Virtual Machine Task Classification Using Support Vector Machine and Improved MFO Based Task Scheduling

D.Radhika
Assistant professor
Department of computer Science and Engineering
Vivekanandha College of Engineering for women
Tirunchengodu, Namakkal- 637 205
Radhikadeva6@gmail.com

Dr.M.Duraipandian
Professor
Department of computer Science and Engineering
Nehru Institute of Technology
Kaliyapuram, Coimbatore
Tamil Nadu – 641 105
durainithi@gmail.com

Abstract

The processing of data in Big data computing necessitates a significant number of CPU cycles and network bandwidth. Dataflow is a huge data processing programming model that comprises of jobs structured in a graph structure. Scheduling these jobs is one of the most active study fields, with the primary goal of allocating tasks based on available resources. It is critical to efficiently schedule jobs in a way that minimizes task completion time and maximizes resource utilization. In recent years, many research works on task scheduling problems in cloud computing and various heuristic approaches have been evolved which the thesis focuses upon. Most of these efforts are focused on improving the performance such as minimizing the makespan and efficient utilization of cloud resources that benefits the cloud users and the providers. In this paper, Big Data analysis processing in cloud environment for efficient dynamic task scheduling by using different techniques as machine learning classifier and optimization approach. In machine learning classifier as Support Vector Machine (SVM) for classification of virtual machine task classification. In this classifier can classify the incoming request efficiently and reduce the makespan and execution time. Further, we used moth flame optimization technique for allocating the classified task from SVM classifier. In this proposed system have classification of virtual machine (VM) task and evaluate the resource allocation decision making procedures. In this experiment, we carried out by using cloud simulation environment to evaluate and analysis the proposed method. In this proposed scheme can efficiently reduce the makespan time and load balancing to improve the better VM Classification. And aslo we compare the proposed moth flame optimization (MFO) with min-max algorithm and particle swarm optimization to compare the performance respectively.

Keyword: cloud computing, moth flame optimization, makespan, virtual machine and load balancing.

1 Introduction

The rapid development of internet made a great impact on delivering commercial services to the peoples all over the world. Big data analytics is empowered by machine learning or statistical algorithms to process big data and understand meaningful information out of it. Consequently, it is important to process those data within a limited time. In the term Cloud Computing is a computing framework which provides on demand access to collection of configurable calculating resources over the internet. It is a pretty bigger, and it is successful every day where great computation concentrated tasks can be computed on distributed computing resources [1]. Due to the speedy sharing of huge web resources over the internet, the cloud framework revolution has propelled the IT industry to new heights in recent years. Cloud computing environment that includes set of web resources such as Servers, Software Platforms, Applications, and Storages [2]. The cloud users interface the cloud computing environment through virtualization in order to utilize the resources via internet. The Internet serves as a backbone of cloud computing environments to provide ubiquitous access to the resources from anywhere in the world using computing terminals such as Personal Computers, Laptop, Smart Phones and other IOT Devices [3]. A virtual machine (VM) is a product that uses a machine to execute programs in the same way as a PM does. VMs are classified into two types based on their usage and degree of resemblance to an actual computer. A VM is designed to run a single application, implying that it operates as a single process. Such VMs are often well matched to at least one programming dialect and provide system ease and versatility [4]. In the datacenter, cloud users request specific resources such as storage, CPU, and memory. After approving this client request, the DC will distribute the VM to a server and hold every single key resource on the server. Depending on the server's capacity and the VMs' resource requirements, a server can host a large number of virtual machines. VMC technique is incorporated in the Cloud Resource Management System (CRMS) to increase the energy-efficiency of Cloud. Hardware failure of existing PMs and expansion of new PMs is nonstop occasions in the server farm [5]. Besides, asset prerequisite to achieve the rest of the undertakings of existing administration demand develops with the course of time. Subsequently, as time advances, remapping of outstanding workload to presently accessible assets end up plainly unavoidable to maintain the improvement of Cloud asset use. The VMC technique is classified as two groups namely, Dynamic and Static. The workload or resource requirements of any VM and its location (i.e., its hosting PM) can be dynamic in Dynamic VMC method, since they vary over time. Static VMC technique calculation doesn't consider the current VM-to-Server task while picking another PM for a VM [6].

In Cloud Computing, DC work provides coveted power to respond to engineer's requests on process, stockpiling, and system administration limits. Likewise, a large DC, running a virtualization determination, grants for higher use of the hardware's energy through the connected math multiplexing of engineers' applications. Cloud computing resources are managed through the centralized resource manager. The unified resource administrator allocates the tasks to the required VMs. The assets of cloud DC are accessible to the clients/applications through VMs [7]. VMs are utilized to meet the asset prerequisite and run time bolster for the applications. Specifically executing an application for required asset can be made accessible through two stages: making a case of the VM as required by the application (VMs provisioning) and booking the demand to the physical assets also called asset provisioning. The VM is a product reflection with the looks of a PC framework's equipment (genuine machine). The VM innovation has turned out to be well known as of late in server farms and distributed computing situations since it has various advantages including server combination, live movement, and security seclusion [8].

Distributed computing depends up on the idea of virtualization that epitomizes different administrations that can meet the client prerequisite in a cloud computing environment. VMs are intended to keep running on a server to furnish a numerous OS condition with the help of different applications. At least one VM can be set or conveyed on a PM that meets the necessity for the VM [9]. The errand can be planned by dynamic load adjusting between the host in distributed computing conditions are accomplished utilizing representation innovation. An essential technique of resource allocation as Static Scheduling Algorithm, Dynamic Scheduling Algorithm, Heuristic Scheduling Algorithms, Opportunistic Load Balancing, Min-Min technique and Max-Min techniques [10]. The main attraction of cloud computing is to manage computing power, storage, various kinds of platforms and services which are entirely managed by resource management algorithms. Dynamic load balancing, Ant Colony Optimization (ACO), Genetic and Enhanced GAs, First and best fit decreasing (FFD and BFT), and Service Level Agreement (SLA) are the common algorithms used for the management of resources in the VMs [11].

