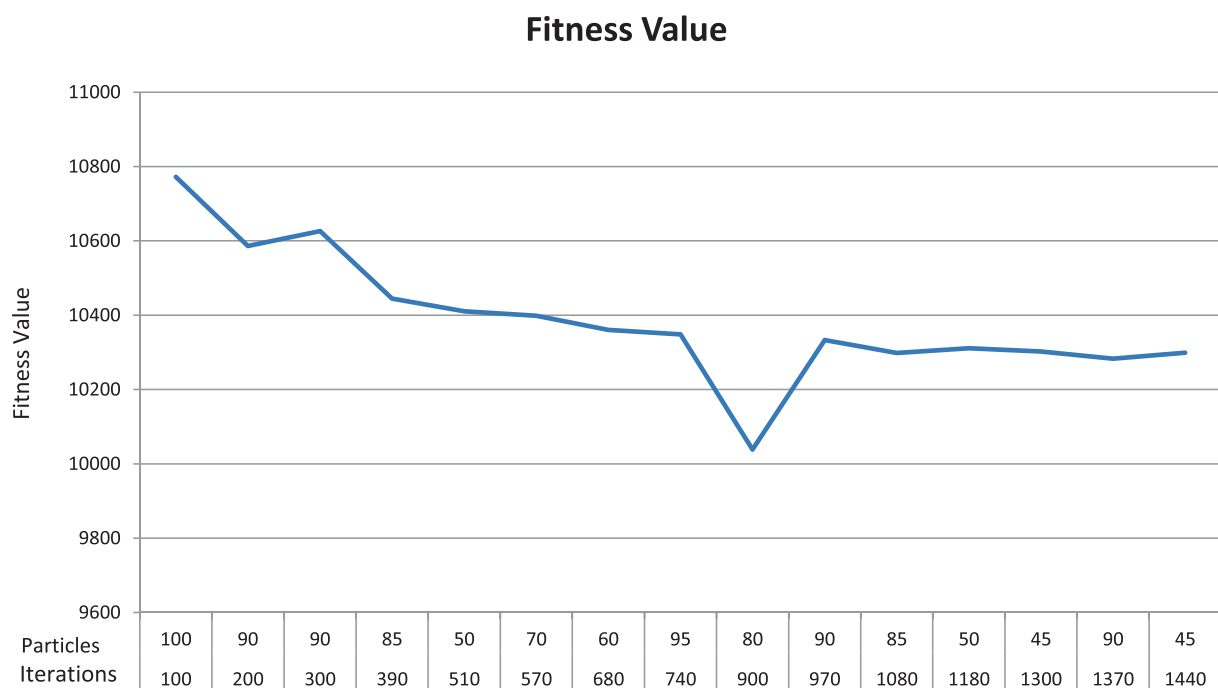


FIGURE 6 Fitness value comparison by iterations and population

to the data classification module is the collection of data in the form of video, text, audio, and images. The classification module takes the input data randomly and then performs classification on this data using polynomial SVM.

- The Ant Colony Optimization File Type Format (ACOFTF) hybrid algorithm (Junaid, Sohail, Ahmed, et al., 2020) is based on Ant Colony Optimization (ACO) and SVM. The proposed model uses the SVM classifier to classify the user's tasks into different types, such as text, images, video, and audio. Once the user's tasks are classified into appropriate categories, these classified tasks are input to the load balancing phase, which is based on ACO and efficiently distributes the load among VMs. The proposed approach is divided into two main modules: 'Data Classification based on SVM' and 'Load Balancing using ACO'. The input to the data classification module is the collection of data in the form of video, text, audio, and images. The classification module takes the input data randomly and then performs the classification of this data using polynomial SVM.

4.3 | Performance metric

The proposed model is evaluated and compared with the existing load balancing techniques explained in the above section. The QoS performance metrics such as makespan, response time, and throughput are analysed here.

1. **Makespan:** Makespan is the finish time for completing all the tasks using the available resources (please see Equation (3)).
2. **Response Time:** Response time in cloud computing is the total time it takes to respond to a cloud task requests (see Equation (5)).
3. **Throughput:** Throughput in cloud load balancing is the number of tasks processed in a unit time (see Equation (6)).

