A Multi-objective Virtual Machine Scheduling Algorithm in Fault Tolerance Aware Cloud Environments

Heyang Xu[1(✉)], Pengyue Cheng[1], Yang Liu[1], Wei Wei[1], and Wenjie Zhang[2]

[1] Henan University of Technology, Zhengzhou 450001, Henan, China
xuheyang124@126.com, chengy_cathy@163.com,
liu_yang@haut.edu.cn, nsyncw@126.com
[2] Information Engineering University, Zhengzhou 450001, Henan, China
xy_zwj@sohu.com

Abstract. In modern cloud datacenters, virtual machine (VM) scheduling is a complex problem, especially taking consideration of the factor of service reliability. Failures may occur on physical servers while they are running cloud users' applications. To provide high-reliability service, cloud providers can adopt some fault tolerance techniques, which will influence performance criteria of VM scheduling, such as the actual execution time and users' expenditure. However, only few studies consider fault tolerance and its influence. In this paper, we investigate fault tolerance aware VM scheduling problem and formulate it as a bi-objective optimization model with quality of service (QoS) constraints. The proposed model tries to minimize users' total expenditure and, at the same time maximize the successful execution rate of their VM requests. The both objectives are important concerns for users to improve their satisfactions, which can offer them sufficient incentives to stay and play in the clouds and keep the cloud ecosystem sustainable. Based on a defined cost efficiency factor, a heuristic algorithm is then developed. Experimental results show that, indeed, fault tolerance significantly influences some performance criteria of VM scheduling and the developed algorithm can decrease users' expenditure, improve successful execution rate of their VM requests and thus perform better under fault tolerance aware cloud environments.

Keywords: VM scheduling · Cloud computing · Fault tolerance · QoS · Users' expenditure

1 Introduction

In modern cloud environments, providers can offer their customers the opportunities to configure service requests with specific resource requirements, such as hardware and software resource, and then encapsulate all these resource together into virtual machines (VMs) [1]. By renting VMs from cloud providers and uploading their computing requests to cloud data centers, customers can conveniently access and manage their applications

X. Zhang et al. (Eds.): CloudComp 2019/SmartGift 2019, LNICST 322, pp. 529–543, 2020.
https://doi.org/10.1007/978-3-030-48513-9_42

from anywhere at any time. They no longer need to purchase and maintain the sophisticated hardware and software resources for their peak loads and thus can decrease their total cost of ownership [2, 3]. Thus cloud computing has now become one of the most popular information communications technology paradigms and is widely accessed by nearly all internet users in direct or indirect manners [4].

How to allocate each of the cloud users' VM request to an appropriate cloud resource is the central issue involved in VM scheduling under cloud computing environments. This is a complicated and changeling problem, in which different performance criteria should be optimized and at the same time, users' QoS requirements (e.g. deadline and budget) should be satisfied [5]. For cloud users, they generally want their service request are successfully completed (i.e., their respective deadlines and budgets are fulfilled) with the possible minimal expenditure. Conversely, if their service requests often miss the deadlines, or the incurred expense is high, then users will undergo low level of satisfaction degree and may lose interest in the cloud system and could finally leave it, which will be adverse to the sustainable cloud ecosystem [3]. Therefore, this paper mainly focuses on two major performance criteria of VM scheduling, i.e., successful execution rate of users' VM requests (SERoV) [3, 6] and the total expenditure (TE) that users need to spend on completing users' requests [7–9].

Moreover, another important concern in real-world cloud datacenters is that failures may occur on physical servers (PSs) while executing cloud users' VM requests. This may also depress the level of users' satisfaction degree (LoUSD) [9]. So in order to increase the LoUSD, cloud providers generally adopt some fault tolerance techniques in their data centers to improve the offered service reliability, such as fault recovery. Fault recovery employs check-pointing and roll-back schemes and can enable a failed VM to recover and resume the executing of the VM from an error [10]. However, it will in turn influence the considered performance criteria in VM scheduling, such as SERoV and TE, which is worthy of further research.

Many researches recently have investigated VM scheduling under cloud environments, from which mostly focusing on optimizing different performance criteria of different parties, such cloud providers and users (details are given in Sect. 2). For example, Zhang et al. [8], from users' standpoint, explored the cost-efficient VM scheduling with the endeavor of minimizing the total expenditure for executing their VM requests. For adjusting to the changing resource demands, Meng and Sun [11] mainly considered the situation of workload fluctuations and developed a feedback-aware resource scheduling algorithm based on the resource granularity of containers. In [12], Yu et al. tried to optimize some criteria from cloud providers' aspect and proposed a probability-based guarantee scheme, which can effectively decrease the total migration overhead by avoiding unnecessary VM migrations. In previous work [3], we formulated the VM scheduling problem as a multi-objective optimization model by considering the major concerns of both parties, namely minimizing users' total expenditure and at the same time, guaranteeing the profit fairness among all the cloud providers. Some useful approaches have been proposed to resolve VM scheduling in cloud computing. However, most of these studies are failed to consider the influences of resource failures and their recoveries, which leads to a dilemma that the proposed approaches can't applied to the real-world cloud data centers. Therefore, this paper further studies the VM scheduling problem by

taking into account the influences of resources failures and their recoveries, which can be denoted by fault tolerance aware VM scheduling problem.