All data files are saved in the cloud storage system in cloud computing. Each data file is broken up into a number of segments. Fragmentation occurs because files aren't stored together in a logical manner, which leaves no place for other files and wastes capacity in the storage system. As a result, both computational and operational costs rise. Since the workload is imbalanced, the task's total execution time is large (makespan) [12]. This study focuses on optimizing resource allocation of a cloud computing environment to enhance the performance of the cloud and to make it energy efficient by maximizing the utilization of resources and reduce the makespan in a cloud computing environment.

2 Literature review

In order to make understand the techniques presented in the proposed research, this chapter presents a thorough survey of literature. The survey comprises of the different methodologies for resource allocation, load balancing, task scheduling and optimal allocation of VM.

Pratik P. Pandya and Hitesh A. Bheda [13] designed a dynamic resource allocation technique for management of cloud data access. As part of the coordination of cloud supplier activities for using and allocating rare assets within the confines of the cloud condition, Resource Allocation Strategy (RAS) is used to handle challenges related to a particular cloud application. It requires the rules and techniques of the cloud environment to achieve an improved data transfer. Dynamic Resource Allocation (DRA) is especially prominent research domain in cloud computing because of its live application in server application. Experimental results of clouds are heterogeneous as well as dynamic in nature. In proposed system, details of allocation method, affinity of VM and ideas behind the good performance over non-affinity group are identified and some idea about new technique to improve performance is suggested as well. In this literature had shown some basic techniques for allocation of VM and grouping of VM.

N. S. Raghava and Deepti Singh [14] presented an overall review of the recent load balancing ideas in Cloud system. The load balancing algorithms aims at achieving higher throughput, a few goes for accomplishing least reaction time, some different intends to accomplish most extreme resource use while a few goes for accomplishing an exchange off between every one of these measurements. The strategy uses the advantages of both appropriated and incorporated approach of registering. A correlation has been done on the premise of various criteria like versatility, resource utilization, network overhead, algorithm toughness, fault resisting, and response time, etc.

R.S. Mohana [15] has presented a position balanced Parallel Particle Swarm Optimization (PPSO) for the role of resource assignment in cloud. A creative procedure PB-PPSO is presented for dispensing assets. The fundamental goal of PB-PPSO is to identify cutting-edge assets for setting up projects with a shorter make-span and lower cost. A collection of procedures is compiled from the most efficient training resource available. Resource allocation to new users is learned by training during the testing phase. It is clear that the PB-PPSO strategy is superior to current approaches like Support Vector Machines and Artificial Neural Networks (ANN). In the PB-PPSO strategy, the advanced arrangement of assets is resolved for the arrangement of errands by utilizing the PSO calculation. At that point the standards are created for the grouping procedure. On the off chance that the entry rate of clients is 500, the aggregate benefit is 720$ and the reaction time is 78ms. In light of the examination and the outcomes from the test demonstrates the proposed approach works superior to anything the other existing frameworks with high benefit and less normal reaction time.

Daji Ergu et al. [16] developed an analytical process of hierarchical systems for scheduling the tasks or jobs allocation and the assumption of resources in the cloud environment. Because there are various alternative PCs with varying constraints, resource allocation is a complicated operation in the cloud computing scenario. The goal of this research is to offer a methodology for performing focused asset distribution in a cloud computing environment. The pairwise examination network technique and the Analytic Hierarchy Process are used to position the resource assignment project, which provides the available assets and customer preferences. The calculating assets might be assigned based on the rank of the projects. Furthermore, a started inclination lattice is employed to differentiate conflicting components and improve the consistency percentage when conflicting weights in distinct jobs are assigned. The results show that it is worthwhile to further measure the inconsistent data, and that the improvement of the consistency ratio and the job weight are utilized to dynamically assign computer resources in a cloud computing environment.

As a multi-constrained and bi-objective problem, the authors in paper [17] optimized cost and energy by scheduling in a Federation of dispersed datacenters. Multi-objective evolutionary algorithms were used in combination with low-level backfilling heuristics to discover the best time to launch. The suggested approach aimed to improve energy efficiency and a number of service-related parameters. With medium and large workload sizes, the algorithm was assessed and compared to a round-robin scheduler in order to represent genuine high-performance computing applications. Because of its efficiency and accuracy, the algorithm had been able to produce Pareto-optimal plans where no single plan could significantly outperform the others, even if they were both cheaper. In spite of this, their algorithm failed to account for tasks' varying bandwidth needs.

3 Proposed methodology

In figure 1 represent that the proposed methodology, Initializing the data collection by collecting the data and then pre-process the data to the structure tasks. All the virtual machines is the first process, obtain the best VM for allocating tasks depends on the load, and once VM is identified, tasks are assigned and the process is terminated by the dynamic task scheduler. The status of virtual machine utilization is monitored and observed through support vector machine classifier results.