4.4 | Evaluation of analysis

Evaluation of the proposed algorithm is done in two phases; (1) Classification QoS measures and (2) Load balancing QoS measures.

4.4.1 | Classification evaluation

To validate the proposed classification model, accuracy, recall, precision, and F-measure were used as evaluation parameters. The results of the proposed approach were compared with state-of-the-art content-based planning algorithms such as DTFT (Junaid, Sohail, Rais, et al., 2020), ACOFTF (Junaid, Sohail, Ahmed, et al., 2020), and Bayes Net (Çiğşar & Ünal, 2019) (as shown in Table 11). The results of the comparison are shown in Figure 7. In the result-related figures, the x-axis shows the accuracy, recognition, precision, and f-measure, while the y-axis shows the value of each classification QoS measure. The classification evaluation shows that CA-MLBS performs better in all classification evaluations.

4.4.2 | Load balancing evaluation

1. **Evaluation based on Makespan:** We conducted a thorough analysis of CA-MLBS for the makespan metric and compared it to DTFT and ACOFTF scheduling techniques. Our analysis is based on different configurations of VMs and tasks. We presented the statistics of our analysis in Table 12. We used the abbreviations Makespan (M), Response Time (RT), and Throughput (T) to make the table more readable. We have presented the graph of statistics in Figure 8. In the above figure, the x-axis shows the number of tasks and the y-axis shows the makespan value. In each column, the respective algorithm is indicated. The results show that CA-MLBS optimized makespan up to 15% compared to ACOFTF and by 29% compared to DTFT.

TABLE 11 Comparison of all classifiers

Measure	CA-MLBS	ACOFTF (Junaid, Sohail, Ahmed, et al., 2020)	DTFT (Junaid, Sohail, Rais, et al., 2020)	Bayes net (Çiğşar & Ünal, 2019)
Accuracy	0.988	0.984	0.97	0.82
Recall	0.968	0.959	0.95	0.81
Precision	0.979	0.973	0.96	0.81
F-measure	0.98	0.966	0.97	0.82

