IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

SPECIAL SECTION ON BIG DATA LEARNING AND DISCOVERY

Received November 12, 2018, accepted November 27, 2018, date of publication November 29, 2018,
date of current version December 27, 2018.

Digital Object Identifier 10.1109/ACCESS.2018.2884028

HirePool: Optimizing Resource Reuse Based on a Hybrid Resource Pool in the Cloud

RUNQUN XIONG[1], XIUYANG LI[1], JIYUAN SHI[2], ZHIANG WU[3], (Member, IEEE), AND JIAHUI JIN[1]

[1]School of Computer Science and Engineering, Southeast University, Nanjing 211189, China
[2]Huawei Technologies Co., Ltd., Nanjing 210012, China
[3]Jiangsu Provincial Key Laboratory of E-Business, Nanjing University of Finance and Economics, Nanjing 210003, China

Corresponding author: Jiahui Jin (jjin@seu.edu.cn)

This work was supported in part by the National Science Foundation of China under Grant 61602112, Grant 61632008, Grant 61572129, and Grant 61502097, in part by the Natural Science Foundation of Jiangsu Province under Grant BK20160695 and Grant BK20170689, in part by the Jiangsu Provincial Key Laboratory of Network and Information Security under Grant BM2003201, in part by the Key Laboratory of Computer Network and Information Integration of the Ministry of Education of China under Grant 93K-9, in part by the Collaborative Innovation Center of Novel Software Technology and Industrialization, and in part by the Collaborative Innovation Center of Wireless Communications Technology.

ABSTRACT In a cloud environment, the primary way to optimize physical resources is to reuse a physical machine (PM) by consolidating complementary multiple virtual machines (VMs) on it. When considering VMs' dynamically changing resource demands, one hot research topic revolves around reusing VM migration resources more efficiently. The challenge here is finding the best tradeoff between the VM migration optimization performance and complexity. On one hand, to improve the migration efficiency, VM migration planning is adopted to achieve efficient resource reuse while minimizing the number of VM migrations. On the other hand, the huge number of PMs and VMs in a cloud datacenter often adds considerable complexity to migration planning, which hampers the decision-making process in VM migration. To address these issues, this paper proposes a hybrid resource pool model to reduce the complexity of VM migration planning by limiting the scope of VM migration decisions. Then, based on this model, we use our novel resource-reuse optimization mechanism (called HirePool) to improve efficiency by maximizing resource usage with only a few VM migrations. Finally, we perform simulation tests and actual experiments running on a real cloud platform to evaluate HirePool. Results show that HirePool improves average resource usage by 13%, saves the number of PMs used by 12%, and reduces the average number of migrations (compared with contrast mechanisms) by 31%.

INDEX TERMS Resource reuse, cloud environment, dynamic resource requirement, virtual machine migration, optimization.

I. INTRODUCTION

With the development of cloud computing technology, an increasing number of enterprises and research institutions are deploying various commercial big data processing and scientific computing applications in cloud datacenters [1]–[4]. Cloud datacenters can elastically provide storage space, computing resources (CPUs cores or CPU time), and network bandwidth in the form of virtual machines (VMs) deployed on physical machines (PMs) to satisfy multifarious users' requirements.

To reduce energy consumption and operating costs, the efficiency of PM resource reuse has become a tremendous concern for cloud datacenter managers [5]. Reusing a PM as a resource (by consolidating multiple complementary VMs on it) is an effective way to optimize physical resource usage. Some related research mainly focuses on estimating VM resource requirements, and optimizing VM allocation for higher resource usage [6]–[11]. However, current approaches are for coarse-grained resource reuse, which is only suitable for one-off VM deployment; these approaches are so inefficient that they cannot provide the fine-grained resource adjustment needed to meet running VM dynamic resource requirements.

In recent years, when considering VMs' dynamically changing resource requirements, many researchers have aimed to increase resource reuse's efficiency via

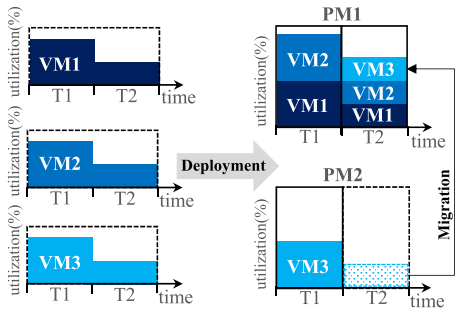


FIGURE 1. Increasing resource reuse efficiency with virtual machine (VM) migration (PM stands for physical machine).

VM migration [12]–[19]. As Fig.1 shows, VM3 can migrate from PM2 to PM1 during the T2 time period according to VM dynamic use. This greatly increases VM1's utility, but it causes VM2 to go out of service. This process can (and should be) improved. The key challenge here is coming up with an effective VM migration plan that adjusts VM deployments in different periods as VM resource requirements dynamically change.

