Int. j. inf. tecnol. (October 2022) 14(6):2951–2961
https://doi.org/10.1007/s41870-022-01045-1

ORIGINAL RESEARCH

An optimistic approach for task scheduling in cloud computing

Mehak Sharma[1] · **Mohit Kumar**[1] · **Jitendra Kumar Samriya**[2]

Received: 20 April 2022 / Accepted: 22 July 2022 / Published online: 15 August 2022

Abstract Cloud computing is gaining popularity around the globe over the last decade due to offering various types of services to end-users. Virtual machines (VM) Configuration is an imperative part of the Cloud which has turned into the focal point of attraction for many researchers nowadays. The objective of VM placement is to find an optimal physical host in the data center for deploying the VM. It becomes a challenging issue due to dynamism, uncertainty, and elasticity-based demands of end-users as well as the heterogeneous nature of cloud resources, which directly affect the quality-of-service (QoS) parameters like cost, energy consumption, utilization of resources, time, etc. Lots of VM placement techniques have been proposed in cloud computing, but the entire approach failed to achieve the optimal results in the form of QoS parameters and is unable to reduce the overhead. Hence, we have proposed an optimistic technique for VM placement that improves the mentioned QoS parameters. Further, the goal of this article is to reduce the configuration overhead in Cloud through effective scheduling techniques in which dynamically configured VMs are consolidated over fewer hosts. The experiments are conducted over the CloudSim simulation toolkit to analyze the performance of the proposed approach. The simulation results demonstrate that the proposed optimistic approach improves the QoS parameters in a significant way in comparison with other scheduling approaches like PSO-COGENT, Min-Min, and FCFS.

Keywords Cloud computing · VM placement · Virtual machine · QoS parameters · Scheduling

1 Introduction

Cloud Computing is a computing paradigm that offered on-demand, ubiquitous, continent services to end-users anywhere and anytime that were not available in other computing paradigms such as Grid computing, Cluster Computing, etc. It delivered the resources (hardware and software) in the form of services through virtualization technology [1]. Cloud users, as well as service providers, can leverage the advantage of virtualization technology that improves the utilization of cloud resources by allocating them in a multi-tenant environment. Each instance works as an individual computing machine exclusively which helps the customer to have a sense of security while dealing with sensitive information in the Cloud. The customers of the cloud can deploy their applications and use the applications installed on virtual machines by using internet technology [2]. The end users can expect QoS-aware-based service and ensure high availability, scalability, elasticity, resource pooling, etc. [3, 4]. It consists of a pool of resources that are connected in private or public networks, to give dynamically scalable infrastructure for application, data, and file storage with minimal management effort [5].

The cloud provides three types of delivery models such as infrastructure as a service (IaaS), platform as a service (PaaS) and software as a service (SaaS), and four types of deployment models public cloud, private cloud, hybrid cloud, and community cloud. End-users is benefitted from the services and models, while service provider wants to earn maximum profit from the cloud resources [6]. Organizations do not need to purchase the high computing machine,

✉ Mohit Kumar
kumarmohit@nitj.ac.in

1 Department of CSE, SLIET, Longowal, India

2 Department of IT, NIT Jalandhar, Jalandhar, India

and software resources based on their rising demand instead of that they can hire the resources from the cloud service providers on a payment basis within a few seconds. Due to the enormous benefits of cloud-based services, leading cloud service providers like Amazon, Google, Microsoft, IBM, Apple, etc. are attracting individuals to the cloud-based services [7]. Configuration of Virtual Machines is another essential aspect of the Cloud, a VM is configured by providing underline physical resources such as CPU, memory, storage and other peripheral devices based upon the requirement through the hypervisor or manually by the administrator [8].

If we go through the deep about the configuration of Virtual Machines (VMs), we would realize that VMs are made up of files that consist of records and depict the properties of VMs. These files contain the server definition, the number of virtual processors (vCPUs), random access memory (RAM), the number of system interface cards (NICs) in the virtual server, and many more [6]. Once a VM is configured, it can be powered on like a physical server, loaded with an operating system and other software solutions. Configuring VMs again and again in the Cloud to meet the demands prompts a noteworthy increment in the configuration time of VMs that leads to overhead as well as service level agreement (SLA) violations. Multiple virtual machines are deployed over the physical host based upon the demand of end-users as well as the configuration resources of the physical host. There are lots of hosts are available in a data center, hence the placement of a VM over the optimal host is always a critical issue for the service provider. The objective of the VM placement problem is to find the most appropriate physical host as per the requirement of end-users and avoid the possibility of an under-loaded or over-loaded physical host. The optimal VM placement approach optimizes the QoS parameters like energy consumption, utilization of resources, cost, and time without violating the SLA [9, 10]. Hence, we have proposed an optimistic approach for Virtual Machines Placement that improves the mentioned QoS parameters and reduces the VM migration as well as overhead without violating the SLA and other constraints in cloud computing.

The rest of the paper is organized as follows. Section 2 demonstrates related work. In Sect. 3, the system model of the proposed work has been explained. Section 4 includes the Proposed Task scheduling model for efficient VM placement. In Sect. 5, Results and discussions have been made followed by a Conclusion in Sect. 6.

2 Related work

Many algorithms have been proposed for VM placement and task scheduling in the last decade. A task scheduling strategy in the light of virtual machine matching was proposed by Zhang and Zhou [11]. The goal of the proposed approach was to schedule tasks in the minimum amount of time and to limit non-reasonable task mapping. A failure-aware virtual machine reconfiguration strategy for Cloud in the context to failures was proposed by Luo and Li Qi [12]. This work presented a failure expectation system as a proactive strategy to encourage task scheduling and virtual machine reconfiguration. The methodology proposed failure forecast strategies to decrease the potential failure effect on the quality and efficiency of the Cloud environment. Min-Min scheduling technique has been proposed to balance the workload and reduce the make-span time, but the algorithm failed to achieve it due to rescheduling the task [23]. Mohit Kumar and S. C. Sharma proposed an efficient dynamic algorithm for task scheduling based on the last optimal k-interval and improve various QoS parameters within the deadline of tasks. Further, the developed algorithm also provides the flexibility to end-user to expand or shrink (elasticity) the demand at run time [13]. Sahal and Omara proposed the article with the point to inspect the homogeneity and heterogeneity of VM configuration for the task of assignments and satisfy the perfect time and cost [14]. Saha and Hasan, a decision-making algorithm to decide the attainability to migrate a task to a cloud server for computing concentrated tasks has been proposed and actualized [15]. Hermenier et al. [16], state that clusters provide capable processing of tasks but it does not consider the dynamic nature of tasks and makes a rigid task to VM mapping. Dynamic consolidation of VMs is necessary as it helps in overcoming the overload conditions in the Cloud environment as well as helps in packing the VMs over a few numbers physical hosts to reduce the power and energy.