The rest of the paper is organized as follows. Section 2 reviews the related state-of-the-art studies on VM scheduling. Section 3 describes the fault tolerance aware VM scheduling problem in details. In Sect. 3, we formulate the studied problem as a multi-objective optimization model. The developed algorithm is given in Sect. 4. Section 5 presents the experimental configurations, results and analyses. Finally, Sect. 6 concludes this paper.

2 Related Work

Lots of efforts were focused on the research on VM scheduling in clouds and many partly feasible solutions were also proposed. For example, Wei et al. [13] aimed to improving the resource utilization of cloud data center's physical servers and developed a heterogeneity-based VM scheduling algorithm whose main idea is to guarantee that the workloads on different kinds of resources of a PS are balanced. In [14], Imai et al. adopted continuous air traffic optimization method to optimize VM scheduling and proposed a time-series prediction based VM scheduling algorithm. The proposed algorithm can effectively reduce the total running time of used VMs while achieving similar performance. In [15], Secinti and Ovatman studied energy optimization of cloud data centers by minimizing the number of VM migrations and service level agreement violations under fault tolerance aware cloud environments. Moreover, many other researches [16–18] also explored power consumption aware VM scheduling problem and tried to find the best solutions for cloud datacenters' energy optimization. These researches, from cloud providers' standpoint, explored VM scheduling problem by optimizing different performance criteria.

Some related works studied VM scheduling from cloud users' respective and made endeavors to optimize their major concerns. For example, Wang et al. [19] tried to decrease the response time of users' applications with high performance computing requirements and developed a synchronization aware VM scheduling algorithm by considering both intra-VMs' and inter-VMs' synchronization demands. In [20], Kohne et al. further studied service level agreement (SLA) aware VM scheduling problem by considering two service level objectives, i.e., resource usage and availability. Zeng et al. [21] studied workload-aware VM scheduling problem with the objectives of minimizing the network latency and maximizing the bandwidth utilization. Some other effective execution time aware solutions, such as [22, 23], are also proposed for scheduling VMs in cloud data centers. Nevertheless, in cloud environments, another important concern for users to execute their VM requests is the total expenditure, which is not well resolved in these studies.

For reducing users' execution cost, some researches explored cost-based approaches to address the problem [3, 8, 24–26]. In [24], Li et al. tried to cut down users' expenditures for renting the required computing resources under hybrid cloud environments and developed an online resource scheduling algorithm by adopting Lyapunov optimization framework. A particle swarm optimization (PSO)-based scheduling algorithm is developed in [25] with the aim of minimizing users' total cost for executing their submitted

VM requests. From the perspective of reducing the number of VMs rented from cloud providers, Ran et al. [26] developed a dynamic VM scheduling strategy to determining the number of rented VMs according to users' varied QoS requirements. In [27], Sotiriadis et al. discovered that resource usages of running VMs vary highly among their whole runtime by extensive experiments. Based on this discovery, they developed a performance aware VM scheduling algorithm.

It can be found that although related studies have explored many aspects of performance criteria of VM scheduling, one of the most important factors for both cloud providers and users, i.e. service reliability, is not well addressed. In order to improve service reliability, modern cloud data centers generally adopt fault recovery techniques, which in turn will influence the performance criteria of VM scheduling, i.e., increasing the actual runtimes of users' VM requests and inevitably raising users' expenditures. This influence is worthy of deep research. However, only few of existing studies explored this influence. For example, Sun et al. [28] investigated the tradeoff between performance and energy consumption when reliability factor is considered. Our recent work [29] explored the impact of resource failures and their recoveries and proposed a cost optimization model under fault tolerance aware cloud environments.

From the above review of the related work, it can be noted that existing studies have explored VM scheduling problem from different aspects by optimizing different objectives, however, only few works consider the influence of resource failures and their recoveries, in which the level of users' satisfaction degree, one of the most attractive concerns for cloud users, is not well addressed. On the basis of the existing works, this paper further takes users' satisfactions into consideration and explores fault tolerance aware VM scheduling problem by considering the influences of resource failures and recoveries.

3 Problem Description and Optimization Model

3.1 Problem Description

In cloud environments, users may need to execute their applications by submitting VM requests to cloud providers who own the necessary infrastructure resources in their data centers. Users' VM requests can be submitted at random instants with some specific resource demands (e.g. CPU cores, memory size) and certain QoS requirements [3, 7, 8]. Generally, cloud users always want their applications to be successfully completed as soon as possible and at the same time, at the lowest cost. Thus this paper considers two QoS constraints, i.e., budget and deadline which users are more concerned about. Each VM request may need to execute several tasks and all the tasks contained in the same VM request should be executed concurrently on the same physical server [8, 12, 16, 29]. Suppose that cloud users totally submit $n(n \geq 1)$ VM requests to a cloud data center, denoted by $\mathbf{VM} = \{VM_1, VM_2, \ldots, VM_n\}$. The i-th VM request $VM_i(1 \leq i \leq n)$ can be characterized by a tuple with six terms, $VM_i = (K_i, WL_i, mem_i, subt_i, B_i, D_i)$ [7, 8, 25]. In the tuple, the term $K_i(K_i \geq 1)$ is the number of tasks contained by VM request VM_i, each of which requires one CPU core to execute. It means that VM_i can be successfully allocated to a PS only if the PS can provide at least K_i CPU cores for it. $WL_i = \{wl_{ik} | 1 \leq k \leq K_i\}$ represents the set of workloads of VM_i's K_i tasks, in

which wl_{ik} is the workload length that VM_i's k-th task need to complete. The term mem_i denotes the required memory size of VM_i. The term $subt_i$ stands for VM_i's arrival time, which indicates when VM_i arrives at the cloud data center. B_i and D_i are VM_i's budget and deadline constraint respectively. The former indicates that a user's expenditure for executing VM_i must be less than B_i and the latter indicates that the cloud user desires its VM request VM_i to be completed no later than D_i.