Over the years, many solutions have been proposed to resolve this issue; however, existing work offers an insufficient trade-off between VM migration optimization performance and complexity. On one hand, most solutions lack effective VM migration planning, which results in poor migration efficiency or too many migrations. In other words, they only consider the current, immediate resource demands, and ignore the dynamically changing resource requirements over multiple time periods [12], [15]–[17]. On the other hand, multiple efficient models have emerged that reduce the number of migrations [13], [14], [18], [21], but they are difficult to deploy in a real cloud datacenter. The reason why is that the enormous number of PMs and VMs in a cloud datacenter dramatically increases the solution's complexity. For example, assume that there are m PMs and each PM is allocated n VMs in a datacenter. They must consider $n^{m\times t}$ possible number of migrations in the next t time periods. Such a high solution space complexity is unable to keep up with the needs of rapid decision-making for VM migration planning.

To solve this contradiction, here we propose an efficient resource-reuse optimization mechanism based on our novel hybrid resource pool model named HirePool. The model divides all PMs into two pools in a datacenter, and limits the scope of VM migration decision to two PMs that belong to the two pools, respectively. In this way, it can decrease the solution-space and decision-making complexity of VM migration planning from $n^{m\times t}$ to $2^{m\times t}$. Furthermore, based on this model, HirePool improves the efficiency of resource reuse by maximizing physical resource usage with fewer VM migrations. HirePool can adjust how it allocates VMs through reasonable migration planning, according to the dynamically changing resource requirements of multiple time periods in the future.

The main contributions of this paper are as follows:

1) We propose a hybrid resource pooling model that reduces the complexity of VM migration planning for resource reuse in large-scale cloud datacenters.
2) We design a resource-reuse optimization mechanism (HirePool) to improve the efficiency of resource reuse by maximizing resource usage with fewer VM migrations.
3) We verify HirePool's efficiency through simulation tests and testbed experiments in a real cloud platform.

The rest of this paper is organized as follows. Section II introduces related work. Section III briefly provides background knowledge. A novel hybrid resource pool model for resource reuse is presented in Section IV. Then, the detailed math model and design of HirePool are depicted, too. Section VI evaluates HirePool's performance using experiments, and Section VII concludes the paper and discusses future work.

II. RELATED WORK

In trying to optimize cloud datacenter resource utilization with resource reuse, many researchers have focused on VM placement (or allocation) and VM migration.

In the field of VM placement, related work states that VMs can efficiently reuse a PM's resource by allocating physical resources to VMs based on their resource demands and consolidating complementary VMs on this PM. In practice, some research [7], [8] has modeled the change of resource demands as a temporal relations model, and deployed the VM appropriately along a given time interval based on the model, then co-located VMs with complementary resource requirements in the same PM to optimize resource utilization. Meanwhile, other work has used the queuing theory [6], [11], probability statistics [9], [10], and other mathematical tools to model the dynamic change of VM resource demands and on that basis, researchers proposed mechanisms to optimize VM placement, in an effort to achieve high physical resource usage in cloud datacenters.

Recent VM migration researchers have pursued approaches that only focus on migrating VMs according to the most immediate or current resource requirements. Because these approaches only consider resource demands at one or each time point instead of planning holistically for the overall VM migration process, they solve specific, limited instances in the short term but in the long term, they lead to inadequate availability and use of physical resources. For example, Ruan and Chen [15] presented a novel VM allocation algorithm that leverages the performance-to-power ratios for various host types, and optimizes the balance in VM migration between host usage and energy consumption. Chen *et al.* [16] proposed a resource-intensity-aware load-balancing method in clouds that migrates VMs from overloaded PMs to lightly loaded PMs, which significantly reduced the time and cost to achieve load balance and avoided future load imbalance. Eyraud-Dubois and Larchevêque [17] classified VMs by their resource demands and sorted PMs by their physical

resource load, and then presented a heuristic algorithm to handle the variations in VM resource requirements through live VM migrations. Homsi *et al.* [19] studied the workload consolidation for cloud datacenters with guaranteed QoS using VM migration to improve resource utilization and power consumption. Armant *et al.* [20] proposed some semi-online task assignment policies for workload consolidation in cloud computing systems.

Other VM migration researchers have focused on designing VM migration plans to improve the migration's efficiency. Han *et al.* [13] presented a migration planning method based on the Markov decision process (MDP). The authors adopted the Markov chain model to study the VM management problem, and used the horizon large-scale MDP to plan VM migration. Dabbagh *et al.* [14] proposed an efficient resource-allocation framework for overcommitted clouds that yields significant energy savings by minimizing PM overloads via resource usage prediction, and reducing the number of active PMs via efficient VM placement and migration planning. Ye *et al.* [18] proposed a profiling-based server consolidation framework that minimizes the number of PMs used in datacenters while maintaining satisfactory performance of various workloads. They designed two modules: first, a consolidation planning module that, given a set of workloads, minimizes the number of PMs by an integer programming model; and second, a migration planning module that, given a source VM placement scenario and a target VM placement scenario, minimizes the number of VM migrations using a polynomial time algorithm. Gaggero and Caviglione [21] researched how to plan VM migrations with dynamically changing resource requirements within the next few periods. The authors modeled this issue as a control optimization problem based on integer programming. However, because of its complexity, the authors used a Monte Carlo optimization method to approximate the problem.