Past consolidation techniques for task placement consider only local optimization while neglecting the overall global optimization. The Entropy resource manager for homogeneous clusters, not only provides dynamic consolidation but also keeps in mind the migration overhead in this article. The task mapping is done in a better way by taking not only local optimization but also global optimization into 19 accounts. This paper is focused on migration overhead using the concept of entropy to reduce the migration overhead. Guo et al. [17], a delay-optimal VM scheduling algorithm is proposed that consists of resources like CPU, storage, and memory. Different kinds of VMs are allocated to upcoming demand to complete it before the deadline. A queuing model is used to handle the dynamic workloads of consumers. The basic decision process for delaying optimal VM scheduling depends upon the vector of VM configuration and the objective is also to reduce the average task completion time. Shortest job first (SJF) and min-min best fit (MMBF) scheduling algorithms are combined to make a new hybrid algorithm, but most of the proposed algorithms are unable to select the optimal host for the VM placement to fulfill the demand of end-users which leads to the problem of over or under-loaded host along with overhead. Hence, we have proposed an optimistic model to overcome the mentioned issue and find an optimal host for the VM to execute the end-user tasks in minimum time and cost without any SLA violation.

3 Problem definition and system model

This section discusses the problem definition related to VM placement at optimal host along with their important QoS parameters. Further, we will formulate the problem in mathematical format to define the fitness and objective function by considering the Cloud datacenter model.

3.1 Problem Definition

The objective of the proposed approach is to find the optimal host for deploying the virtual machine and reduce the possibility of over and under-utilization of the physical host.

3.2 Datacenter model

The Cloud datacenter D is considered as a finite set of M physical hosts represented as $D = \{PH_1, PH_2, PH_3 \ldots\ldots\ldots PH_M\}$. Each physical host PH_i has attributes with the name CPU capacity C_i(in MIPS), Memory M_i (in GB), disk storage capacity D_i (in GB), Bandwidth BW_i and working state S_i [18, 19].

$$PH_i = \{C_i, M_i, D_i, BW_i, S_i\} \text{ where } (1 \le i \le M) \tag{1}$$

In the present research work, we have considered heterogeneous types of physical hosts with different CPU capacities, and range of main memory from 1 to 16 GB for experimental purposes. Working state of host H_i. is defined by Eq. 2 (active or inactive state).

$$S_i = \begin{cases} 0 \text{ if host is inactive state} \\ 1 \text{ if host is in Active state} \end{cases} (2) \tag{2}$$

The physical host is always ready for the execution of the user's applications in active mode while the physical host is switch to a power-saving mode and needs to be activated for the execution of applications in an inactive state. In an inactive state, the n number of VMs can be deployed on the physical host H_i represented by the set $\{VM_{i1}, VM_{i2}, \ldots\ldots VM_{in}\}$ where VM_{ij} denotes jth VM deployed on ith physical host. Each virtual machine is represented with the attributes such as CPU capacity of VM_j at H_i is $C_{VM_{ij}}$ (in MIPS), memory $M_{VM_{ij}}$ (in GB), disk storage capacity $D_{VM_{ij}}$ (in GB), bandwidth $BW_{VM_{ij}}$, state $S_{VM_{ij}}$ and $ST_{VM_{ij}}$ depicts 'scheduled task' on the virtual machine VM_{ij}.

$$VM_{ij} = \{C_{VM_{ij}}, M_{VM_{ij}}, D_{VM_{ij}}, BW_{VM_{ij}}, S_{VM_{ij}}, ST_{VM_{ij}}\} \tag{3}$$

Working state of virtual machines $S_{VM_{ij}}$ can be defined (Executing or Non-Executing) by Eq. 4.

$$S_{VM_{ij}} = \begin{cases} \text{EXECUTING if executing a task} \\ \text{NON - EXECUTING if not executing a task} \end{cases} \tag{4}$$

In the Executing state, a task is scheduled and executed on a VM while in a non-executing state VM remains idle and does not execute any task. To efficiently utilize the resources of physical hosts in the data center, firstly find the optimal physical host and then deployed VMs over the host without degrading the QoS parameters.

3.3 Task model

In this section, we defined the model for end user's tasks/applications. Suppose, there are set of CPU intensive tasks are coming at cloud end for the services $T = \{t_1, t_2, t_3 \ldots\ldots\ldots t_p\}$, where each task t_k is represented as:

$$t_k = \{a_k, ts_k, d_k, s_k, E_k, f_k\} \tag{5}$$

a_k represents the 'arrival time', ts_k represents the size of the task in million instructions (MI), d_k represents the deadline of end-user's tasks, s_k is the time, when tasks start their execution over the virtual machines, E_k represents the execution time of the task and f_k represents the expected finish time of the task t_k. Concerning the CPU capacity of the VM, the execution time E_k of a task t_k running on VM_{ij} is calculated as:

$$E_k = \frac{ts_k}{C_{VM_{ij}}} \tag{6}$$

Execution time of all the tasks T is defined by Eq. 7

$$\text{Total Execution time } ET_{t_k VM_j} = \sum_{i=1}^{p} \psi_{t_k VM_j} * \frac{ts_k}{C_{VM_{ij}}}. \tag{7}$$

where $\psi_{t_k VM_j}$ is binary decision variable and its value is defined in Eq. 8 [20]

$$\psi_{t_k VM_j} = \begin{cases} 1 \text{ if } t_k \text{ is assigned to } VM_j \\ 0 \text{ if } t_k \text{ is not assigned to } VM_j \end{cases} \tag{8}$$