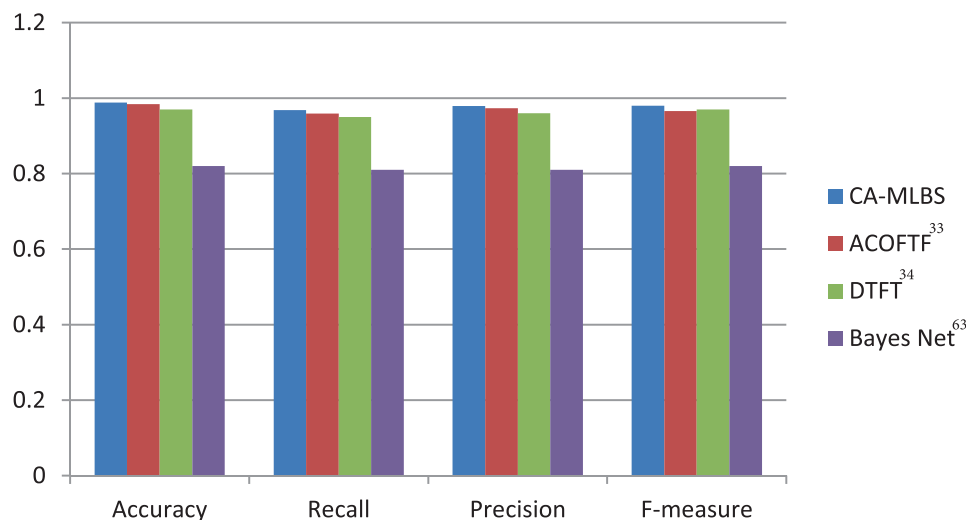
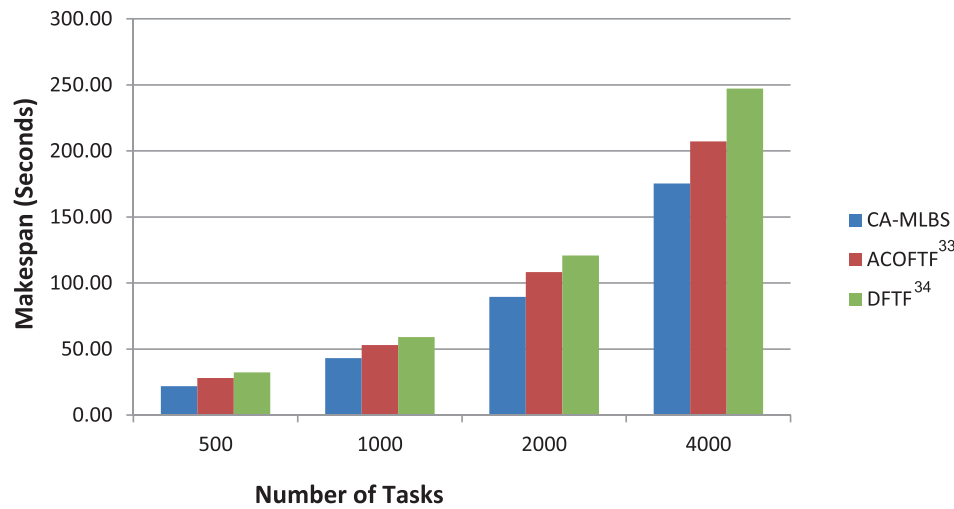
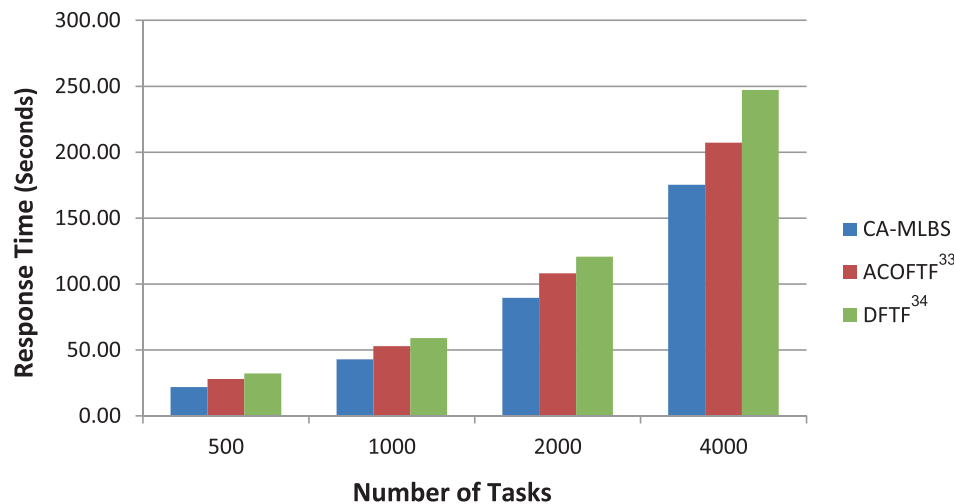


FIGURE 7 Classification results for accuracy, recall, precision, and F-measure

TABLE 12 Load balancing's QoS measures comparison of all algorithms

Tasks	CA-MLBS			ACOFTF (Junaid, Sohail, Ahmed, et al., 2020)			DFTF (Junaid, Sohail, Rais, et al., 2020)		
	M	RT	T	M	RT	T	M	RT	T
500	21.92	21.82	23.81	28.10	28.03	16.05	32.27	32.22	14.97
1000	43.04	42.94	23.23	53.02	52.96	17.44	59.04	58.99	16.33
2000	89.57	89.47	22.33	108.23	108.17	17.12	120.85	120.78	16.13
4000	175.32	175.22	22.82	207.23	207.16	17.88	247.24	247.17	15.80

Note: M is the short-form of makespan in the table. RT is the short-form of response time in the table. T is the short-form of throughput in the table.