In order to satisfy performance requirements of the worldwide users, cloud providers have established some cloud data centers around the world. A cloud data center contains some infrastructure resources, which may consist of thousands of heterogeneous physical servers (PSs). Without loss of generality, we use $\mathbf{PS} = \{PS_1, PS_2, \ldots, PS_m\}$ to represent the $m(m \geq 1)$ heterogeneous PSs in the cloud data center. The j-th $(1 \leq j \leq m)$ physical server, PS_j, can also be characterized by a tuple with six terms, $PS_j = (Core_j, Mem_j, s_j, p_j, \lambda_j, \mu_j)$ [3, 8, 29]. In PS_j, the terms $Core_j$ and Mem_j represent PS_j's CPU and memory capacities, i.e., the number of available CPU cores and memory size respectively. The parameter s_j denotes the processing speed of each of PS_j's CPU cores. p_j represents the price of each of PS_j's CPU cores, which is the cost of renting a single CPU core for a time unit. Suppose that the failures on physical server PS_j follow a Poisson process with the failure rate, denoted by λ_j $(\lambda_j \geq 0)$ and the failures on different PSs are independent of one another. That is to say, the interval time series of successive failures on PS_j are independent and obey a negative exponential distribution with the same parameters λ_j, i.e., $F(t) = 1 - e^{-\lambda jt}$, $t \geq 0$. Once a failure happens on a PS, it will initiate a repair process. Accordingly, we assume that the repair times of failures on physical server PS_j are also independent and follow a negative exponential distribution with the same repair rates, denoted by $\mu_j(\mu_j \geq 0)$, i.e., $G(t) = 1 - e^{-\mu jt}$, $t \geq 0$ [9, 10].

Thus, the fault tolerance aware VM scheduling problem can be formally described as follows: suppose that there is a cloud data center with m heterogeneous PSs $\mathbf{PS} = \{PS_1, PS_2, \ldots, PS_m\}$, and all cloud users submit totally n VM requests, $\mathbf{VM}=\{V_1, V_2, \ldots, V_n\}$, the studied problem is how to map each VM request to a suitable PS, so as to maximize the SERoV and at the same time, minimize users' TE under the cloud environment where fault tolerance techniques are adopted.

3.2 Optimization Model

Definition 1 Scheduling Matrix (SM). Denote by $X = (x_{ij})_{n \times m}$ the scheduling matrix, where the decision variable x_{ij} indicates that whether VM_i is assigned to PS_j. The reasonable range of x_{ij} is 0 or 1: If VM_i is assigned to PS_j, then $x_{ij} = 1$; otherwise, $x_{ij} = 0$.

Denote by τ_{ikj} the *ideal execution time* of the k-th task of VM request VM_i on physical server PS_j, so we have

$$\tau_{ikj} = \frac{wl_{ik}}{s_j}, \tag{1}$$

where wl_{ik}, in terms of millions of instructions (MI), is the workload length of the kth task of VM_i and s_j is the processing speed of PS_j's each CPU core, in terms of million instructions per second (MIPS).

However, in real-life cloud environment, failures may happen on PSs and fault recovery techniques are generally adopted. In this situation, the *actual execution time* of k-th task of VM_i on PS_j, denoted by AT_{ikj}, is different from its ideal execution time. Denote by $N_j(\tau_{ikj})$ the total number of failures that occur on PS_j during the time interval τ_{ikj} and the failure rate of PS_j is λ_j, then the probability of $N_j(\tau_{ikj}) = M$ ($M = 0, 1, 2, \ldots$) can be given by

$$\Pr\{N_j(\tau_{ikj}) = M\} = \frac{(\lambda_j \tau_{ikj})^M}{M!} e^{-\lambda_j \tau_{ikj}}, \quad M = 0, 1, 2, \ldots. \tag{2}$$

It can be seen that $N_j(\tau_{ikj})$ is a stochastic variable whose expectation can be given by

$$E[N_j(\tau_{ikj})] = \lambda_j \tau_{ikj}. \tag{3}$$

Denote by $RT_j^{(M)}$ the recovery time of the M-th ($M = 0, 1, 2, \ldots$) failure occurred on PS_j. By summing up all $N_j(\tau_{ikj})$ failures' recovery times, we can get the total recovery time of PS_j during $(0, \tau_{ikj}]$, denoted by $RT_j(\tau_{ikj})$.

$$RT_j(\tau_{ikj}) = \sum_{M=1}^{N_j(\tau_{ikj})} RT_j^{(M)}. \tag{4}$$

The actual execution time of k-th task of VM_i on PS_j, AT_{ikj}, is the sum of the ideal execution time and the total recovery time on PS_j during executing the task, as shown in Eq. (5).