The expected finish time of the task t_k at the VM_j is calculated by Eq. 9.

$$f_k = a_k + E_k \tag{9}$$

The makespan (MS) time of the entire workload at the cloud end can be calculated by Eq. 10

$$MS = Max \sum_{j=1}^{n} f_{VM_j} \tag{10}$$

The objective of the proposed approach is to minimize the makespan time by efficiently utilizing the resources so that end users have to pay less amount for the services. Task guarantee ratio depicts the number of tasks effectively depart from Cloud after completion of their execution within the deadline. It is calculated using equation the 11.

$$\text{Task Guarantee Ratio}: \frac{Number\ of\ tasks\ depart\ successfully}{Total\ number\ of\ tasks} * 100 \tag{11}$$

VM Consumption represents the ratio to configured the virtual machines for processing the upcoming tasks and it is calculated using the Eq. 12.

$$\text{VM Consumption} = \frac{Number\ of\ VMs\ Configured}{Total\ number\ of\ tasks} * 100 \tag{12}$$

Task execution overhead depicts the aggregate sum of time required in seconds to configure VMs and time for tasks migration as well as task execution. It is calculated by Eq. 13.

$$\text{Task execution Overhead}: \frac{Total\ overhead}{Total\ number\ of\ tasks} * 100 \tag{13}$$

4 Proposed task scheduling model

In this research work, we present a scheduling model for the reduction of VM configuration overhead in the Cloud as shown in Fig. 1. The function of each component in the architecture is explained corresponding to the task's execution cycle as follows:

Task Receiver The Task Receiver is the initial component of the architecture that receives the upcoming requests (tasks) dynamically submitted by end-users in the Cloud environment. It has two sub-components named Arrival Queue and Type Checker. The Arrival Queue is an input buffer where the incoming tasks requests are queued for further execution and the Type Checker examines the parameters of the task like task size, its deadline, type of tasks and required resources etc. There may be different types of tasks i.e., some tasks are required high computation oriented VM, low computation oriented VM, medium computation oriented VM etc. as shown in Table 1. Hence, type 0 task required the CPU capacity less than or equal to 250 MIPS. A task of type 0 can also be scheduled on VMs

Table 1 Virtual machine classes

Class	CPU capacity (in MIPS)
1	≤ 250
2	251–500
3	501–750
4	751–1000

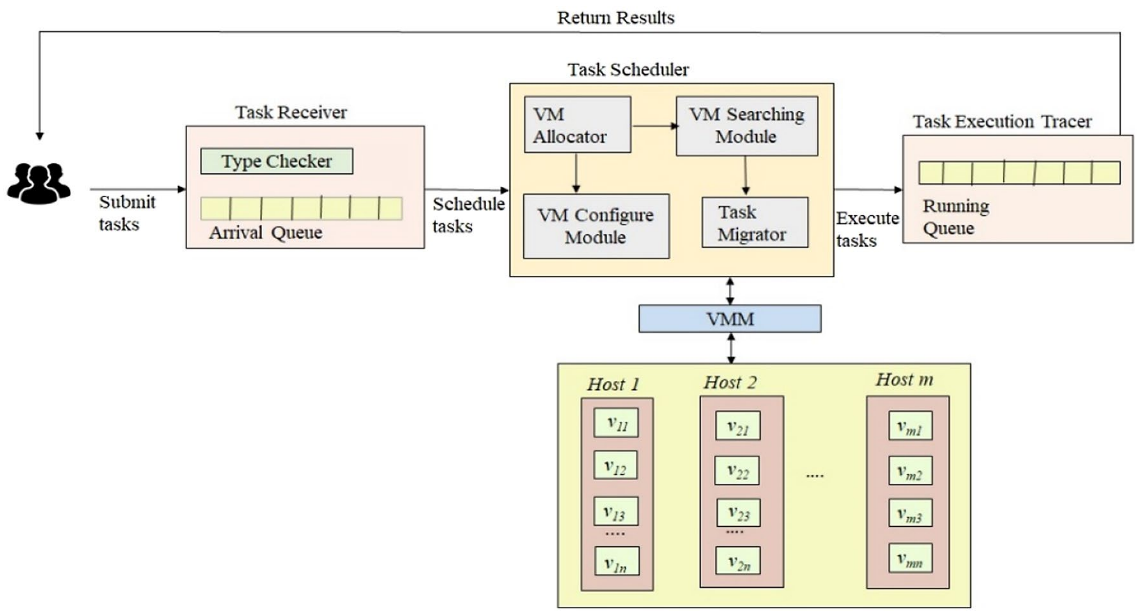

Fig. 1 Proposed Architecture for scheduling the tasks

with higher CPU capacity if low computation-oriented VM is not free for execution. Similarly, if a task has type 1, it can be scheduled on VM of class either 2, 3 or 4. A task of type 2 can be scheduled on VM of class either 3 or 4 and a task of type 3 must be scheduled on a VM of class 4 only.

Task Scheduler The aim of task scheduler is to schedule the upcoming workload over the running heterogeneous VMs for execution. It has sub-components as VM Allocator, Task Migrator, VM Configure Module, and VM Searching Module. The VM Allocator allocates the tasks to the optimal VM instance for the execution with the help of other sub-components. First, it is checked if any pre-configured VM is exist for the execution of tasks when it arrived. If no pre-configured VM is found, then, VM Allocator calls to VM Configure Module and VM Searching Module to search and configure the optimal VM according to the requirements of the task. VM searching module searches pre-configured VMs in decreasing order of their resource capacity. Upcoming task request is mapped with the available running VMs and find the optimal resource for the execution of tasks. In this way, VM Searching Module finds the optimal virtual machines for entire upcoming requests. If any pre-configured VM is found in idle condition and capable of meeting the task's requirements within the deadline, then, the task will be scheduled on this VM and further searching is stopped. If no pre-configured VM is found during the search which is both idle and capable of meeting the task's requirements, then, VM Searching Module calls Task Migrator to carry out further processing. The task will now enter the Migration queue.