After investigating current research, it is apparent that the following problems still exist (regarding resource reuse in a cloud environment):

1) Despite many previous VM placement strategies allocating physical resources to VMs, still yet they only consider resource demands at one or each time point, thereby failing to utilize different resources constantly and fully, because they ignore dynamically changing resource requirements.
2) Most research work related to VM migration lacks overall migration planning, which generates high migration overhead and degrades VM performance.
3) While some proposed VM migration planning mechanisms strive to improve migration efficiency, in practice, the huge number of PMs and VMs in a cloud datacenter often creates such a complex migration planning problem that it affects and hampers rapid decision-making for VM migration.

As a solution, we use a hybrid resource pooling model to reduce VM migration planning's complexity. We then use HirePool, the efficient resource-reuse optimization mechanism we designed to optimize physical resources' usage for large-scale cloud datacenters with efficient VM migration planning.

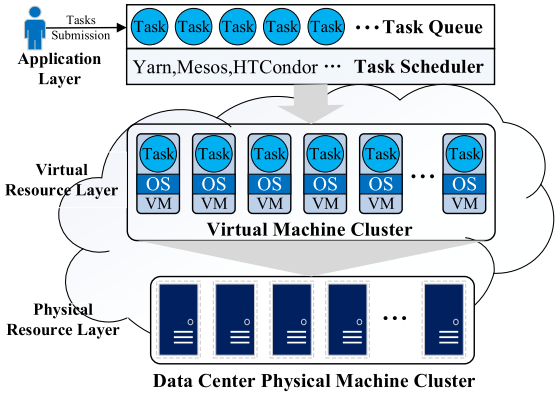


FIGURE 2. Cloud datacenter architecture.

III. PRELIMINARIES

To better understand the issues involved, here we provide background knowledge about a cloud datacenter's architecture and its resource-reuse mechanism(s).

A. CLOUD DATACENTER ARCHITECTURE

Fig.2 shows the cloud datacenter architecture used in this paper, which consists of three layers: the physical resource, virtual resource, and application layers. Cloud providers use visualization technologies to allocate physical resources to tenant VMs, which constitute a VM cluster that can provide flexible on-demand resources to service multiple tasks requested from the application layer. That's because VMs are easy to deploy and configure quickly, and the scale of virtual clusters (the number of VMs) conveniently is adjustable to the application's resource demands. Meanwhile, a cloud datacenter hosts multiple applications through a task scheduler (such as Yarn [22], Mesos [23], HTCondor [24]) that is responsible for allocating VMs to various running applications, subject to the familiar constraints of capacities, queues, and so forth. On such platforms, the cloud provider must find the best way to allocate VMs onto PMs to exploit the platform's resources, while still maintaining resources' availability guarantees [17].

B. RESOURCE REUSE

Currently, industry and academic research mainly use resource overcommitment as the foundation of resource reuse [25], [26]. As Fig.3(a) shows, on an overcommitted PM, the aggregate capacity requested by the provisioned VMs exceeds the actual PM's physical capacity, which allows multiplexing physical resources among multiple VMs, greatly improving resource usage which is the connotation of **resource reuse**. For instance, as Fig.3(b) shows, although the resource requirements of tasks running in VM1 and VM2 dynamically change at different time periods, these

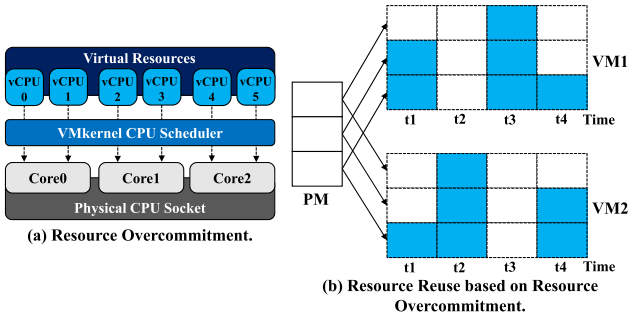


FIGURE 3. Resource reuse based on resource overcommitment.

two VMs that demand three virtual CPUs in peak can be deployed safely on a PM that only has three physical CPUs by harnessing a resource-reuse mechanism. The reason why is because the aggregate VM1 and VM2 resource usage at any given time does not exceed the total PM capacity. Under the circumstances, the cloud's physical resource usage is maximized. Thus, herein the term *overcommitment ratio* defines the degree of physical resource reuse.