**FIGURE 8** Makespan based results comparison**FIGURE 9** Response time based results comparison

2. Evaluation based on response time: We performed a thorough analysis of the CA-MLBS based on response time and compared it to DTFT and ACOFTF. The analysis was performed based on different configurations of VMs and tasks. We have presented the statistics of these analyzes in Table 12 and a comparison graph is presented in Figure 9. In the figure, the number of tasks is shown on the x-axis and the response time value is shown on the y-axis. In each column, the respective algorithm is indicated. The results show that CA-MLBS optimized the response time by up to 15% compared to ACOFTF and by up to 29% compared to DFTF.

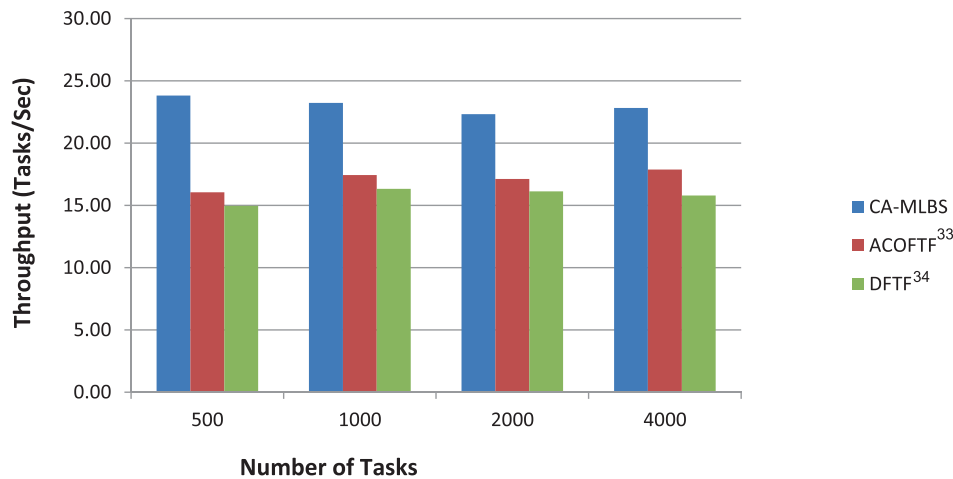


FIGURE 10 Throughput based results comparison

3. **Evaluation based on throughput:** We performed the in-depth analysis of CA-MLBS with a comparison of DTFT and ACOFTF approaches. The analysis was performed based on different configurations of VMs and tasks. We have presented the statistics of these analyzes in Table 12 and Figure 10. In the aforementioned figure, the x-axis shows the number of tasks and the y-axis shows the throughput value. In each column, the respective algorithm is shown. These results show that the CA-MLBS approach optimizes throughput by up to 21% compared to ACOFTF and 44% compared to DFTF.

5 | CONCLUSION AND FUTURE WORK

Load balancing based on the content of cloud tasks in the cloud computing environment can significantly improve workload distribution. Several studies have been presented that classify cloud tasks based on content types. Some of them use machine learning to improve the balanced distribution of workload across VMs. We proposed a hybrid load balancing model CA-MLBS that provides improved load balancing compared to current approaches. Our proposed model is based on two steps: (1) a content-based classification and (2) a load balancing step. The content-based classification process performs classification of tasks into four classes: Video, Audio, Image, and Text. This reduces the processing overhead for the load balancer through preprocessing, learning, feature extraction, and classification. After classification, we converted the classified tasks to comma-separated value format. The classified tasks also have different task lengths and sizes depending on the task type. In the load balancing step, we input four groups of VMs and classified task collections. Each VM group is dedicated to a specific task type and has different resource configurations. Our model uses a PSO-based load balancing algorithm to distribute the balanced tasks to the corresponding type of VMs. We conducted a thorough analysis of our model and compared the results of the analysis with state-of-the-art approaches such as DFTF and ACOFTF. As QoS metrics, we used makespan, response time, and throughput. The results of the comparison show that CA-MLBS improved makespan, response time, and throughput compared to DFTF and ACOFTF. Moreover, CA-MLBS is an easy-to-use and simplified approach compared to the existing content type algorithms.

In the future, the proposed approach will be extended by using meta-heuristics such as the genetic algorithm, and QoS parameters such as energy consumption, overhead time, migration time, and optimization time will also be considered. The proposed approach is based on the assumption that the input tasks lie within one of the four categories like video, audio, image, or text without overlapping. Moreover, the overlapping collections of tasks and multi-objective methodologies will also be considered in the future.