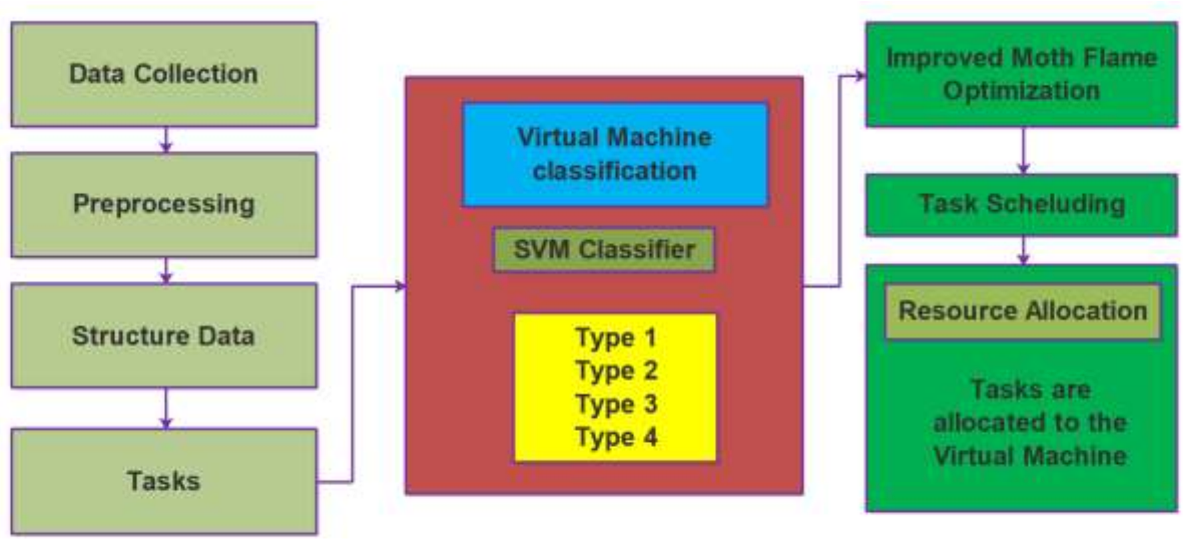

Figure 1: Proposed flow diagram

Each task has a different quality of services so that depends on the priority the utilization ratio, the tasks are allocated to the optimal VM. Based on the fitness function the tasks are assigned to the respective VM. The proposed dynamic task scheduling is formulated based on moth flame optimization which is a population-based metaheuristic optimization algorithm. To obtain the best solution using the moth fly optimization algorithm, it is essential to reduce the impossible solution. Impossible solution refers to overutilization of VM if more number of tasks are assigned to a particular VM. So the proposed model eliminates the impossible solutions and updates the best solution based on the moth movements.

3.1 TASK SCHELUDING

Task scheduling is critical because it allocates a number of related and/or unrelated jobs to the computers in the clusters that have sufficient resources. Depending on the execution time and resource availability, a good scheduler can discover the best way to distribute tasks among the cluster's machines to maximize efficiency. The mean execution time of the planned jobs is minimized using an optimal task distribution, and the assigned resources are utilized to their full potential. This is to ensure that the received calculations (tasks to be processed) respond as quickly as possible, and to minimize (prevent) resource waste. Each cloud-based big-data platform has a scheduler for managing work assignments. Several well-known Hadoop schedulers that have received a lot of attention from industry and academics are briefly discussed here as examples. The JobTracker in Hadoop manages the submission of jobs and tasks and schedules and provides resources for them. It has a scheduling mechanism based on the FIFO (First In, First Out) principle in its initial implementation. The functions of planning were initially reorganized under a single source of inspiration.

3.2 Virtual machine Classification

The VMs in this study are separated into several categories and then categorized. It's decided by using the notion of SVM. CPU and memory utilization ratios are used to classify VMs, and the results are shown as a time series with values ranging from 0 to 100. (in percentages). Each group's VMs will be sorted ascending by utilization percentage. We've broken down the types of VMs being used into four categories:

Table 1: Virtual machine Classification types

Type	**VM status**	**Resource utilization requirement**
Type 1 (T_1)	Unstable Virtual machines	High resource utilization
Type 1 (T_2)	Moderately stable compared to type 1	Moderate resource utilization
Type 1 (T_3)	Stable virtual machines	High resource utilization
Type 1 (T_4)	Stable virtual machines	Less resource utilization

3.2.1 VM classification using SVM classifier

Figure 1 shows the SVM's classification of spatial locations. For data classification and machine learning, it's a well-known approach. Using the preexisting categories as input, the classifier learns to sort the data according to the categories and data types. Classification borders are delineated in SVM using the concept of classification boundary. By using data training, SVM discovers the boundaries between the two categories, i.e., liner division and nonlinear division.