Definition 1 (Overcommitment Ratio): We define an overcommitment ratio (OR) to represent the proportion of a virtual resource provided by a PM and its physical resource. For example, if a PM has 12 physical CPUs, it could provide 24 virtual CPUs by using virtualization technology, so the overcommitment ratio of this PM is OR=24÷12=2.

However, while executing a resource-reuse mechanism, the risk of resource congestion or resource leave unused may occur naturally, and then it needs a resource-reuse optimization mechanism to further improve physical resource usage. Specifically, at some time point, the aggregate VMs' resource usage exceeds the shared resources' available physical capacity, and then VM performance may significantly degrade. In this case, we need to migrate some VMs out of the PM to ensure that all the resource requirements for the remaining VMs can be satisfied. At other time periods, inversely, a large fraction of VM workloads rarely use all their resources, which leads to low physical resource usage. In this case, we need to migrate other VMs onto the PM to maximize resource usage.

IV. MECHANISM DESIGN AND IMPLEMENTATION

Resource reuse optimization adopts VM migration planning, to adjust deploying VMs in a way that maximizes resource usage and reduces the number of VM migrations. This requires effective migration planning, to determine which, where, and when VMs should be migrated for the adjacent time periods. Suppose that there are m VMs hosted by a PM and n other PMs can be migrated onto in t periods of time; the number of possible migration schemes could be $n^{m\times t}$, which leads to the burden and curse of dimensionality for decision-making. Consequently, we first propose a hybrid resource pool to limit VM migrations between two PMs in this pool (to reduce VM migration planning's complexity), and then design an efficient resource-reuse optimization mechanism (HirePool) based on that, to improve physical resource usage.

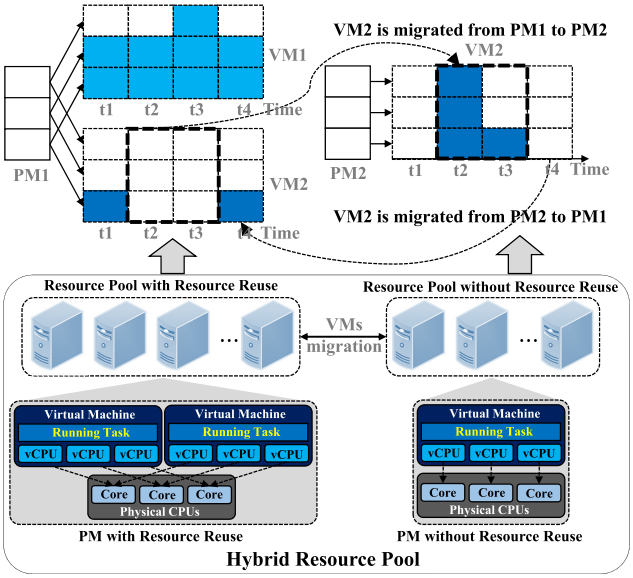


FIGURE 4. Hybrid resource pool model (a vCPU is a virtual CPU).

A. HYBRID RESOURCE POOL MODEL

Fig.4 shows the proposed hybrid resource pool model. The physical resource in the cloud datacenter is divided into a resource pool with resource reuse (named rePool whose OR>1), and a resource pool without resource reuse (named norePool, whose OR=1). In this paper, we appoint rePool to play the lead in this hybrid resource pool, and VMs are deployed on PMs in rePool acquiescently to maximize resource usage. For each PM in rePool, if the aggregate VMs resource usage exceeds its resource capacity, some VMs running on it will be migrated onto PMs in norePool to ensure that all application tasks can be executed smoothly. As time passes, once the PM's actual physical capacity can satisfy the demands of VMs that have been migrated to norePool, these VMs will be recalled and redeployed on their former PM.

B. HIREPOOL

Here, in the process of HirePool, each active PM in rePool is the optimized object, and the migration planning procedure only determines how many VMs on it will be migrated to norePool at each time period. Thus, it decreases the number of VMs that should be migrated and limits the scope of VM migration decision to two PMs belonging to rePool and norePool, respectively. In this way, it lowers the solution-space complexity of VM migration planning decision-making from $n^{m\times t}$ to $2^{m\times t}$, where, the meanings of variable n, m, t are the same with description of the first paragraph in Section IV.

1) OPTIMIZATION OBJECTIVES

HirePool realizes cloud resource-reuse optimization objectives from two perspectives.

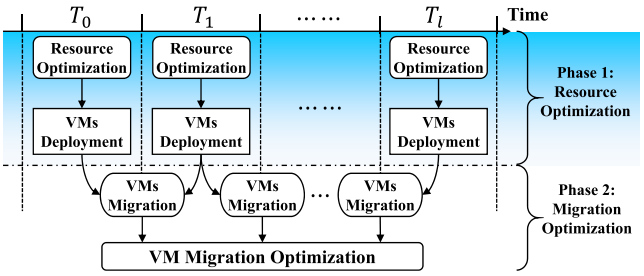


FIGURE 5. Resource-reuse optimization for a future *l* period in time T_0.