DATA AVAILABILITY STATEMENT

The data is openly available in a public repository, which can be directly accessed by FILE FRAGMENT TYPE (FFT)-75 DATASET. (<https://iee-dataport.org/open-access/file-fragment-type-fft-75-dataset>).

ORCID

Muhammad Aleem  <https://orcid.org/0000-0001-8342-5757>

Vicente Garcia Diaz  <https://orcid.org/0000-0003-2037-8548>

Jerry Chun-Wei Lin  <https://orcid.org/0000-0001-8768-9709>

REFERENCES

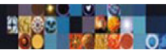
- Abrol, P., Guupta, S., & Singh, S. (2020). Nature-inspired metaheuristics in cloud: A review. *ICT Systems and Sustainability*, 1077, 13–34.
- Adhikari, M., & Amgoth, T. (2018). Heuristic-based load-balancing algorithm for IaaS cloud. *Future Generation Computer Systems*, 81, 156–165.
- Adil, M., Nabi, S., & MS, R. A. Z. A. (2022). PSO-CALBA: Particle swarm optimization based content-aware load balancing algorithm in cloud computing environment. *Computing and Informatics*.
- Agarwal, R., Baghel, N., & Khan, M. A. (2020). Load balancing in cloud computing using mutation based particle swarm optimization. In: 2020 International Conference on Contemporary Computing and Applications (IC3A) IEEE, pp. 191–195.
- Ahmad, M. O., & Khan, R. Z. (2018). Load balancing tools and techniques in cloud computing: A systematic review. *Advances in Computer and Computational Sciences*, 554, 181–195.
- Amazon EC. (2015). Amazon web services.
- Attiya, I., Abd Elaziz, M., & Xiong, S. (2020). Job scheduling in cloud computing using a modified Harris hawks optimization and simulated annealing algorithm. *Computational Intelligence and Neuroscience*, 2020, 1–17.
- AWSACC Services. Using a PostgreSQL database as an AWS DMS source.
- Boniface, M., Nasser, B., Papay J, Phillips, S. C., Servin, A., Yang, X., Zlatev, Z., Gogouvitis, S. V., Katsaros, G., Konstanteli, K., Kousiouris, G., Menycthas, A., & Kyriazis, D. (2010). Platform-as-a-service architecture for real-time quality of service management in clouds. In: 2010 Fifth International Conference on Internet and Web Applications and Services IEEE, pp. 155–160.
- Calheiros, R. N., Ranjan, R., Beloglazov, A., De Rose, C. A., & Buyya, R. (2011). CloudSim: A toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms. *Software: Practice and Experience*, 41(1), 23–50.
- Celebi, M. E., & Aydin, K. (2016). *Unsupervised learning algorithms*. Springer.
- Cervantes, J., Garcia-Lamont, F., Rodríguez-Mazahua, L., & Lopez, A. (2020). A comprehensive survey on support vector machine classification: Applications, challenges and trends. *Neurocomputing*, 408, 189–215.
- Chang, C. C., & Lin, C. J. (2011). LIBSVM: A library for support vector machines. *ACM Transactions on Intelligent Systems and Technology (TIST)*, 2(3), 1–27.
- Chappell, D. (2010). *Introducing the windows azure platform*. David Chappell & Associates White Paper.
- Çiğşar, B., & Ünal, D. (2019). Comparison of data mining classification algorithms determining the default risk. *Scientific Programming*, 2019, 1–8.
- Cingolani P. (2011). Jswarm-PSO. <http://jswarm-psy.sourceforge.net/>
- Cusumano, M. (2010). Cloud computing and SaaS as new computing platforms. *Communications of the ACM*, 53(4), 27–29.
- Devaraj, A. F. S., Elhoseny, M., Dhanasekaran, S., Lydia, E. L., & Shankar, K. (2020). Hybridization of firefly and improved multi-objective particle swarm optimization algorithm for energy efficient load balancing in cloud computing environments. *Journal of Parallel and Distributed Computing*, 142, 36–45.
- Fadlallah, S. O., Anderson, T. N., & Nates, R. J. (2021). Artificial neural network–particle swarm optimization (ANN-PSO) approach for behaviour prediction and structural optimization of lightweight sandwich composite heliostats. *Arabian Journal for Science and Engineering*, 46(12), 12721–12742.