$$AT_{ikj} = \tau_{ikj} + RT_j(\tau_{ikj}). \tag{5}$$

It can be seen that AT_{ikj} is a random variable whose expectation can be given by

$$E[AT_{ikj}] = \frac{(\mu_j + \lambda_j)\tau_{ikj}}{\mu_j}. \tag{6}$$

Generally, cloud users always hope that their VM requests can be successfully completed at lowest possible expenditure. If users need to spend much expenditures or their VM executions often violate QoS constraints, then they are very likely to lose interest in the data center. Therefore, this paper makes an endeavor to *maximize the successful execution rate of VM requests (SERoV)* and *minimize the total expenditure (TE)* for executing all cloud users' VM requests. Users' total expenditure equals to the sum of the execution costs for all the successfully completed VM requests, as shown in Eq. (7).

$$TE = \sum_{i=1}^{n} \sum_{j=1}^{m} x_{ij} \cdot \left[K_i \cdot \max_{1 \le k \le K_i} (AT_{ikj}) \cdot p_j \right], \tag{7}$$

in which x_{ij}, shown in Definition 1, is the decision variable of the studied problem. The term $\max_{1 \le k \le K_i}(AT_{ikj})$ is the actual completion time of VM_i on PS_j, which equals to

the maximum actual completion time among VM_i's K_i tasks. It can be noted that TE is a random variable whose expectation can be given by

$$E[TE] = \sum_{i=1}^{n}\sum_{j=1}^{m} x_{ij} \cdot \left[K_i \cdot E[\max_{1\le k\le K_i}(AT_{ikj})] \cdot p_j \right]. \tag{8}$$

Denote by θ the SERoV, which can be calculated by the number of successfully completed VM requests divided by n, i.e., the total number of VM requests submitted by all cloud users. Then we have

$$\theta = \frac{1}{n}\sum_{i=1}^{n} \varphi_i, \tag{9}$$

in which the variable φ_i, given by Eq. (10), indicates whether VM request VM_i is successfully assigned to a PS.

$$\varphi_i = \sum_{j=1}^{m} x_{ij}. \tag{10}$$

This paper explores the fault tolerance aware VM scheduling by focusing on optimizing two major performance criteria for cloud users, i.e., *maximizing the SERoV* and *minimizing TE*, under certain budget and deadline requirements. Thus the studied problem can be formulated as a multi-objective optimization model with some constraints.

Objectives:

$$\text{Min} \quad E[TC] = \sum_{i=1}^{n}\sum_{j=1}^{m} x_{ij} \cdot \left[K_i \cdot E[\max_{1\le k\le K_i}(AT_{ikj})] \cdot p_j \right]; \tag{I}$$

$$\text{Max} \qquad \theta = \frac{1}{n}\sum_{i=1}^{n} \varphi_i. \tag{II}$$

Subject to:

i). For each $i \in \{1, 2, \ldots, n\}$ and each $j \in \{1, 2, \ldots, m\}$, the value of decision variable x_{ij} is within the range of $\{0, 1\}$;
ii). For each $i \in \{1, 2, \ldots, n\}$, then $\sum_{j=1}^{m} x_{ij} \le 1$;
iii). For each $j (1 \le j \le m\}$, then $\sum_{i=1}^{n} x_{ij} K_i \le Core_j$ and $\sum_{i=1}^{n} x_{ij} mem_i \le Mem_j$;
iv). For each $i \in \{1, 2, \ldots, n\}$, if the value of decision variable x_{ij} is 1, then physical server PS_j must satisfy Eq. (11) and Eq. (12).

$$wt_i + E[\max_{1\le k\le K_i}(AT_{ikj})] \le D_i; \tag{11}$$

$$K_i \cdot E[\max_{1\le k\le K_i}(AT_{ikj})] \cdot p_j \le B_i. \tag{12}$$

The first optimization objective, (I), is to minimize the expectation of total expenditure for executing all cloud users' VM requests. The second one, (II), is to maximize the successful rate of their VM requests. By optimizing the two objectives, cloud providers can offer sufficient incentives for their users to stay and play in the data centers, which cloud help to keep the cloud ecosystem sustainable.

The constraint i) gives the decision variable x_{ij}'s feasible region. The second constraint, ii), guarantees that a VM request must be allocated to exactly one PS. The third constraint, iii), is PS_j's resource capacity restrictions, which means that the total CPU cores and memory size required by VM requests assigned to PS_j must not exceed its CPU and memory capacities respectively. The last one, constraint iv), ensures that if VM_i is allocated to PS_j, then the physical server should satisfy VM_i's deadline and budget requirements. The term wt_i, in Eq. (11), is the waiting time interval that VM request VM_i waits for being executed on PS_j. The term $E[\max_{1\leq k\leq K_i}(AT_{ikj})]$ is the expectation of actual execution time of VM_i on PS_j. Equation (11) guarantees that VM request VM_i's waiting time sums up its actual execution time should not exceed the required deadline. Equation (12) guarantees that the expenditure for executing VM_i on PS_j must not exceed the required budget.

From the above descriptions, it can be seen that the proposed multi-objective optimization model belongs to the combinatorial optimization, which is NP-hard problem [30]. Heuristic-based approaches, which can offer an approximately optimal solution in an acceptable time frame proportional to the number of variables, have recently attracted much attention to deal with VM scheduling problem [3, 30]. Therefore, this paper develops a heuristic method, i.e., *greedy-based best fit decreasing scheduling* (*GBFDS*) *algorithm*, to solve the fault tolerance aware VM scheduling problem.