- **Resource optimization.** This entails maximizing VMs' resource usage in the hybrid resource pool. If a VM's maximum resource requirement is given, increasing the resource usage reveals that the objective is to minimize the physical resource occupied by the VM.
- **Migration optimization.** This entails minimizing the number of VM migrations between two resource pools based on the premise that a resource optimization objective is guaranteed. The reason why is that too many VM migrations could lead to degraded system performance or crashed VMs.

2) OPTIMIZATION PROCEDURES

Considering the two aforementioned optimization objectives, HirePool optimizes resource reuse in two phases, as Fig.5 shows.

Phase 1 (Resource Optimization): HirePool processes resource optimization in each time period, and achieves current global optimal deployments for all VMs according to their resource demands while the resource optimization is assured.

Phase 2 (Migration optimization): HirePool optimizes all VM migration from the perspective of an entire time period, and selects the optimal deployment from all deployments for each time period to minimize the number of migrations.

C. MATHEMATICAL MODELING

Based on the aforementioned hybrid resource pool model, the mathematical model of a PM's resource reuse problem in `rePool` is defined as follows. Notations used in this paper are listed in Table 1. We assume that the number of physical CPUs of this PM is N_{pc}, there are m VMs deployed on it, and $c_i (i = 1, 2, \ldots, m)$ represents the number of virtual CPUs owned by VM i. There are m tasks scheduled to this PM, and each task is serviced by a VM. Considering the number of CPUs requested by tasks in the next T time periods, we use $r_i(t) \in [1, c_i](t = T_0, \ldots, T_0 + T, i = 1, 2, \ldots, m)$ to denote the CPU number requested by task i in t time period, and use $x_i(t)(\in 0, 1)$ to express the deployment plan of VM i in t time period. Herein, $x_i(t) = 0$ represents that the VM is supposed to be deployed on a PM belonging to `rePool` and $x_i(t) = 1$ represents that the VM is supposed to be deployed on a PM of `norePool` in this time period. We use

TABLE 1. Notations used.

Notation	Explanation
N_{pc}	the number of physical CPUs of a physical machine
c_i	the number of virtual CPUs owned by the i-th VM
$r_i(t)$	the CPU number requested by task i in t time period
$x_i(t)$	the deployment plan of VM i in t time period
$s(t)$	a VM's deployment plan in t time period
$MC(s_1, s_2)$	the migration cost function, which means the number of migrations by adjusting deployment s_1 to s_2
$U(s(t))$	the objective function of the resource optimization problem as a knapsack problem
I_n	the set of VMs with their $x_i(t) = 1$
E_n	the set of VMs with their $x_i(t) = 0$
U_n	the set of VMs whose $x_i(t)$ is uncertain

$s(t) = (x_1(t), \ldots, x_m(t))$ to represent a VM's deployment plan in the current time period. Based on the optimization objectives mentioned in Section IV-B.1, these two optimization problems are defined as follows.

Definition 2: (Resource optimization problem). We use $r_i(t)$ to represent the resource requirement of each VM in period of time t, and we need to minimize the physical resources consumed by those deployed VMs to maximize resource utilization. The optimal deployment set is defined as $S(t) = \{s_1(t), s_2(t), \ldots\}$, which consists of all the VMs' allocation candidates that can satisfy the optimization objective. The resource optimization problem can be formally defined as

$$S(t) = \arg\min_{s(t)} \left(U(s(t)) = N_{pc} + \sum_{i=1}^{m} x_i(t) c_i \right) \tag{1}$$

$$\text{s.t.} \sum_{i=1}^{m} (1 - x_i(t)) r_i(t) \leq N_{pc} \tag{2}$$

$$x_i(t) = 0 \text{ or } 1 \quad (i = 1, 2, \ldots, m) \tag{3}$$

Formula (1) is the optimization objective, which means the consumed CPU resource in the hybrid resource pool after optimizing VM deployment (N_{pc} represents the resource in `rePool`, and $\sum_{i=1}^{m} x_i(t) c_i$ represents the resource in `norePool`). Formula (2) is the constraint, which means the aggregate VMs' resource usage cannot exceed the available PMs' physical capacity belonging to `rePool`.

Definition 3: (Migration optimization problem). Given the optimal migration set $S(t) = \{s_1(t), s_2(t), \ldots\}$ in each period of time t, we need to pick out the final deployment plan for each period of time $t(s_f(t) \in S(t))$ to minimize the number of VM migrations. The migration optimization problem can be formally defined as

$$\min \sum_{t=T_0}^{T_0+T-1} MC(s_f(t), s_f(t+1)) \tag{4}$$

$$MC(s_1, s_2) = \sum_{i=1}^{m} \left| x_i^1 - x_i^2 \right| \tag{5}$$

where $MC(s_1, s_2)$ is the migration cost function, which means the number of migrations by adjusting deployment s_1

to s_2. If $s_1 = (x_1^1, \ldots, x_m^1)$, $s_2 = (x_1^2, \ldots, x_m^2)$, then we can determine $MC(s_1, s_2)$ by (5).