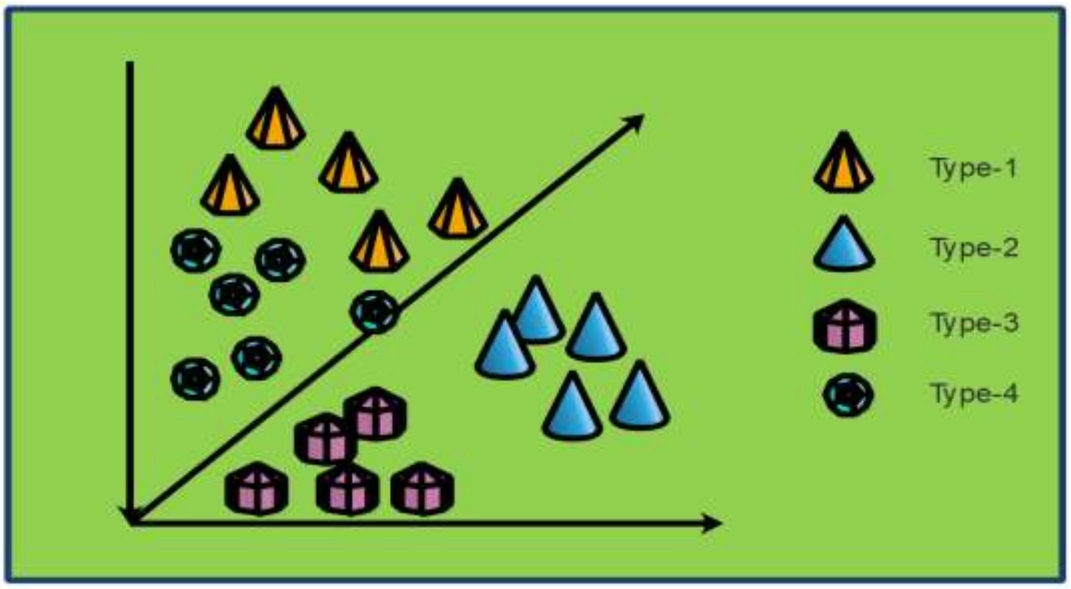

Figure 2: SVM classification

There are two major elements that determine the types of virtual machines in this study. VM average utilization and virtual machine stability are both important metrics to consider. Stability is a measure of how much demand there is for a given resource over a given length of time. To put it another way, if the VM's resource utilization is constantly changing, it is considered unstable. The VMs can be divided into four groups based on these two characteristics. SVM can be used to discover the limits. The four VM kinds are designated as A, B, C, and D, respectively. It's possible that the remaining resources will be insufficient if you use type A VMs. It's unreliable and uses a lot of resources. Type B VMs are less reliable than type A VMs, but they use fewer resources. Type C and D VMs, in contrast to A and B VMs, are more stable, meaning that the amount of resources required remains constant.

Type C VMs are more stable, but they use a lot of resources, whereas Type D VMs use a lot less. The SVM tool is used to classify these four VM types because their boundaries are uncertain. Figure 2 depicts the categorization method's data flow.

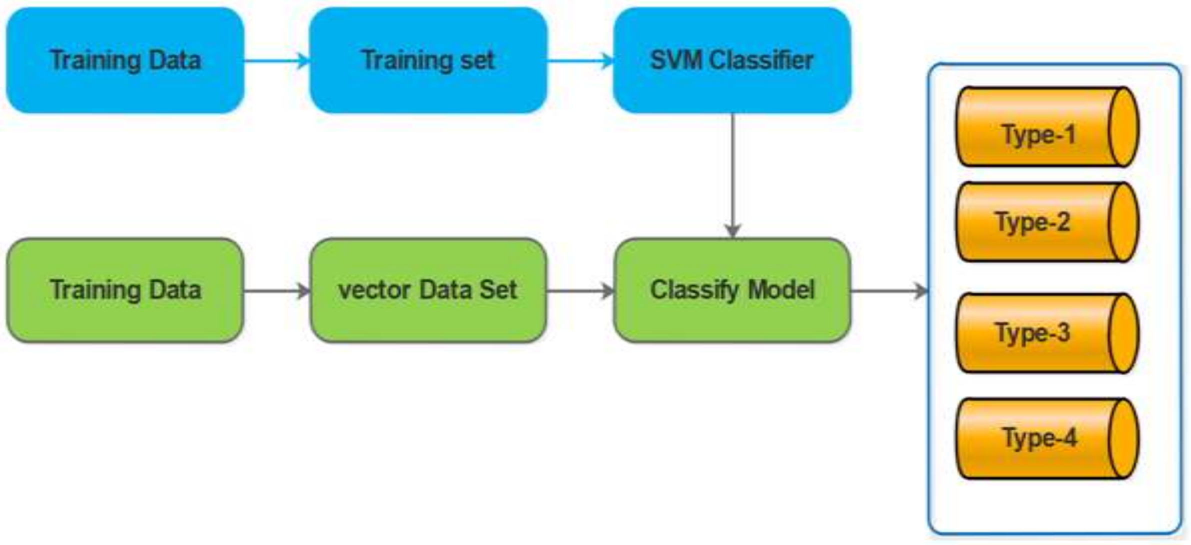

Figure 3: Flow of classification method

We must give SVM training data in order to create the decision model. The decision model is created by following the rule of the training data. VMs are pre-executed, and training data is pre-selected and prepared in advance. Prior to learning anything new, the raw data needs to be vectorized using a hash function and then turned into a training set. The SVM gains a decision model from four different VM types using machine learning. The decision model can finally classify and categorize the incoming data.

```
Step-1: Calculate and categorize the utilization of each VM (CPU & RAM) in percentages (0 percent - 100 percent).
Step-2: Retrieve data from log files, such as user duration, task type, size, and so on.
Step-3: Calculate the needed resources for each task (T) based on required resources, in percentages, and divide them into three groups before saving them in an index table in ascending order.
Step-4: If the Task (T) is high (H) then
 Allocate it to the lightest VM in the light or idle group.
Else if the (T) is normal (N) then
 Allocate it to the less utilized VM in the normal or light group respectively.
Else
Step-5: Allocate it to the lightest VM in the less (L) or moderate (M) group.
Step-6: Calculate each VM's usage and update the VM index database based on its percentage.
```