- Hanine, M., & Benlahmar, E. H. (2020). A load-balancing approach using an improved simulated annealing algorithm. *Journal of Information Processing Systems*, 16(1), 132–144.
- Harris, C. R., Millman, K. J., Walt, V., Gommers, R., Virtanen, P., Cournapeau, D., Wieser, E., Taylor, J., Berg, S., Smith, N. J., Kern, R., Picus, M., Hoyer, S., van Kerkwijk, M. H., Brett, M., Haldane, A., Del Río, J. F., Wiebe, M., Peterson, P., ... Oliphant, T. E. (2020). Array programming with NumPy. *Nature*, 585(7825), 357–362.
- Hussain, A., Aleem, M., Khan, A., Iqbal, M. A., & Islam, M. A. (2018). RALBA: A computation-aware load balancing scheduler for cloud computing. *Cluster Computing*, 21(3), 1667–1680.
- Ibrahim, M., Nabi, S., Baz, A., Alhakami, H., Summair Raza, M., Hussain, A., Salah, K., & Djemame, K. (2020). An in-depth empirical investigation of state-of-the-art scheduling approaches for cloud computing. *IEEE Access*, 8, 128282–128294.
- Ibrahim, M., Nabi, S., Baz, A., Naveed, N., & Alhakami, H. (2020). Towards a task and resource aware task scheduling in cloud computing: An experimental comparative evaluation. *International Journal of Networked and Distributed Computing*, 8(3), 131–138.
- Jena, U., Das, P., & Kabat, M. (2022). Hybridization of meta-heuristic algorithm for load balancing in cloud computing environment. *Journal of King Saud University-Computer and Information Sciences*, 34, 2332–2342.
- Junaid, M., Sohail, A., Ahmed, A., Baz, A., Khan, I. A., & Alhakami, H. (2020). A hybrid model for load balancing in cloud using file type formatting. *IEEE Access*, 8, 118135–118155.
- Junaid, M., Sohail, A., Rais, R. N. B., Ahmed, A., Khalid, O., Khan, I. A., Hussain, S. S., & Ejaz, N. (2020). Modeling an optimized approach for load balancing in cloud. *IEEE Access*, 8, 173208–173226.
- Kaur, A., & Kaur, B. (2022). Load balancing optimization based on hybrid heuristic-metaheuristic techniques in cloud environment. *Journal of King Saud University-Computer and Information Sciences*, 34, 813–824.
- Kaur, G. (2020). Framework for resource Management in Cloud Computing. In *International conference on information and communication Technology for Intelligent Systems* (pp. 25–32). Springer.
- Khorsand, R., Ghobaei-Arani, M., & Ramezanzpour, M. (2019). A self-learning fuzzy approach for proactive resource provisioning in cloud environment. *Software: Practice and Experience*, 49(11), 1618–1642.
- Kong, L., Mapetu, J. P. B., & Chen, Z. (2020). Heuristic load balancing based zero imbalance mechanism in cloud computing. *Journal of Grid Computing*, 18(1), 123–148.
- Krishnan, S., & Gonzalez, J. L. U. (2015). *Building your next big thing with google cloud platform: A guide for developers and enterprise architects*. Springer.
- Kumar, K. P., Ragnathan, T., Vasumathi, D., & Prasad, P. K. (2020). An efficient load balancing technique based on cuckoo search and firefly algorithm in cloud. *Algorithms*, 423, 422–432.
- Lilhore, U. K., Simaiya, S., Maheshwari, S., Manhar, A., & Kumar, S. (2020). Cloud performance evaluation: Hybrid load balancing model based on modified particle swarm optimization and improved metaheuristic firefly algorithms. *International Journal of Advanced Science and Technology*, 29(5), 12315–12331.
- Liu, B. (2011). Supervised learning. In *Web data mining* (pp. 63–132). Springer.
- Majdisova, Z., & Skala, V. (2017). Radial basis function approximations: Comparison and applications. *Applied Mathematical Modelling*, 51, 728–743.
- Megharaj, G., & Kabadi, M. G. (2019). Metaheuristic-based virtual machine task migration technique for load balancing in the cloud. In *Integrated intelligent computing Communication and Security* (pp. 435–446). Springer.