Task Migrator performs the migration to schedule the task on pre-configured VMs which are non-idle, and it follows the two main steps as:

- Non-idle VMs (VMs on which tasks are already scheduled and state as "EXECUTING") are searched in decreasing order of their CPU capacity. While searching non-idle VMs, if any VM is found capable of meeting the task's requirements within the deadline, then, the currently executing task on this non-idle VM (termed as Source VM) needs to be migrated to some other VM which is idle.
- Idle VMs (No tasks are scheduled and state as "NON-EXECUTING") are searched which could execute the remaining size of currently executing a task on the Source VM within the deadline. If such VM is discovered (termed as Destination VM), then, the task currently executing on the Source VM will be migrated to the Destination VM and the task (to be executed) will be scheduled on the Source VM. The task (to be executed) will now enter the Scheduled queue.

Task Migrator makes space for the tasks on pre-configured non-idle VMs by migrating their tasks to other suitable and idle VMs with the goal to configure another VM for the task. The entire procedure of migrating a task from Source VM to Destination VM takes approximately 2 s [21]. VM Allocator will call the VM Configure module to configure a VM for the task execution. Before configuring the VM, all active hosts have been arranged accordingly to BHF (Busiest Host First) strategy in which the hosts are sorted in increasing order of their Remaining CPU Capacity [22]. This will help in configuring VM over the host first that has a noteworthy usage of its resources and attempting to consolidate VMs over a smaller number of active hosts and utilizing the resources of hosts in a greater degree.

VM Configure Module configures a VM with the help of Virtual Machine Monitor (VMM) that provides underline physical resources to a VM according to its requirements. If a VM is successfully configured on an active host arranged by BHF strategy, then, the task will be scheduled on this VM, otherwise, an Inactive host is waked up and it takes approximately 90 s.

Task Execution Tracer A task enters in the running queue in a FIFO manner and its execution initiates with the help of Task Execution Tracer on Cloud. The tasks' expected finish time is computed and the tasks are permitted to depart at their expected finish time. Task's inter-arrival Time depicts the average time of the next task arrival in the Cloud. Tasks' Inter-Arrival Time is calculated as:

$$\text{Inter} - \text{Arrival Time (IAT)} = \sum_{k=1}^{p} \frac{t_{k+1} - t_k}{p-1} \tag{14}$$

Since the value of tasks' Inter-Arrival Time can change, subsequently, its value is calculated after a regular interval of time. After being configured for the task's inter-arrival Time, if it is discovered, then some other tasks are now scheduled on the idle VM that are capable of meeting its requirements during the task's inter-arrival Time. Flow chart of the proposed approach is shown in Fig. 2.

Proposed Algorithm Algorithm 1 describes the steps required for scheduling tasks in Cloud. Every task is submitted dynamically by end-users. Step 3–5 of the proposed approach represents the configured VMs/VM exists or not. If the condition is true, then VM Search function is called otherwise VM is configured according to the requirements of the task using VM deploy function in step 6. In steps 7–10, it is checked if the expected finish time of task ft_k is less than or equal to the deadline of task d_{t_k}. If the condition is true, then task t_k successfully departs otherwise fails. After execution of the task over the VM, it remains configured for tasks' Inter-Arrival Time represented as IAT in step 11. After being configured for IAT VM undeploy function is called in step 12.

Algorithm1 Function task scheduling ()

```
flag1← FALSE, flag2← FALSE
1.  for each task $t_k$ do
2.      Add $t_k$ to arrival queue;
3.      if VM configure! = NULL
4.          Call VM Search ();
5.      elseif flag1==FALSE && flag2==FALSE
6.          Call VM deploy
7.  if $f_k$ <= $dt_k$
8.      $t_k$ successfully departs;
9.  else
10.     $t_k$ fails;
11. VM remain configured for IAT;
12. Call VM undeploy ();
```

Algorithm 2 describes the steps involved in searching a configured VM suitable for the execution of the task t_k submitted by the end users. If a VM_j is found idle (*NON-EXECUTING*) and the capacity of VM ($C_{VM_{ij}}$) is greater than or equal to the capacity required by the task t_k. The task t_k is scheduled on this VM and added in the scheduled queue. If the status of a VM is found *EXECUTING* (which means some tasks, say t_m, is already scheduled over VM) and capacity of VM ($C_{VM_{ij}}$) is greater than or equal to the capacity required by the task t_k, then, VM will work as v_{source}. Further, VMs over active hosts have searched again. If another VM is found idle during the search and the capacity of VM_{ij} is greater than or equal to the capacity required by task t_m scheduled over non-idle v_{source}. The task t_m is migrated to VM $v_{destination}$ and the task t_k is scheduled over Virtual

Algorithm 2 Function VM Search()

```
1.  for every host $PH_i$ do
2.    if $S_i$== ACTIVE
3.      for each Virtual Machine $VM_j$ on host $PH_i$ do
4.        Search $VM_{ij}$ decreasing order of $C_{VM_{ij}}$
5.          if $S_{VM_{ij}}$== NON-EXECUTING && $C_{VM_{ij}} >= req_{t_k}$
6.            Schedule $t_k$ over the virtual machine $VM_j$;
7.            Add $t_k$ to scheduled queue;
8.            flag1← TRUE;
9.                break; end if, end for
10.     end if, end for
11. if flag1==FALSE
12.         for each host $PH_i$ do
13.             if $S_i$= =ACTIVE
14.         for each $VM_j$ search in decreasing order of capacity $C_{VM_{ij}}$
15.                 If $S_{VM_{ij}}$= EXECUTING && $C_{VM_{ij}} >= req_{t_k}$
16.                   $V_{source}$←$VM_{ij}$
17.                   break; end if, end for
18.           end if, end for
19. for each host $PH_i$ do
20.     for each $VM_j$ on host $PH_i$ do
21.         if $S_i$== ACTIVE &&$S_{VM_{ij}}$== NON-EXECUTING && $C_{VM_{ij}}$= $rem_{t_m}$
22.           $V_{destination}$←$VM_{ij}$
23.           Migrate task $t_m$ scheduled on $V_{source}$ to $V_{destination}$
24.           $st_{v_{source}}$← $t_k$;
25.           flag2← TRUE;
26.           Add $t_k$ to scheduled queue;
27.           break;   end if, end for, end for
28.     if flag1==TRUE || flag2==TRUE
29.               Execute $t_k$ and calculate $f_k$;
30.               Add $t_k$ to running queue;
31.               $s_{v_{destination}}$← EXECUTING;
32.               $st_{v_{destination}}$← $t_m$ ;
33.       end if
```