4 Proposed Algorithm

Before describing the details of GBFDS algorithm, we first define two concepts used in the developed algorithm.

Definition 2. Cost Efficiency (CE) factor: The cost efficiency of physical server PS_j is mainly influenced by four parameters, i.e., its CPU cores' price (p_j), processing speed (s_j), failure rate (λ_j) and recovery rate (μ_j). If failures frequently occur on PS_j, then VM requests assigned on the server will experience long executing time, which will inevidently increase users' expenditure. Similarly, if the price of PS_j's CPU core is high, it will also increase users' cost for executing their VM requests. On the contrary, the higher its processing speed is, the shorter time that the PS needs to complete a VM request. And similarly the higher its recovery rate is, the shorter time the PS recovers from a failure. Therefore, the CE factor of PS_j, denoted by $CE(PS_j)$, can be defined as the product of its processing speed, recovery rate and the inverse of its failure rate and CPU cores' price, as shown in Eq. (13). The higher a PS's CE factor is, the lower the cost for executing a VM request is.

$$CE(PS_j) = \frac{\mu_j s_j}{\lambda_j p_j}. \tag{13}$$

Definition 3. Candidate Server Set (CSS): For an arbitrary VM request $VM_i \in \mathbf{VM}$, if the available resources (CPU cores and memory size) of a physical server $PS_j \in \mathbf{PS}$ are no less than VM_i's resource requirements and, at the same time, can successfully complete VM_i's tasks before the specified deadline with the expenditure no more than its budget, then PS_j is a candidate server of VM_i. The candidate server set of VM_i, denoted by CSS_i, is composed of all its candidate servers.

If a VM request's CSS contains few candidate servers, it means that the VM request's QoS constraints are tight or its resource demands are very high and few PSs can satisfy its requirements. Granting high priorities to these VM requests can effectively increase their successful execution rate.

Based on above-mentioned definitions and analysis, the proposed algorithm preferentially schedules the VM request whose CSS has fewest candidate physical servers to improve SERoV. In order to decrease could users' total expenditure, GBFDS algorithm allocates the VM request, VM_i, whose CSS contains more than one candidate server to the server whose cost efficiency factor is the smallest among VM_i's candidate servers. The processes of GBFDS algorithm are shown as follows.

Algorithm 1: Greedy-based Best Fit Decreasing Scheduling (GBFDS) algorithm

Inputs: $\mathbf{VM}=\{VM_1, VM_2, \ldots, VM_n\}$ and $\mathbf{PS}=\{PS_1, PS_2, \ldots, PS_m\}$.
Outputs: the scheduling matric $\mathbf{X}=(x_{ij})_{n\times m}$.

```
1  Initialization: mark all VM requests' states as unscheduled
                   set X=0 and all VM requests' CSSs as Φ;
2  sort physical servers in descending order by CE factor, such as PS1', PS2', …, PSm';
3  foreach unscheduled VM request VMi ∈ V do
4   foreach physical server PSj' (1≤j≤m) do
5    if PSj can satisfy VMi's resource demands and QoS requirements then
6     CSSi = CSSi ∪ PSj';
7  if CSSi = Φ then
8   add VM request VMi to unsuccessful scheduled set U;
9   change VM request VMi's state to failed scheduling;
10 else if |CSSi|=1 then
11  assign VM request VMi to the candidate server (PSj' might as well) in CSSi;
12  change available resource capacities (CPU cores and memory size) of PSj';
13  set xij=1;
14  change VM request VMi's state to scheduled;
15 while there exists unscheduled VM request in V do
16  find the candidate server set with fewest candidate servers, CSSi might as well;
17  assign VMi to the physical server PSj' whose subscript is the smallest in CSSi;
18  change the available resource capacity of PSj';
19  set xij=1;
20  change VM request VMi's state to scheduled;
21 return X;
```

Initially, GBFDS algorithm sets all VM requests' states as *unscheduled*, all VM requests' CSSs as empty set and all the elements x_{ij} of $\mathbf{X} = (x_{ij})_{n\times m}$ as 0. After this, all the PSs are sorted in descending order by their values of CE factors. Might as well,

the order of PSs after sorting is PS_1', PS_2', …, PS_m'. Then, the following steps will be iteratively executed by GBFDS algorithm.

For each VM request VM_i ($1 \leq i \leq m$) in the state of *unscheduled*, GBFDS algorithm firstly find out its candidate server set, CSS_i, by orderly checking PS_1', PS_2', …, PS_m'. During this process, if one VM request VM_i's candidate server set CSS_i is empty, which means that no existing physical server can satisfy VM_i's resources or QoS requirements, then the VM request, VM_i, can't be successfully scheduled and its state will be changed to *failed*. Otherwise, if the VM request VM_i's candidate server set only contains one candidate server, PS_j', then GBFDS algorithm preferentially schedules VM_i and assigns it to PS_j' and then changes the relevant parameters. Secondly, for the other VM requests with more than one candidate server in their candidate server sets, the proposed algorithm preferentially schedules the VM request VM_i, which contains fewest candidate servers in its candidate server set CNS_i and then assigns VM_i to its candidate server PS_j' whose CE factor value is the smallest among all the PS in CNS_i. The scheduling process will be iteratively invoked until all VM requests are not in *unscheduled* state. Finally, GBFDS algorithm returns the values of the obtained scheduling matric, $\mathbf{X} = (x_{ij})_{n \times m}$.