- Mirjalili, S., Farris, H., & Aljarah, I. (2020). Introduction to evolutionary machine learning techniques. In *Evolutionary machine learning techniques* (pp. 1–7). Springer.
- Mishra, K., & Majhi, S. K. (2021). A binary bird swarm optimization based load balancing algorithm for cloud computing environment. *Open Computer Science*, 11(1), 146–160.
- Mishra, S. K., Sahoo, B., & Parida, P. P. (2020). Load balancing in cloud computing: A big picture. *Journal of King Saud University-Computer and Information Sciences*, 32(2), 149–158.
- Mittal, G., Korus, P., & Memon, N. (2019). File Fragment Type (FFT)–75 Dataset.
- Muthsamay, G., & Ravi, C. S. (2020). Task scheduling using artificial bee foraging optimization for load balancing in cloud data centers. *Computer Applications in Engineering Education*, 28(4), 769–778.
- Muthusamy, G., & Chandran, S. R. (2021). Cluster-based task scheduling using K-means clustering for load balancing in cloud datacenters. *Journal of Internet Technology*, 22(1), 121–130.
- Nabi, S., Ahmad, M., Ibrahim, M., & Hamam, H. (2022). AdPSO: Adaptive PSO-based task scheduling approach for cloud computing. *Sensors*, 22(3), 920.
- Nabi, S., & Ahmed, M. (2021a). PSO-RDAL: Particle swarm optimization-based resource- and deadline-aware dynamic load balancer for DEADLINE constrained cloud tasks. *The Journal of Supercomputing*, 77, 7476–7508. <https://doi.org/10.1007/s11227-021-04062-2>
- Nabi, S., & Ahmed, M. (2021b). OG-RADL: Overall performance-based resource-aware dynamic load-balancer for deadline constrained cloud tasks. *The Journal of Supercomputing*, 77, 1–33.
- Nabi, S., Aleem, M., Ahmed, M., Islam, M. A., & Iqbal, M. A. (2022). RADL: A resource and deadline-aware dynamic load-balancer for cloud tasks. *The Journal of Supercomputing*, 78, 1–35.
- Nabi, S., Ibrahim, M., & Jimenez, J. M. (2021). DRALBA: Dynamic and resource aware load balanced scheduling approach for cloud computing. *IEEE Access*, 9, 61283–61297.
- Naik, K. J. (2020). A dynamic ACO-based elastic load balancer for cloud computing (D-ACOELB). In *Data engineering and communication technology* (pp. 11–20). Springer.
- Neelima, P., & Reddy, A. R. M. (2020). An efficient load balancing system using adaptive dragonfly algorithm in cloud computing. *Cluster Computing*, 23(4), 2891–2899.
- Phi, N. X., Tin, C. T., Thu, L. N. K., & Hung, T. C. (2018). Proposed load balancing algorithm to reduce response time and processing time on cloud computing. *The International Journal of Computer Networks & Communications*, 10(3), 87–98.
- Pinho, P. d C A., Nedjah, N., & Macedo, M. d L. (2020). Detection and classification of pulmonary nodules using deep learning and swarm intelligence. *Multi-media Tools and Applications*, 79(21), 15437–15465.
- Pradhan, A., Bisoy, S. K., & Das, A. (2021). A survey on pso based meta-heuristic scheduling mechanism in cloud computing environment. *Journal of King Saud University-Computer and Information Sciences*, 34, 4888–4901.
- Rabbani, M., Wang, Y. L., Khoshkangini, R., Jelodar, H., Zhao, R., & Hu, P. (2020). A hybrid machine learning approach for malicious behaviour detection and recognition in cloud computing. *Journal of Network and Computer Applications*, 151, 102507.
- Rafieyan, E., Khorsand, R., & Ramezanpour, M. (2020). An adaptive scheduling approach based on integrated best-worst and VIKOR for cloud computing. *Computers & Industrial Engineering*, 140, 106272.
- Rajagopalan, A., Modale, D. R., & Senthilkumar, R. (2020). Optimal scheduling of tasks in cloud computing using hybrid firefly-genetic algorithm. In *Advances in decision sciences, image processing, security and computer vision* (pp. 678–687). Springer.
- Reddy, Y. P., & Varma, N. M. K. (2020). Review on supervised learning techniques. In *Emerging research in data engineering systems and computer communications* (pp. 577–587). Springer.
- Saeedi, S., Khorsand, R., Bidgoli, S. G., & Ramezanpour, M. (2020). Improved many-objective particle swarm optimization algorithm for scientific workflow scheduling in cloud computing. *Computers & Industrial Engineering*, 147, 106649.
- Sahli, H. (2020). An introduction to machine learning. In *TORUS 1-toward an open resource using Services: Cloud computing for environmental data* (pp. 61–74). Wiley Online Library.
- Semmoud, A., Hakem, M., Benmammar, B., & Charr, J. C. (2020). Load balancing in cloud computing environments based on adaptive starvation threshold. *Concurrency and Computation: Practice and Experience*, 32(11), e5652.
- Sen, P. C., Hajra, M., & Ghosh, M. (2020). Supervised classification algorithms in machine learning: A survey and review. In *Emerging technology in modelling and graphics* (pp. 99–111). Springer.
- Serrano, N., Gallardo, G., & Hernantes, J. (2015). Infrastructure as a service and cloud technologies. *IEEE Software*, 32(2), 30–36.
- Sharma, M., & Garg, R. (2020). HIGA: Harmony-inspired genetic algorithm for rack-aware energy-efficient task scheduling in cloud data centers. *Engineering Science and Technology, An International Journal*, 23(1), 211–224.
- Sunyaev, A. (2020). Cloud computing. In *Internet computing* (pp. 195–236). Springer.
- Zheng, Z., Xie, K., He, S., & Deng, J. (2017). A multi-objective optimization scheduling method based on the improved differential evolution algorithm in cloud computing. In *International conference on cloud computing and security* (pp. 226–238). Springer.