3.3 Improved moth flame optimization

The proposed dynamic task scheduling is formulated based on moth flame optimization which is a population-based metaheuristic optimization algorithm. The behavior of moth around the flame at night time is considered to obtain the optimal solution for the given issue. Generally, moths fly towards the moon at night in a straight line. In between if any light source interferes it considers that as the moon and started to rotate around the light at a specific angle. Moths are considered as search agents and the light source is considered as the best position. This best position is considered as the optimal position for other moths in the group. To obtain the best solution using the moth fly optimization algorithm, it is essential to reduce the impossible solution. Impossible solution refers to overutilization of VM if more number of tasks are assigned to a particular VM. So the proposed model eliminates the impossible solutions and updates the best solution based on the moth movements. Summarized pseudocode for the proposed optimization model is given as follows.

```
Input: Tasks $s = \{s1, s2, s2, s3, \dots sy, \}$
Output: Optimal VM, task allocation
Initialize: SVM classifier parameters, MFO parameters
begin
{
For task Si
check VM status using SVM
Calculate utilization and train the vector machine
Check the status with test data
```

```
Obtain classification results as types (T1, T2, T3, T4)
{
Initialize moths and its position randomly
for x=1…n do
Calculate the fitness function
end
while current iteration < max iteration
update position using $Fopt$
calculate number of flames
evaluate the fitness function
calculate the distance d using N and L
update the position using $S(N_i, L_j) = d_i e^{vt} \cos(2\pi t) + L_j$
obtain the optimal solution using $F_{opt} = round(f_{max} - I_c \times \frac{f_{max} - I_c}{t}$
}
}
end
end
```

This process is related to task scheduling. Initializing all the virtual machines is the first process, obtain the best VM for allocating tasks depends on the load, and once VM is identified, tasks are assigned and the process is terminated by the dynamic task scheduler. To formulate this, a set of tasks $s = \{s1, s2, s2, s3, \ldots sy, \}$ is considered. The status of virtual machine utilization is monitored and observed through support vector machine classifier results. Each task has a different quality of services so that depends on the priority the utilization ratio, the tasks are allocated to the optimal VM. Based on the fitness function the tasks are assigned to the respective VM

```
$Input: n, K, d, @f, fval, b, \alpha, \beta, crmin, crmax, \sigma min, \sigma max$ $Output: Mbest, f\ it\ Mbest$
1 $begin$
2 $f\ it\ Mbest = fval;$
3 $[M, f\ it\ M, Mbest, f\ it\ Mbest] = Initialization(n, d, @f, f\ it\ Mbest);$
4 $H = M; F = M;$
5 $f\ it\ H = f\ it\ M; f\ it\ F = f\ it\ M;$ 6 $\delta 1 = \alpha K; \delta 2 = \beta K;$
7 $k = 1;$
8 $while\ k < K + 1\ do$
9 $FM = Mutation(n, d, Mbest);$
10 $CR = crmax - (crmax - crmin)k/K;$
11 $F\ C = Crossover(n, d, CR, H, FM);$
12 $f\ it\ F\ C(i) = f(F\ Ci);$
13 $f\ it\ F(i) = f(Fi);$
14 $[F, Mbest, f\ it\ Mbest] = Selection(n, F, f\ it\ F, F\ C, f\ it\ F\ C, f\ it\ Mbest);$
15 $r = -1 - k/K;$
16 $t = (r - 1) \cdot rand() + 1;$
17 $if\ k < \delta 1\ then$
18 $M = Exploration(n, d, M, F, b, t);$
19 $end$
```

```
20 else if ($k \geq \delta1$ and $k < \delta2$) then
21 $M = Hybrid(n, d, M, F, b, t, fit Mbest, fit M)$;
22 end
23 else
24 $M = Exploitation(n, d, \sigma min, \sigma max, F, @f, b, t)$;
25 end 26 for $i = 1 : n$ do
27 $fit M(i) = f(M i)$;
 28 $fit H(i) = f(H i)$;
29 end
30 $[H, Mbest, fit Mbest] = BestMoth(n, d, M, H, fit M, fit H, fit Mbest)$;
31 $k = k + 1$;
32 end
33 end 15 Journal P
```

4 Results and Discussion

In this section we evaluate the effectiveness the proposed approach is by comparing it to other existing methods. cloud computing system and application provisioning policies may be simulated with CloudSim 3.03 toolkit. This simulation toolkit is used to model the cloud components. There are no resource management or resource provisioning responsibilities for the cloud user. If implemented, it will outperform both PSO and min-max algorithms. The suggested MFO method reduces the makespan while increasing the average resource consumption rate..

Table 2. Simulation parameters

S.No	Parameter	Value
1	Number of data center	4
2	Number of Hosts	8
3	VM used	30
4	Bandwidth	3600Mbps
5	Memory capacity	20GB
6	Number of tasks	300-500

4.1 Resource utilization performance

It shows how much helpful work the resource has done in the period it has been working. Average Resource Utilization is a percentage indicator used to assess how effectively a company is utilizing its resources. It makes a comparison between the real and ideal use of virtual machines. Resources are costly and must be used optimally. This criteria is used to check the percentage time for which the resource was used. The value should be as high as possible with maximum of 100.