D. IMPLEMENTING ALGORITHMS

HirePool adopts two optimization algorithms to solve the resource and migration optimization problems presented in Section IV-C, respectively.

1) RoA: RESOURCE OPTIMIZATION ALGORITHM

RoA uses a binary search tree (BST) based on the branch and bound (B&B) method to determine optimal VM migration planning. To expedite the B&B search, RoA formulates the resource optimization problem as a knapsack problem and derives the objective function $U(s(t))$'s optimal value using the dynamic programming (DP) method. Then, RoA can optimize the search process by using this value.

a: OBTAIN THE OPTIMAL VALUE OF THE OBJECTIVE FUNCTION

To get the minimum of objective function $U(s(t))$, RoA constructs a function $U'(s(t)) = \sum_{i=1}^{m}(1 - x_i(t))c_i$, and then transfers $U(s(t))$ as follows:

$$U(s(t)) = N_{pc} + \sum_{i=1}^{m} x_i(t)c_i \tag{6}$$

$$= N_{pc} + \sum_{i=1}^{m} c_i - \sum_{i=1}^{m}(1 - x_i(t))c_i \tag{7}$$

$$= N_{pc} + \sum_{i=1}^{m} c_i - U'(s(t)). \tag{8}$$

Obviously, the minimum of $U(s(t))$ equals the maximum of $U'(s(t))$. If $y_i = 1 - x_i(t)$, then the resource optimization problem can be formulated to the 0-1 knapsack problem defined elsewhere (9), (10), (11). Here, c_i is the objects' value, $r_i(t)$ is the objects' weight, and N_{pc} is the knapsack's capacity.

$$\max \quad U'(s(t)) = \sum_{i=1}^{m} y_i \times c_i \tag{9}$$

$$\text{s.t.} \sum_{i=1}^{m} y_i \times r_i(t) \le N_{pc} \tag{10}$$

$$y_i = 0 \text{ or } 1, \quad (i = 1, \ldots, m). \tag{11}$$

Then, RoA uses the DP algorithm [27] to solve the knapsack problem, and gets the minimum M' of $U'(s(t))$. Next, the minimum of $U(s(t))$ is calculated in (12).

$$M = N_{pc} + \sum_{i=1}^{m} c_i - M' \tag{12}$$

b: SEARCH FOR THE OPTIMAL SET OF VM DEPLOYMENT

To determine the optimal VM deployment $(S(t) = s_1(t), s_2(t), \ldots)$ with the corresponding minimum M, RoA builds a BST based on the B&B search method and uses M as a pruning bound to speed up the B&B search process.

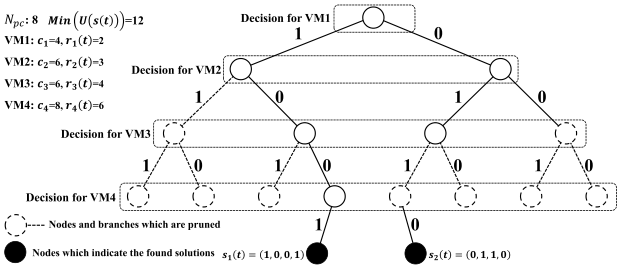


FIGURE 6. Binary search tree based on the branch and bound (B&B) method.

Building BST: RoA will build an $(m + 1)$-layer BST for the resource optimization problem of m VMs. The 2^m leaves in this BST means 2^m possible VM deployment schemes. Each branch node in BST represents a deployment scheme set (corresponding to the subtree leaf nodes) whose root is that node. The i^{th} layer branch in BST denotes the deployment decision of the i^{th} VM. If a branch's decision variable $x_i(t) = 1$, it means that the VM is deployed in `norePool`, else if $x_i(t) = 0$, the corresponding VM will be placed in `rePool`. Fig.6 shows a BST with four VMs.

Calculating Bound: Suppose the current branch node is n; we use (I_n, E_n, U_n) to represent the set of deployment for node n. Here, I_n denotes the set of VMs with their $x_i(t) = 1$, E_n represents the set of VMs with their $x_i(t) = 0$, and U_n delegates the set of VMs whose $x_i(t)$ is uncertain. The minimum objective function's lower bound corresponds to node n, which we solve in (13).

$$B(n) = N_{pc} + \sum_{i=1}^{m} c_i - B'(n), \tag{13}$$

where $B'(n)$ is the upper bound of the maximum of corresponding function $U'(s(t))$ for the current set of deployment (I_n, E_n, U_n). We can set $y_i = 1 - x_i(t)$ and can get $B'(n)$ by solving the following function:

$$\max \quad U'(s(t)) = \sum_{i=1}^{m} y_i \times c_i \tag{14}$$

$$\text{s.t.} \sum_{i=1}^{m} y_i \times r_i(t) \le N_{pc} \tag{15}$$

$$y_i = 0, \quad i \in I_n \tag{16}$$

$$y_i = 1, \quad i \in E_n \tag{17}$$

$$0 \le y_i \le 1, \quad i \in U_n. \tag{18}$$

Considering the aforementioned function, we adopt a method similar to [28] to get $B'(n)$. First, for all VMs in the set U_n, we sort their value of $c_j/r_j(t)$ in descending order to reconstruct it as $U_n = [1, \ldots, j, \ldots]$. Then we can calculate s by (19). Finally, we obtain $B'(n)$ in (21).