5 Performance Evaluation

5.1 Simulation Configurations

In the experiments, we simulate a cloud data center, which totally contains one hundred physical servers. The number of CPU cores of each PS is an integer within the range of {2, 4, 8, 16}, which is randomly generated by a uniform distribution. The memory size of each PS is also randomly generated by a uniform distribution within the range of {4 GB, 8 GB, 16 GB, 32 GB}. For the parameters of processing speed and price of each physical server's CPU cores, we use the similar approach adopted in [3, 9, 10, 19, 29] to generate the values. The former parameter is in the range of 100–200, which is randomly generated by a uniform distribution with the average speed of 150, in terms of million instructions per second (MIPS). The CPU core's price of each PS is within the range of 0.35–1 and roughly linear with its processing speed, which is used to guarantee that a faster PS needs more renting expenditure than a slower one for executing the same VM request. The failure rates and the recovery rates of physical servers are uniformly distributed within the range of [0.01, 0.1] and [0.05, 0.15], respectively [9, 10, 28].

For the parameters of users' VM requests, all the experiments randomly set their arrival times within the range of (0, 100], which means that all VM requests arrive at the data center within a scheduling interval ($T = 100$ s). Their required number of tasks (i.e., CPU cores) and memory sizes are both integers, which randomly generate by uniform distributions within the range of [1, 7] and {1 GB, 2 GB, 3 GB, 4 GB} respectively. Tasks' workload lengths are uniformly generated within the value set, {100000, 120000, 140000, 160000, …, 500000}, in terms of million instructions (MI). The required deadline of each VM request is roughly linear with its average estimated runtime of the largest workload among all its tasks with a 10% variation and the budget requirement is set as its deadline multiplying by the number of task, as well as the average price with a 10% variation. For eliminating the influence of causal factors, all

the experiments are repeatedly conducted 100 times and the presented results are the mean value of these repeated experiments.

5.2 Performance Metrics

In order to test and verify the feasibility of the proposed optimization model and algorithm, we compare the developed algorithm with two other related ones, i.e., FCFS [19] and MBFD [29], under four popular performance metrics. The first measured metric is the SERoV, which is one of the optimization objectives of the proposed model and can be calculated by Eq. (9) and Eq. (10). The second one is the *average expenditure* (AE), which can be calculated as the expectation of total expenditure (shown in Eq. (8)) divided by the number of successfully executed VM requests. The third metric is the *average execution time* (AET), which is the average value of all the successfully executed VM requests' actual completion times. The actual completion time of a successfully executed VM request is defined as the time span between its submitted time and completed time. The last measured metric is the *overall user satisfaction* (OUS). Generally, the cloud users want their submitted VM requests can be successfully completed as soon as possible and at the same time, at the lowest possible expenditure. Therefore, the satisfaction of the cloud user who submits VM require VM_i, denoted by us_i, is influenced by three factors, i.e., whether VM request VM_i is successfully executed or not (φ_i), the used time and expenditure for executing VM_i. Denote by $Etime_i$ and $Ecost_i$ the execution time and expenditure of VM request VM_i respectively, so we have

$$us_i = \varphi_i \cdot [\alpha \cdot \frac{d_i - Etime_i}{d_i} + \beta \cdot \frac{b_i - Ecost_i}{b}]. \tag{14}$$

In Eq. (14), α and β are cloud users' preference coefficients for time and cost with the values satisfying $\alpha + \beta = 1$. If a cloud user prefers its VM request to be completed as soon as possible, then we can set $\alpha > \beta$, and vice versa. In this paper, we trade time and expenditure as two factors with equal importance and thus set $\alpha = \beta = 0.5$

Summing up the values of all cloud users' satisfactions, we can get the overall user satisfaction, OUS (as shown in Eq. (15)).

$$OUS = \sum_{i=1}^{n} us_i. \tag{15}$$

5.3 Simulation Results

In this section, we conduct two experiments to fully evaluate the performance of the proposed model and developed algorithm.

Experiment 1. In this experiment, we first fixed all physical servers' recovery rate as 0.1 to observe the influence of failure rate under varied values from 0.01 to 0.1. The obtained results are shown in Table 1. It can be seen that, with the increasing of PSs' failure rate, the results of SERoV and OUS smoothly decrease and these of AE and

AET gradually increase, which means that all the measured metrics get worse. This is because, with the increasing of failure rates, the probability that failures happen on a physical server increases, which will evidently increase the execution time of users' VM requests, thus increasing users' expenditures. This will be more likely to violate users' deadline and budget constraints and thus decrease the SERoV.

Table 1. Results obtained by GBFD algorithm with different failure rates when physical servers' recovery rate is fixed as 0.1.

Failure rete	SERoV	AE	AET	OUS
0.01	100%	2.22	0.84	102.11
0.02	100%	2.42	0.91	93.37
0.03	100%	2.63	0.99	84.64
0.04	100%	2.83	1.06	75.90
0.05	100%	3.02	1.11	68.10
0.06	100%	3.24	1.19	59.15
0.07	100%	3.44	1.24	51.86
0.08	100%	3.66	1.33	42.18
0.09	97%	3.90	1.37	33.45
0.1	72%	3.70	1.56	17.08

Second, Table 2 shows the results of measured metrics obtained by GBFDS algorithm with varied recovery rates under the situation that all physical servers' failure rate is fixed as 0.01. It can be found that all the measured metrics get better with recovery rate varying from 0.05 to 0.14. The reason lies in that, with the increasing of physical servers' recovery rates, the recovery time that a failed PS needed to recover from a failure is decreased and thus the average execution cost decreases, which means that the PS will be more likely to successfully complete the served VM requests and thus increase successful execution rate and cloud users' satisfactions.