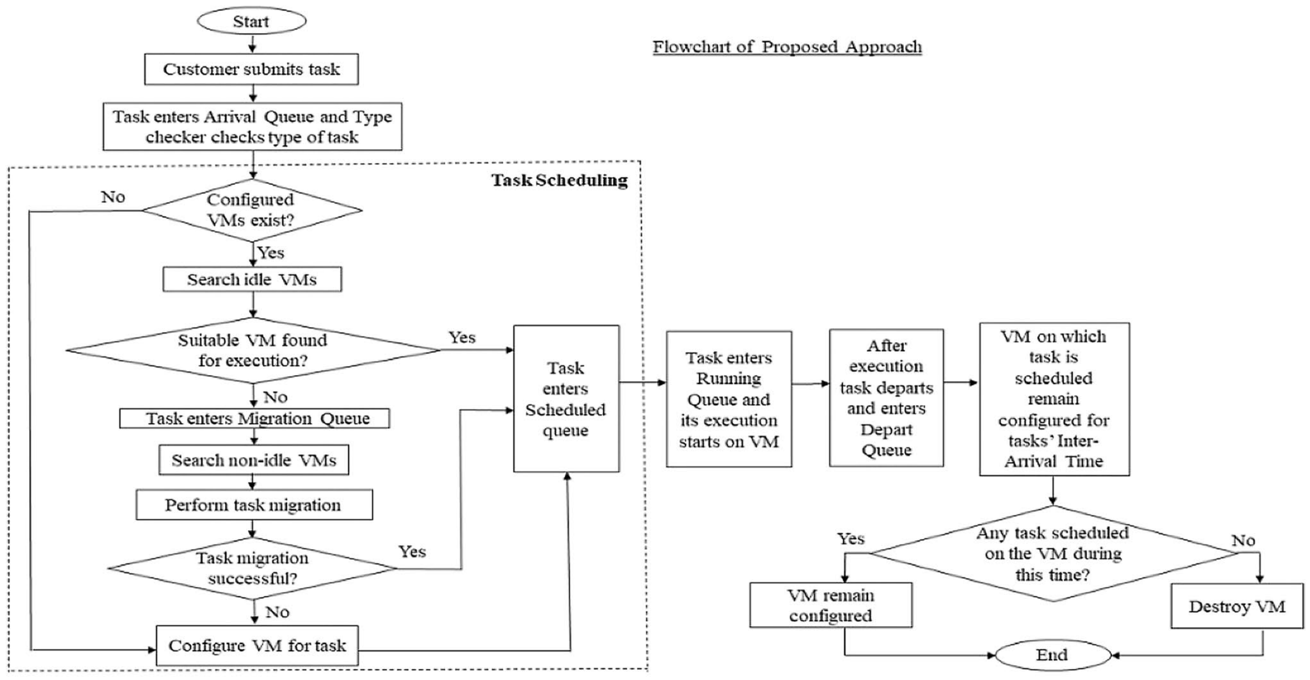
Flowchart of Proposed Approach
Start
Customer submits task
Task enters Arrival Queue and Type checker checks type of task
Task Scheduling
Configured VMs exist?
No
Yes
Search idle VMs
Suitable VM found for execution?
Yes
No
Task enters Migration Queue
Search non-idle VMs
Perform task migration
Task migration successful?
Yes
No
Configure VM for task
Task enters Scheduled queue
Task enters Running Queue and its execution starts on VM
After execution task departs and enters Depart Queue
VM on which task is scheduled remain configured for tasks' Inter-Arrival Time
Any task scheduled on the VM during this time?
Yes
No
VM remain configured
Destroy VM
End

Fig. 2 Flowchart of Proposed Approach

Machine v_{source}. In addition, the task is migrated from v_{source} to $v_{destination}$. The task t_k is added to the scheduled queue as it gets a VM for execution during task migration. If $flag1 == TRUE$ which means an idle VM suitable for the execution of the task t_k is found or $flag2 == TRUE$ which states that a VM is found for the execution of the task t_k during task migration, the task t_k is added to the running queue and task t_m migrated to $v_{destination}$ will also start its execution and status of VM $v_{destination}$ is set to *EXECUTING*. Algorithm 3 depicts the function VM deploy (), and algorithm 4, VM undeploy () function.

Algorithm 3. Function VM Deploy()

```
1.  for every host $PH_i$ do
2.    If $S_i == ACTIVE$ && $C_{VM_j} >= req_{t_k}$
3.      Configure $VM_j$ over $PH_i$;
4.      else
5.        If $S_i == INACTIVE$
6.          $S_i \leftarrow ACTIVE$;
7.        Configure $VM_j$ over $PH_i$;
8.          end if
9.    Add $t_k$ to scheduled queue;
10. end for
11. Execute $t_k$ and add $t_k$ to running queue;
12. calculate $f_k$;
```

Algorithm 4. Function VM Undeploy()

```
1.  for every host $PH_i$ do
2.    If $S_i == ACTIVE$
3.      for each $VM_j$ over $PH_i$ do
4.        If $st_{VM_{ij}}$==NULL&&($dt_{VM_{ij}}$+ IAT)== current time
5.          Destroy $VM_{ij}$;
6.        else
7.          $VM_{ij}$ remain configured;
8.        end for, end if
9.  end for
```

5 Results and discussion

In this section, initially, we will discuss the experimental environment, where we have tested the proposed algorithm as well as other state-of-art algorithms. After that, the performance of the proposed approach is compared with existing state-of-arts algorithms.