AUTHOR BIOGRAPHIES

Muhammad Adil is a student of M.S in computer sciences at Virtual University of Pakistan. Her received the Master in Computer Sciences degree from the Virtual University of Pakistan, in 2007. He currently works as Product Manager at the Aspose Pty Ltd. Aspose is a development software company that offers numerous award-winning Cloud APIs.

Dr. Said Nabi is currently serving as an Instructor of IT and Computer Science at the Department of Computer Sciences and Information Technology, Virtual University of Pakistan Rawalpindi Campus. He has completed his PhD in computer sciences from Capital University of



Science and Technology (CUST). He served as Software Engineer and Developer at Esided Solutions (US Based Company) and also worked as Free Lance Software Developer in PHP and MySQL.

Muhammad Aleem received a Ph.D. degree in computer science from the Leopold-Franzens-University, Innsbruck, Austria in 2012. His research interests include parallel and distributed computing comprising programming environments, multi-/many-core computing, performance analysis, cloud computing, and big-data processing. He is currently working as Full Professor at National University of Computer and Emerging Sciences, Islamabad, Pakistan.

Vicente Garcia Diaz is an Associate Professor in the Department of Computer Science at the University of Oviedo (Spain). His teaching interests are primarily in the design and analysis of algorithms and the design of domain-specific languages. His current research interests include decision support systems, health informatics and eLearning.

Jerry Chun-Wei Lin is a full professor in Western Norway University of Applied Sciences, Norway. His research interests include big data analytics, AI/ML/DL, optimization, privacy and security preservation, IoTs and bio-informatics.

How to cite this article: Adil, M., Nabi, S., Aleem, M., Diaz, V. G., & Lin, J. C.-W. (2022). CA-MLBS: content-aware machine learning based load balancing scheduler in the cloud environment. *Expert Systems*, e13150. <https://doi.org/10.1111/exsy.13150>