Thus, we can conclude that, as expected, adopting fault tolerance techniques in cloud data centers does have significant impact on the performance metrics of VM scheduling.

Experiment 2. In this experiment, we evaluate the performance of the developed GBFDS algorithm by comparing with other ones. The results of the four performance criteria obtained by different algorithm are presented in Table 3. From the results, we can the following trends: First, the developed GBFDS algorithm can obtain the highest successful execution rate. Compared with FCFS and MBFD, GBFDS algorithm can successfully execute more VM requests by 12.05% and 4.25%, respectively. This is because GBFDS algorithm preferentially schedules tight-QoS-constrained VM requests whose candidate server sets may have few candidate servers. Granting higher priority to schedule these VM requests can increase their probability of being successfully completed and thus improve SERoV. Second, the developed GBFDS algorithm can achieve

Table 2. Results obtained by GBFD algorithm with different recovery rates when physical servers' failure rate is fixed as 0.05.

Recovery rete	SERoV	AE	AET	OUS
0.05	71.5%	3.70	1.56	17.08
0.06	100%	3.73	1.34	39.42
0.07	100%	3.48	1.26	50.13
0.08	100%	3.29	1.20	57.69
0.09	100%	3.14	1.15	63.62
0.10	100%	3.02	1.11	68.10
0.11	100%	2.94	1.10	70.57
0.12	100%	2.86	1.07	74.44
0.13	100%	2.80	1.05	77.24
0.14	100%	2.74	1.03	79.64

the smallest average expenditure (AE) of all successfully executed VM requests. Specifically, compared with other methods, GBFDS algorithm can cut down the users' AE by about 2.79% and 0.71%, respectively. Third and most important, the developed GBFDS algorithm can obtain the highest degree of user satisfaction. Compared with the other methods, GBFDS algorithm can improve cloud users' satisfaction by about 20.1% and 10.4%, respectively. It can be concluded that, compared with some popular algorithms, the developed GBFDS algorithm can achieve higher successful execution rate, lower execution cost, and most important, higher degree of user satisfaction in most cases. Thus the proposed GBFDS algorithm can meet users' satisfaction better.

Table 3. Results obtained by compared algorithms under fault tolerance-aware environments.

Metrics	FCFS	MBFD	GBFDS
SERoV	82.3%	90.1%	**94.35%**
AE	2.87	2.81	**2.79**
AET	1.23	**1.19**	1.21
OUS	57.69	63.74	**69.26**

6 Conclusions

In this paper, we deal with the problem of fault tolerance aware VM scheduling. By considering the impact of fault tolerance techniques, two stochastic models of actual execution time and cost are deduced. Then the studied problem is formulated as a multi-objective optimization model with multiple QoS constraints. A greedy-based best fit

decreasing algorithm is then developed. Finally, the experimental results demonstrate the feasibility of the proposed model and algorithm. As future work, we will consider the situation that not all failures are recoverable and VM requests may fail.

Acknowledgements. This work is partially supported by the National Natural and Science Foundation of China (No. 61472460, 61702162 and U1504607), Natural Science Project of the Education Department of Henan Province (No. 19A520021), Program for Innovative Research Team (in Science and Technology) in University of Henan Province (No. 17IRTSTHN011), Science and Technology Project of Science and Technology Department of Henan Province (No. 172102110013), Plan for Nature Science Fundamental Research of Henan University of Technology (No. 2018QNJH26), Plan For Scientific Innovation Talent of Henan University of Technology (No. 2018RCJH07) and the Research Foundation for Advanced Talents of Henan University of Technology (2017BS016).

References

1. Armbrust, A.M., Fox, A., Griffith, R., et al.: A view of cloud computing. Commun. ACM **53**(4), 50–58 (2010)
2. Liu, L., Qiu, Z.: A survey on virtual machine scheduling in cloud computing. In: Proceedings of 2nd International Conference on Computer and Communications, Chengdu, China, pp. 2717–2721. IEEE (2016)
3. Xu, H., Liu, Y., Wei, W., Zhang, W.: Incentive-aware virtual machine scheduling in cloud computing. J. Supercomput. **74**(7), 3016–3038 (2018)
4. Madni, S.H.H., Latiff, M.S.A., Coulibaly, Y.: Resource scheduling for infrastructure as a service (IaaS) in cloud computing: challenges and opportunities. J. Netw. Comput. Appl. **68**(1), 173–200 (2016)
5. Mann, Z.Á.: Allocation of virtual machines in cloud data centers—a survey of problem models and optimization algorithms. ACM Comput. Surv. **48**(1), 11 (2015)
6. Wang, Z., Hayat, M.M., Ghani, N., et al.: Optimizing cloud-service performance: efficient resource provisioning via optimal workload allocation. IEEE Trans. Parallel Distrib. Syst. **28**(6), 1689–1702 (2017)
7. Singh, S., Chana, I.: QRSF: QoS-aware resource scheduling framework in cloud computing. J. Supercomput. **71**(1), 241–292 (2015)
8. Zhang, R., Wu, K., Li, M., et al.: Online resource scheduling under concave pricing for cloud computing. IEEE Trans. Parallel Distrib. Syst. **27**(4), 1131–1145 (2016)
9. Xu, H., Yang, B., Qi, W., Ahene, E.: A multi-objective optimization approach to workflow scheduling in clouds considering fault recovery. KSII. Trans. Int. Inf. **10**(3), 976–995 (2016)
10. Sun, P., Wu, D., Qiu, X., Luo, L., Li, H.: Performance analysis of cloud service considering reliability. In: Proceedings of IEEE International Conference on Software Quality, Reliability and Security Companion (QRS-C), Vienna, Austria, pp. 339–343. IEEE (2016)
11. Meng, L., Sun, Y.: Context sensitive efficient automatic resource scheduling for cloud applications. In: Proceedings of the 11th International Conference on Cloud Computing, Seattle, USA, pp. 391–397 (2018)
12. Yu, L., Chen, L., et al.: Stochastic load balancing for virtual resource management in datacenters. IEEE Trans. Cloud Comput. (in press online). https://doi.org/10.1109/tcc.2016.2525984
13. Wei, L., Foh, C.H., He, B., et al.: Towards efficient resource allocation for heterogeneous workloads in IaaS clouds. IEEE Trans. Cloud Comput. **6**(1), 264–275 (2018)