5.1 Experimental environment

We will test the performance of the developed optimistic technique and its accuracy by correlating it with existing several techniques. The performance evaluation of experimental outcomes is done by using the Cloudsim simulation toolkit and Eclipse IDE platform. All the above-mentioned experimental analysis and development work is performed at the Intel Xeon Platinum 8180 M (2.0 GHz) and 16 GB RAM capacity with 64 bits windows-10 operating system. Here, we have created multiple datacenters to find out the experimental outcome of the proposed algorithm. We can deploy multiple physical hosts over a datacenter based upon the resource configuration of the datacenter.

5.2 Results and discussion about QoS parameters

This section demonstrates the results of experiments performed on the CloudSim toolkit for assessing different parameters, like makespan time, Guarantee Ratio, VM Consumption, energy consumption, and Task Execution Overhead.

Makespan Time: We have generated synthetic workloads that have different types of tasks and they required different processing speeds for the execution. The virtual machines in our simulation have heterogeneous processing power and running over an optimal host. To assess, the performance of the developed algorithm, the end user's workload is allocated to the running host and calculated the makespan time of the developed algorithm. We have considered two cases to assess the performance.

Case 1: We have deployed 500 heterogeneous virtual machines over the physical host and started to allocate the workload based upon the proposed algorithm. Initially, the end user's workload consists of 1000 diverse tasks and their requirement varies in terms of computation resources. After allocation of workload to 500 VMs by developed algorithm, makespan time of VMs is calculated. The same synthetic workload is generated for the other state-of-arts algorithms and allocated to the virtual machines. After the simulation experiment, Fig. 3 shows that the proposed optimistic algorithm provides the optimal makespan time as compared with others scheduling approaches such as PSO-COGENT [20], Min-Min [23], first come first serve (FCFS) [24]. Further, we have increased the number of tasks from 1000 to 4000 and, simulation-based results are (shown in Fig. 3) verified that the proposed optimistic approach is much better than other scheduling policies.

Case 2: We have generated the synthetic workload that consists of 4000 diverse tasks and vary the number of cloud resources (VMs) from 300 to 1000 that are deployed over the physical host. Initially, the proposed algorithm allocates the workload among the 300 VMs to perform the execution. The same scenario is created for other scheduling algorithm to calculate the makespan time. Figure 4 shows that the proposed optimistic algorithm provides the optimal makespan time as compared with others scheduling approaches such as PSO-COGENT [20], Min-Min [23], FCFS [24]. Further, we have increased the number of VMs from 300 to 1000 and simulation-based results are (shown in Fig. 4) verified that the proposed optimistic approach is much better than other scheduling policies.

Task guarantee ratio It is calculated using the Eq. 11, where ratio of departed task and total number of submitted task is calculated in the simulation environment. To calculate the task guarantee ratio in the simulation environment, we have generated synthetic workload (diverse tasks) along with the deadline and assigned to running virtual machines using the proposed approach as well as the existing scheduling approach. Calculated results shown in Fig. 5 proved that the proposed approach provides a better task guarantee ratio as compared with other scheduling approaches like PSO-COGENT [20], Min-Min [23], FCFS [24]. Further, as we increased the number of tasks in the simulation environment, the task guarantee ratio slightly decreases as shown in Fig. 5.

VM Consumption: VM Consumption portrays the number of virtual machines configured over the physical host to execute the tasks. It is calculated by Eq. 12. Synthetic workload (diverse tasks) has been generated to calculate the consumption of virtual machines using the proposed algorithm as well as other scheduling approaches. As Fig. 6 demonstrates the quantity of VMs configured to execute tasks in the proposed approach are roughly 50–60% of the number utilized in the existing methodology, while other approaches