$$3s = \min \{k \in U_n : \sum_{j=1}^{k} r_j(t) > N_{pc} - \sum_{i \in E_n} r_i(t)\} \tag{19}$$

Algorithm 1 RoA: Resource Optimization Algorithm

Require:
N_{pc}: the number of physical CPUs;
c_i: the number of virtual CPUs utilized by VM i;
$r_i \in [1, c_i]$: the number of virtual CPUs used by VM i;
M: the minimum of object function $U(s(t))$;

Ensure:
$S(t) = \{s_1(t), s_2(t), \ldots\}$: the set of optimal deployment;

1: $V = [1, \ldots, i, \ldots] \leftarrow$ Sort all VMs by the value of $c_i/r_i(t)$ in descending order;
2: Build an empty queue Q and add the binary tree's root node into Q;
3: **while** $Q \neq \varnothing$ **do**
4: Pick out a branch node $n = (I_n, E_n, U_n)$ from Q;
5: Calculate the bound value $B(n)$;
6: **if** $B(n) \leq M$ **then**
7: $O \leftarrow N_{pc} + \sum_{i \in I_n} c_i$;
8: $R \leftarrow \sum_{i \in E_n} r_i(t)$;
9: $R' \leftarrow \sum_{i \in E_n \cup U_n} r_i(t)$;
10: **if** $O == M$ and $R' \leq N_{pc}$ **then**
11: Find out an optimal deployment solution and add this solution to $S(t) : \{x_i(t) = 1 | i \in I_n, x_i(t) = 0 | i \in E_n \cup U_n\}$;
12: **else if** $R \leq N_{pc}$ **then**
13: Create two new child nodes $(I_n + \{i\}, E_n, U_n - \{i\})$, $(I_n, E_n + \{i\}, U_n - \{i\})$ according to the layer i of node n, and add them into Q;
14: **end if**
15: **end if**
16: **end while**
17: **return** $S(t)$;

$$b = N_{pc} - \sum_{i \in E_n} r_i(t) - \sum_{j=1}^{s-1} r_j(t) \tag{20}$$

$$B'(n) = \sum_{i \in E_n} c_i + \sum_{j=1}^{s-1} c_j + b(c_s/r_s(t)). \tag{21}$$

B&B Search: At the beginning, we sort the value of $c_i/r_i(t)$ in descending order for all VMs that will be migrated, and then take a breadth-first search in the BST according to the value sorted previously. That is, we will make migration decisions for VMs so that their $c_i/r_i(t)$ is maximal in BST's first branch nodes, and make decisions for VMs so that their $c_i/r_i(t)$ is second-maximal in BST's second branch nodes, and so on. In the actual BST search process, RoA will determine whether the node branches should be pruned according to the bound calculated before. **Algorithm 1** gives a detailed description of RoA.

As lines 5 and 6 of Algorithm 1 show, if the current node's bound value exceeds the objective function's maximum, the node's branch will be pruned. If a current deployment is optimal, as line 10 of RoA shows, there is no need to search for subsequent branches of this node, and the solution will be added to $S(t)$. Algorithm 1's line 12 is used to judge whether the current solution is a feasible option; if not, it will be pruned. As Fig.6 shows, solid lines represent the branches and nodes that actually must be searched, so we can get two optimal deployment solutions $(s_1(t) = (1, 0, 0, 1))$ and $(s_2(t) = (0, 1, 1, 0))$ by searching part of the BST. Essentially, this means that the optimal VM deployment solutions $S(t) = \{s_1(t), s_2(t), \ldots\}$ correspond to the optimal value of objective functions obtained by RoA.

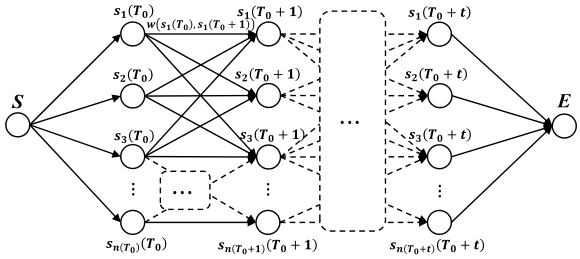


FIGURE 7. Migration cost graph example of t time periods.