14. Imai, S., Patterson, S., Varela, C. A.: Elastic virtual machine scheduling for continuous air traffic optimization. In: Proceedings of 16th International Symposium on Cluster, Cloud and Grid Computing, Cartagena, Colombia, pp. 183–186. IEEE (2016)
15. Secinti, C., Ovatman, T.: Fault tolerant VM consolidation for energy-efficient cloud environments. In: Luo, M., Zhang, L.-J. (eds.) CLOUD 2018. LNCS, vol. 10967, pp. 323–333. Springer, Cham (2018). https://doi.org/10.1007/978-3-319-94295-7_22
16. Xu, H., Yang, B.: Energy-aware resource management in cloud computing considering load balance. J. Inf. Sci. Eng. **33**(1), 1–16 (2017)
17. Mishra, S.K., Puthal, D., Sahoo, B., et al.: An adaptive task allocation technique for green cloud computing. J. Supercomputing **74**(1), 370–385 (2018)
18. Xu, H., Liu, Y., Wei, W., Xue, Y.: Migration cost and energy-aware virtual machine consolidation under cloud environments considering remaining runtime. Int. J. Parallel Prog. **47**(3), 481–501 (2019)
19. Wang, D., Dai, W., Zhang, C., Shi, X., Jin, H.: TPS: an efficient VM scheduling algorithm for HPC applications in cloud. In: Au, M.H.A., Castiglione, A., Choo, K.-K.R., Palmieri, F., Li, K.-C. (eds.) GPC 2017. LNCS, vol. 10232, pp. 152–164. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-57186-7_13
20. Kohne, A., Pasternak, D., Nagel, L., et al.: Evaluation of SLA-based decision strategies for VM scheduling in cloud data centers. In: Proceedings of the 3rd Workshop on Cross Cloud Infrastructures and Platforms, no. 6. ACM, London (2016)
21. Zeng, L., Wang, Y., Fan, X., et al.: Raccoon: a novel network I/O allocation framework for workload-aware VM scheduling in virtual environments. IEEE Trans. Parallel Distrib. Syst. **28**(9), 2651–2662 (2017)
22. Guo, M., Guan, Q., Ke, W.: Optimal scheduling of VMs in queuing cloud computing systems with a heterogeneous workload. IEEE Access **6**, 15178–15191 (2018)
23. Yu, Q., Wan, H., Zhao, X., et al.: Online scheduling for dynamic VM migration in multicast time-sensitive networks. IEEE Trans. Industr. Inf. (online press). https://doi.org/10.1109/tii.2019.2925538
24. Li, S., Zhou, Y., Jiao, L., et al.: Towards operational cost minimization in hybrid clouds for dynamic resource provisioning with delay-aware optimization. IEEE Trans. Serv. Comput. **8**(3), 398–409 (2015)
25. Somasundaram, T.S., Govindarajan, K.: CLOUDRB: a framework for scheduling and managing high-performance computing (HPC) applications in science cloud. Future Gener. Comput. Syst. **34**, 47–65 (2014)
26. Ran, Y., Yang, J., et al.: Dynamic IaaS computing resource provisioning strategy with QoS constraint. IEEE Trans. Serv. Comput. **10**(2), 190–202 (2017)
27. Sotiriadis, S., Bessis, N., Buyya, R.: Self managed virtual machine scheduling in cloud systems. Inf. Sci. **433**, 381–400 (2018)
28. Sun, P., Dai, Y., Qiu, X.: Optimal scheduling and management on correlating reliability, performance, and energy consumption for multi-agent cloud systems. IEEE Trans. Reliab. **66**(2), 547–558 (2017)
29. Xu, H., Cheng, P., Liu, L., Wei, W.: Fault tolerance aware virtual machine scheduling in cloud computing. In: Proceedings of 5th International Symposium on System and Software Reliability, Chengdu, China. IEEE (2019)
30. Kurdi, H., Al-Anazi, A., et al.: A combinatorial optimization algorithm for multiple cloud service composition. Comput. Electr. Eng. **42**, 107–113 (2015)