Average Resource Utilization $\sum_{i=1}^{n} \frac{\text{Time taken by resource i to finish all jobs}}{Makespan\ X\ n}$

(1)

Where, n is denote as the number of resources

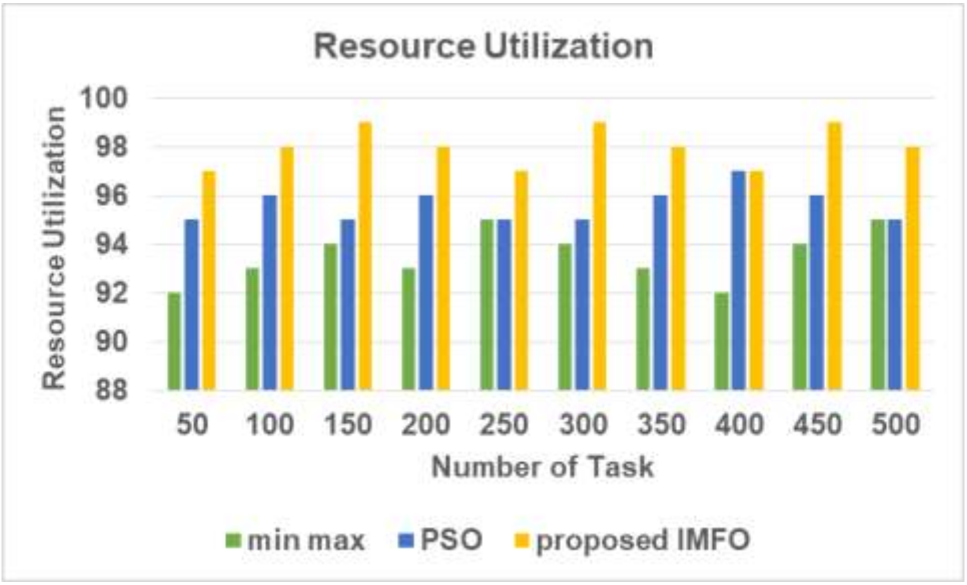

Figure 4: performance analysis of Resource utilization

Resource utilization for the proposed model for set of tasks is depicted in figure 4. Experimental observations demonstrate that proposed optimization model attains better resource utilization due to efficient VM classification. All the VM gets equal workloads which increases the resource utilization.

4.2 Makespan performance

Makespan is the total amount of time it takes to complete all of the jobs currently being worked on. It can be determined for a specific virtual machine or a specific host. The average execution time can be compared when the number of VMs is increased to determine scalability along with makespan. It is the most important optimization criteria that designates the finish time of the last task on the data centre. Makespan indicates the concluding time of the last task. It comes under the category of minimization function.

$$Makespan = maxi€tasks\ \{Fi\} \tag{2}$$

Where, Fi denotes the finishing time if task i.

Figure 5 depicts the makespan comparison for the proposed model and existing models. Makespan is calculated based on the time difference between the starting and finishing of a sequence of tasks. In the proposed model total 500 tasks are allocated to VMs and the sequence difference between the first set of tasks to the next set of tasks is 50. The proposed optimization model takes minimum time to process the sequence of tasks compared to Min-Max and PSO

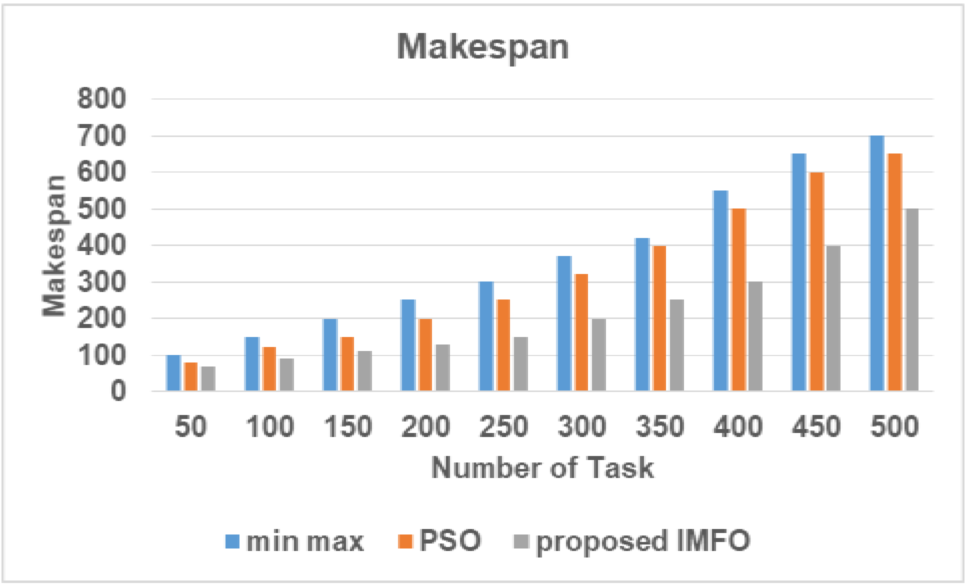

Figure 5: performance analysis of Makespan

4.3 Fitness performance

The fitness function assures that the fitness value of the infeasible solution is smaller than the fitness value of any feasible or ideal option. The higher the fitness rating, the lower the energy usage and VM migrations. The fitness of an individual x in the population of the MFO is defined in Equation (2),

$$Fitness(x) = \begin{cases} \frac{E_{min}}{E(x)} & if\ x\ is\ feasible \\ \frac{E_{min}}{E(x)+E_{max}}, & otherwise \end{cases} \tag{2}$$

Generally, crossover operator deals with two parent individuals and delivers another person. Mutation operator randomly trades the positions in a fractional child. Here, the crossover chooses the hosts with the best use from past VM mappings in view of its hybrid rate or likelihood !and mutation tries to diminish the quantity of PM required by removing one of the PM, if there are two machines with usage less than mutation rate.