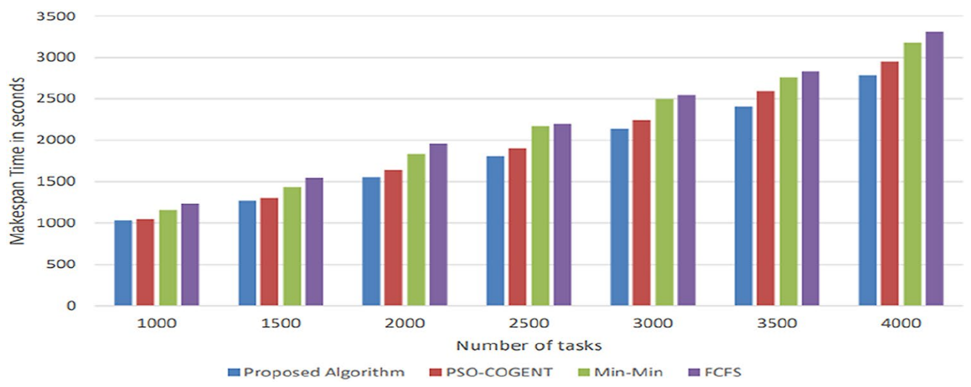

Fig. 3 Makespan time comparison at fixed VMs

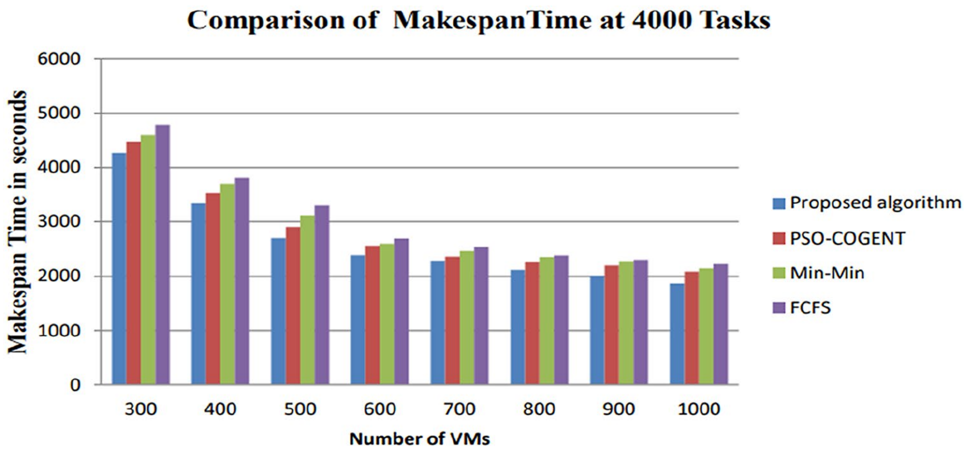

Fig. 4 Makespan time comparison at fixed Tasks

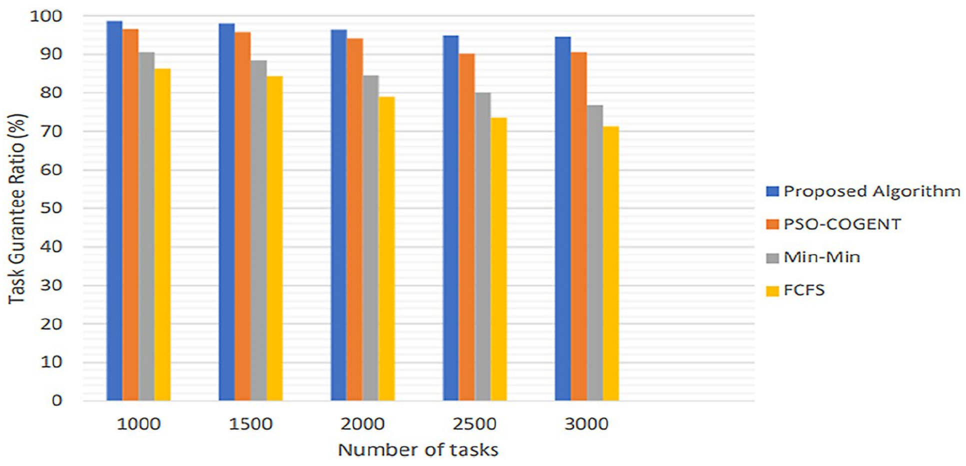

Fig. 5 Task guarantee ratio

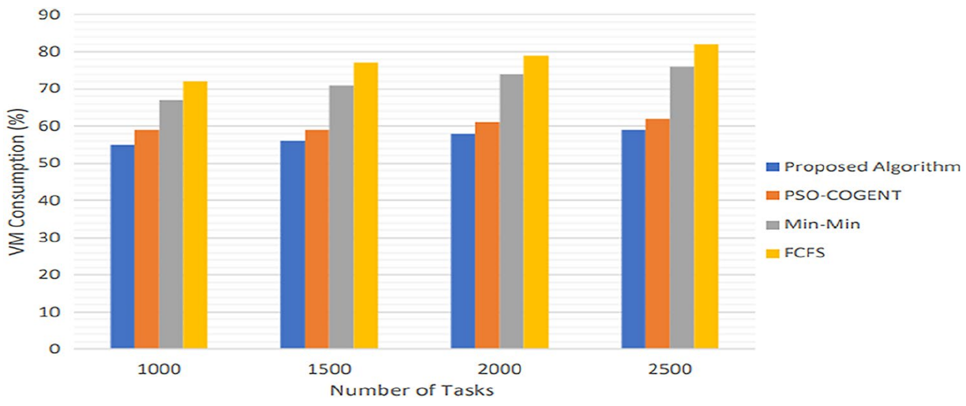

Fig. 6 Consumption of VMs

have approximately 60–80%. The proposed scheduling approach has helped in an incredible way to reduce the quantity of VMs configured to execute tasks.

Task execution Overhead It is the ratio of total overhead with number of submitted requests and calculated by Eq. 13. In the existing approach, task execution overhead is examined, VM configuration, migration overhead and execution are considered because lots of factor affects the task migration during the scheduling. We have generated the number of tasks and processed over the 500 heterogeneous virtual machines using the proposed approach as well as other scheduling approaches like PSO-COGENT [20], Min-Min [23], FCFS [24]. Simulation results show in Fig. 7 represents that the overhead of the proposed approach is less than the other scheduling approaches. The migration and configuration overhead both consolidated in the proposed

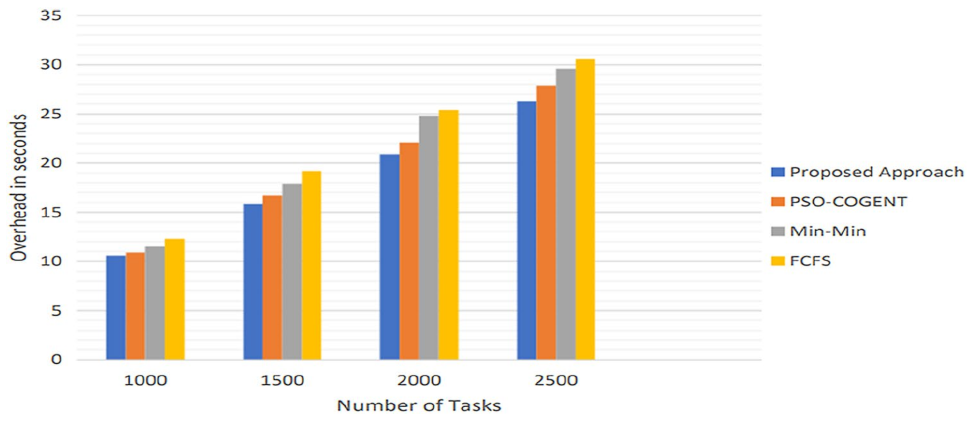

Fig. 7 Tasks overhead of proposed approach and other scheduling approach

approach and the total overhead is roughly half of the overhead in the existing approach while serving customers' requirements dynamically.

6 Conclusion

Optimal virtual placement is one of the interesting topics in the field of the cloud because lots of performance parameters depend upon it. In the existing work, a large number of VMs are configured to serve the requirements of the end-users. Configuring a VM takes a lot of time, which leads to degrade the performance. Hence, we have used an optimistic approach to overcome the above-mentioned issues and reduce the overhead as well as makespan time of end users' tasks. Further, the proposed approach used a productive scheduling strategy to schedule tasks over VMs, and task migration is utilized that helped incredibly to decrease the quantity of VMs for the execution of tasks in a dynamic environment. Also, tasks are attempted to schedule effectively over the pre-configured VMs with the goal that the configuration overhead of VMs can be reduced and avoid the possibility of over or under-utilization of the physical host. To assess the performance of the proposed approach, we have used Cloudsim toolkit 3.03 with synthetic workload. Experimental results are shown in Figs. 3, 4, 5, 6 and 7 proved that our approach of scheduling tasks has helped in an incredible way to improve the QoS parameters. In the future, we will apply the proposed approach to calculate other QoS parameters like reliability, energy consumption, availability, etc. We can apply the machine learning based approach to predict the upcoming workload, to allocate the cloud datacenter resources in more optimal fashion.

Author contributions MS: writing—original draft, results and outcome. MK: software, experimental/simulation work, and Visualization. JKS: conceptualization and methodology, review & editing.

Funding No funds, grants, or other support was receive for this research work.

Data availability The synthetic datasets has been generated by author's to evaluate the performance of the proposed algorithm that can be provided after the request.

Declarations

Conflict of interest The authors have no conflicts of interest to declare that are relevant to the content of this article.

Human and/or animals participants The article did not involve human and/or animals participants.

References

1. Kumar M et al (2019) A comprehensive survey for scheduling techniques in cloud computing. J Netw Comput Appl 143:1–33
2. Mell PM, Grance T (2011) The NIST definition of cloud computing. National Institute of Standards and Technology: U.S. Department of Commerce, Gaithersburg, MD
3. Kumar M et al (2017) Elastic and flexible deadline constraint load Balancing algorithm for Cloud Computing. Proc Comput Sci 125:717–724
4. Gill SS et al (2018) CHOPPER: an intelligent QoS-aware autonomic resource management approach for cloud computing. Cluster Comput 21(2):1203–2124
5. Buyya R et al (2009) Cloud computing and emerging IT platforms: Vision, hype, and reality for delivering computing as the 5th utility. Future Gener Comput Syst 25(6):599–616
6. Masdari M, Nabavi SS, Ahmadi V (2016) An overview of virtual machine placement schemes in cloud computing. J Netw Comput Appl 66:106–127
7. Chen W, Qiao X, Wei J, Huang T (2012) A two-level virtual machine self-reconfiguration mechanism for the cloud computing platforms. In: Proc.—IEEE 9th Int. Conf. Ubiquitous Intell. Comput. IEEE 9th Int. Conf. Auton. Trust. Comput. 50 UIC-ATC 2012, pp 563–570
8. Do AV, Chen J, Wang C, Lee YC, Zomaya AY, Zhou BB (2011) Profiling applications for virtual machine placement in clouds. In: Proceedings of the 4th IEEE international conference on cloud computing, pp 660–667

9. Gupta MK, Jain A, Amgoth T (2018) Power and resource-aware virtual machine placement for IaaS cloud. Sustain Computi Inf Syst 19:52–60
10. Jiang H-P, Chen W-M (2018) Self-adaptive resource allocation for energy-aware virtual machine placement in dynamic computing cloud. J Netw Comput Appl 120:119–129
11. Zhang P, Zhou M (2017) Task scheduling based on virtual machine matching in clouds. In: Proceedings of the 13th IEEE Conference on Automation Science and Engineering (CASE), pp 618–623
12. Luo Y, Qi L (2012) Failure-aware virtual machine configuration for cloud computing. In: Proc.—2012 IEEE Asia-Pacific Serv. Comput. Conf. APSCC 2012, p. 125–132
13. Kumar M, Sharma SC (2018) Deadline constrained based dynamic load balancing algorithm with elasticity in cloud environment. Comput Electr Eng 69:395–411
14. Sahal R, Omara FA (2015) Effective virtual machine configuration for cloud environment. In: 2014 9th Int. Conf. Informatics Syst. INFOS 2014, pp PDC15–PDC20
15. Saha S, Hasan MS (2017) Effective task migration to reduce execution time in mobile cloud computing. In:ICAC 2017—2017 23rd IEEE Int. Conf. Autom. Comput. Addressing Glob. Challenges through Autom. Comput., no. September, pp 7–8
16. Hermenier F, Lorca X, Menaud JM, Muller G, Lawall J (2009) Entropy: a consolidation manager for clusters. In: Proceedings of the 2009 ACM SIGPLAN/SIGOPS international conference on Virtual execution environments, pp 41–50
17. Guo M, Guan Q, Ke W (2018) Optimal Scheduling of VMs in Queueing Cloud Computing Systems with a Heterogeneous Workload. IEEE Access 6:15178–15191
18. Zhu X, Yang LT, Chen H, Wang J, Yin S, Liu X (2014) Real-time tasks oriented energy-aware scheduling in virtualized clouds. IEEE Trans Cloud Comput 2(2):168–180
19. Samimi P, Teimouri Y, Mukhtar M (2016) A combinatorial double auction resource allocation model in cloud computing. Inf Sci (NY) 357:201–216
20. Kumar M, Sharma SC (2018) "PSO-COGENT: Cost and Energy Efficient scheduling in Cloud environment with deadline constraint. Sustain Comput Inf Syst 19:147–164
21. Shamsinezhad E, Shahbahrami A, Hedayati A, Zadeh AK, Banirostam H (2013) Presentation methods for task migration in cloud computing by combination of Yu Router and Post-copy. Int J Comput Sci 10(4)
22. Wang J, Bao W, Zhu X, Yang LT, Xiang Y (2015) FESTAL: fault-tolerant elastic scheduling algorithm for real-time tasks in virtualized clouds. IEEE Trans Comput 64(9):2545–2558
23. Chen et al (2013) User-priority guided min-min scheduling algorithm for load balancing in cloud computing. In: National Conference on Parallel Computing Technologies., Bangalore., KA, 2013, pp 1–8
24. Li W, Shi H (2009) Dynamic load balancing algorithm based on FCFS. In: Innovative computing, information and control (ICICIC), 2009 Fourth International Conference on. IEEE, 2009, pp 1528–1531