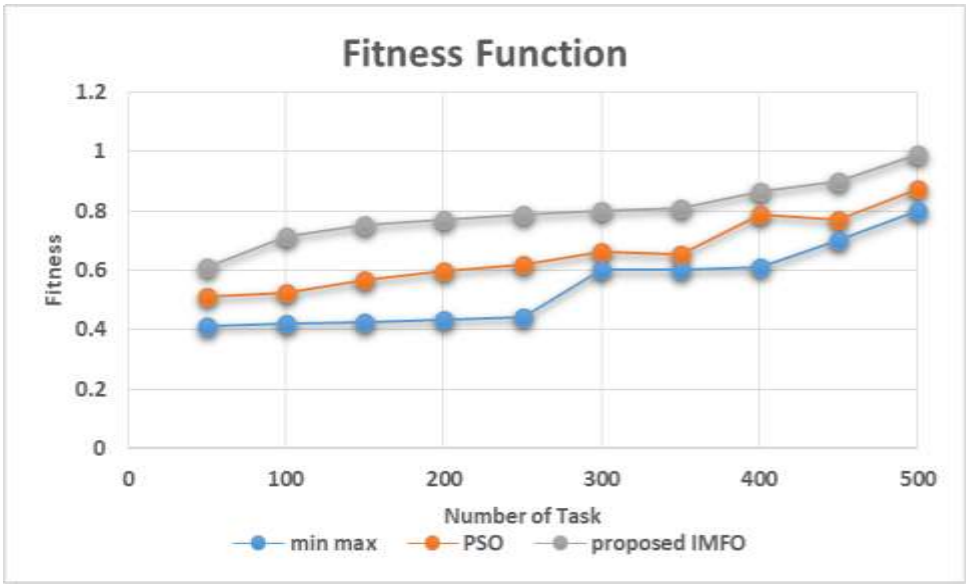

Figure 6: performance analysis of Fitness

To improve makespan and flowtime, the fitness metric is employed to calculate the performance of the scheduling method. The experimental assessment in this study is based on a value of 0.5. Figure 6 depicts the fitness value comparison findings for several heuristics. Equation 3 is used to get the fitness value. The fitness value is implicitly used to quantify makespan since when the fitness value is low, the task scheduling algorithm's makespan is similarly low.

$$Fitness = p * makespan + (1 - p) * \frac{Flowtime}{N} \quad (3)$$

where p is a value ranging from 0 to 1 dependent on the relevance of the measure and N is the number of machines For the purposes of this study, 0.5 is assumed for experimental evaluation. The fitness function for the optimization model is observed for the tasks.

The fitness probability gets varied for each task and it is observed that the proposed optimization model has a better probability ratio which indicates improved performance compared to conventional models. The VM status is already predefined by the SVM classifier which helps the optimization model to evaluate the fitness function thus, scheduling to optimal VM is achieved depends on the observed fitness functions.

4.4 Task waiting performance

The task waiting for the proposed model and other models are compared and depicted in figure 7. The time taken by the scheduler to allocate VM is calculated as waiting time and it is observed that the proposed model consumes less time to allocate resources compared to conventional models

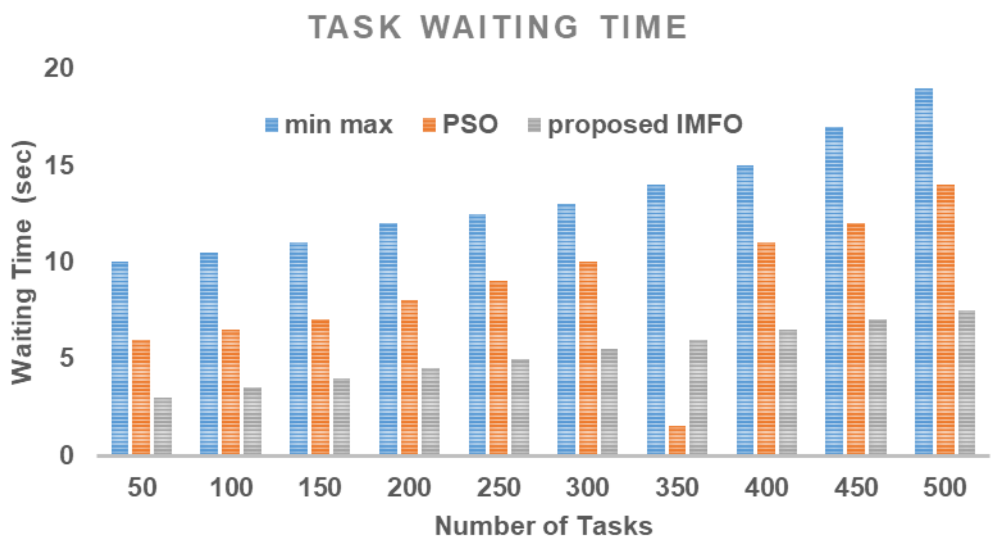

Figure 7: performance analysis of task waiting

4.5 Efficiency performance

The efficiency of the proposed task scheduling algorithm depicted in figure 8 is observed based on resource utilization, average waiting time, and average finishing time

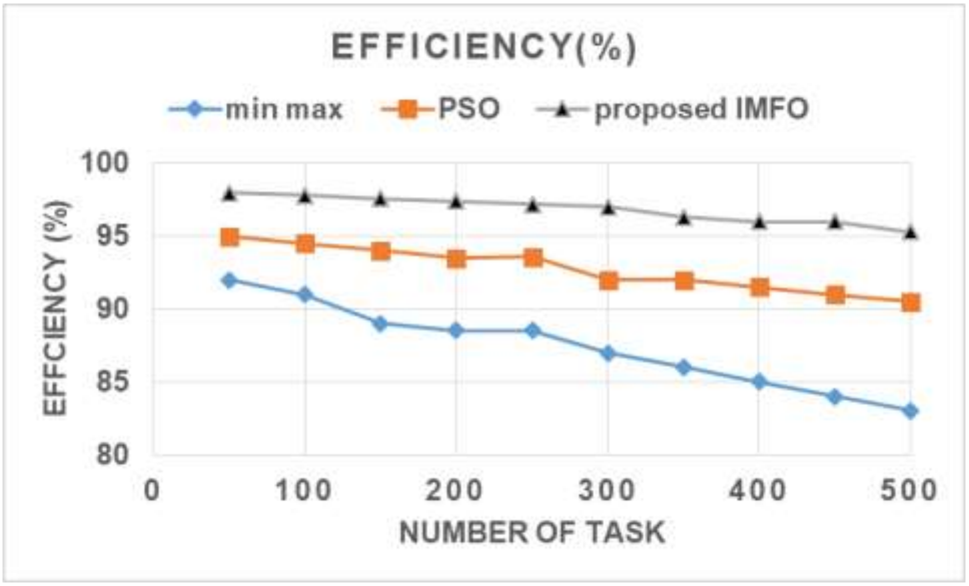

Figure 8: performance analysis of efficiency

The proposed optimization model attains better efficiency than other models. Though, the efficiency is reduced when the number of tasks is increased, however, compared to particle swarm optimization and min-max models the efficiency of the proposed optimization model is much better.

5 Conclusion

One of the most critical concerns that cloud platform providers in big data centers must consider is work scheduling in a cloud environment. The application of the appropriate solution to this problem allows cloud platform providers to make the most use of available resources while also increasing customer satisfaction by providing quality of service metrics. To address this issue, this study proposes a support vector machine-based virtual machine classifier that categorizes virtual machines based on their utilization and workload. Using improved Moth-flame optimization, this study proposes an optimized dynamic task scheduling algorithm. Traditional task scheduling algorithms assign tasks to virtual machines without analyzing their status, resulting in resource overutilization. Using moth flame optimization, tasks are assigned to the optimal virtual machine based on the virtual machine status. The proposed model is experimentally validated, and the results are compared to those of the conventional Min-Max and particle swarm optimization algorithms. The proposed optimization model outperforms the competition in terms of task completion time, task waiting time, makespan, utilization, fitness value, and efficiency.

Reference

[1]. Jghef YS, Zeebaree S. State of art survey for significant relations between cloud computing and distributed computing. International Journal of Science and Business. 2020;4(12):53-61.

[2]. Markova O, Semerikov S, Striuk A, Shalatska H, Nechypurenko P, Tron V. Implementation of cloud service models in training of future information technology specialists.

[3]. Sehgal NK, Bhatt PC. Cloud computing. Springer, Heidelberg; 2018.

[4]. Zhang F, Liu G, Fu X, Yahyapour R. A survey on virtual machine migration: Challenges, techniques, and open issues. IEEE Communications Surveys & Tutorials. 2018 Jan 17;20(2):1206-43.

[5]. Choudhary A, Govil MC, Singh G, Awasthi LK, Pilli ES, Kapil D. A critical survey of live virtual machine migration techniques. Journal of Cloud Computing. 2017 Dec;6(1):1-41.

[6]. Shaw R, Howley E, Barrett E. An energy efficient and interference aware virtual machine consolidation algorithm using workload classification. InInternational Conference on Service-Oriented Computing 2019 Oct 28 (pp. 251-266). Springer, Cham.

[7]. Kim H, Lim H, Jeong J, Jo H, Lee J. Task-aware virtual machine scheduling for I/O performance. InProceedings of the 2009 ACM SIGPLAN/SIGOPS international conference on Virtual execution environments 2009 Mar 11 (pp. 101-110).

[8]. Ahn J, Kim C, Han J, Choi YR, Huh J. Dynamic virtual machine scheduling in clouds for architectural shared resources. In4th {USENIX} Workshop on Hot Topics in Cloud Computing (HotCloud 12) 2012.

[9]. Kokilavani T, Amalarethinam DG. Load balanced min-min algorithm for static meta-task scheduling in grid computing. International Journal of Computer Applications. 2011 Apr;20(2):43-9.

[10]. Mathew T, Sekaran KC, Jose J. Study and analysis of various task scheduling algorithms in the cloud computing environment. In2014 International conference on advances in computing, communications and informatics (ICACCI) 2014 Sep 24 (pp. 658-664). IEEE.

[11]. Pilavare MS, Desai A. A Survey of soft computing techniques based load balancing in cloud computing. International Journal of Computer Applications. 2015 Jan 1;110(14).

[12]. Chirkin AM, Belloum AS, Kovalchuk SV, Makkes MX, Melnik MA, Visheratin AA, Nasonov DA. Execution time estimation for workflow scheduling. Future generation computer systems. 2017 Oct 1;75:376-87.

[13]. Pandya PP, Bheda HA. Dynamic resource allocation techniques in cloud computing. International journal of advance research in computer science and management studies. 2014 Jan;2(1).

[14]. Raghava NS, Singh D. Comparative study on load balancing techniques in cloud computing. Open journal of mobile computing and cloud computing. 2014 Aug;1(1):18-25.

[15]. Mohana RS. A position balanced parallel particle swarm optimization method for resource allocation in cloud. Indian Journal of Science and Technology. 2015 Feb;8(S3):182-8.

[16]. Ergu D, Kou G, Peng Y, Shi Y, Shi Y. The analytic hierarchy process: task scheduling and resource allocation in cloud computing environment. The Journal of Supercomputing. 2013 Jun;64(3):835-48.

[17]. Iturriaga S, Dorronsoro B, Nesmachnow S. Multiobjective evolutionary algorithms for energy and service level scheduling in a federation of distributed datacenters. International Transactions in Operational Research. 2017 Jan;24(1-2):199-228.