2) MoA: MIGRATION OPTIMIZATION ALGORITHM

MoA will construct a migration cost graph to depict the number of VM migrations generated by different VM deployment schemes. In a migration cost graph, each selection path from the source node to the target node represents a VM deployment scenario, and the path's total weight is equal to the number of VM migrations caused by this deployment scenario. To determine an optimal VM deployment, MoA uses Dijkstra's algorithm to find the shortest path in a migration cost graph.

a: CONSTRUCTING A MIGRATION COST GRAPH

In this paper, we use a directed weight graph to describe the migration optimization problem. To solve the migration optimization problem, we assume we must determine the final VMs deployment scheme $s_f(t) \in S(t)$, where $t \in (T_0, \ldots, T_0 + T)$, for each time period t within T time intervals. For each t, MoA discerns the set of optimal deployment solutions $S(t) = \{s_1(t), s_2(t), \ldots\}$, and the number of elements of it is $n(t) = |S(t)|$. Then, we construct a migration cost graph with $t+2$-row nodes, as Fig.7 shows. The first row of the graph only has one root node S, the $(t+2)^{th}$ row of the graph only has one objective node E, and there are $n(t)$ nodes in other t rows, respectively. We use $\{s_1(t), s_2(t), \ldots, s_{n(t)}(t)\}$ to represent them. There is no path between the nodes in the same row, and a path only exists between nodes in different rows. We define the path's weight using the following rules: (1) a path's weight is 0 if it starts with node S or ends up with node E, and (2) a path's weight is $w(s_i(t), s_j(t+1)) = MC(s_i(t), s_j(t+1))$ according to (5), if it starts with node $s_i(t)$ and node $s_j(t+1)$, respectively. The path from root node S to objective node E in the migration cost graph represents a

VM deployment solution in T time intervals, and the key to MoA is to find the shortest path in this migration cost graph.

b: SOLVING A MIGRATION OPTIMIZATION PROBLEM

Once we have the migration cost graph, we can find the optimal VM deployment solution. The basic idea of MoA is to find the shortest path that corresponds to an optimal deployment scheme using Dijkstra's algorithm. Then, MoA can determine which VM should migrate between `rePool` and `norePool` in the adjacent time interval. Algorithm 2 is a detailed description of MoA.

Algorithm 2 MoA: Migration Optimization Algorithm

Require:
G: the graph of migration cost;
S: the root node;
E: the objective node;
Ensure:
F: the final deployment solution $(= \{s_f(T_0), s_f(T_0 + 1), \ldots, s_f(T_0 + t)\}$, that is, the shortest path);

for each vertex v in G **do**
  $d[v] \leftarrow$ infinity;
  $previous[v] \leftarrow$ undefined;
end for
$d[s] \leftarrow 0$;
$Q \leftarrow$ set of all vertices;
while $Q \neq \varnothing$ **do**
  $u \leftarrow$ vertex in Q with min $d[u]$;
  Remove u from Q;
  for each neighbor v of u **do**
    $alt \leftarrow d[u] + w(u, v)$;
    if $alt \leq d[v]$ **then**
      $d[v] \leftarrow alt$;
      $pre[v] \leftarrow u$;
    end if
  end for
end while
$F \leftarrow$ empty sequence;
$u \leftarrow E$;
while $pre[u]$ is defined **do**
  Insert u at the beginning of F;
  $u \leftarrow prev[u]$;
end while
Insert u at the beginning of F;
return F;

V. DEPLOYING THE RESOURCE-REUSE MECHANISM IN THE REAL WORLD

There are still issues to consider when deploying the mechanisms mentioned in this paper in the real world. Here, we detail these problems and extend the resource-reuse mechanism to solve them.

1) The problem of placing VMs on PMs in a resource pool: When a VM needs to migrate from a resource pool with resource reuse to a resource pool without resource reuse, the VM will be deployed in a resource pool without resource reuse, using a VM placement algorithm named FirstFit [29]. Specifically, this method is to deploy the VM on the first PM that can run it. When a VM needs to migrate from a resource pool without resource reuse back to a resource pool without resource reuse, it means that the PM (on which the VM was deployed in the reuse resource pool) can run it. So, the VM will migrate to the original PM in the resource pool.
2) The problem of the number of tasks running in a VM: To easily describe and model this problem, we assume that only one task runs in each VM. But in practice, multiple tasks can run on a VM concurrently. In the actual optimization, you only need to think of these tasks as a logical task abstractly, then you can still use the method we propose to optimize.
3) The problem of VMs' varying resource requirements: In practice, we first employ a method [7] that analyzes tasks' resource requirements of tasks in the datacenter to get the resource requirement model when each type of task is running. In the subsequent optimization, whenever a user submits a new task, we gain resource requirement models according to the type of task that lead to resource requirement changes.
4) The problem of triggering the resource-reuse optimization algorithm: The system performs the new resource-reuse optimization according to new forecasts of the VM resource requirements at each specified time interval.

VI. PERFORMANCE EVALUATION

In this section, we perform a set of experiments in both simulation and real cloud platform environments, to evaluate HirePool's feasibility and efficiency. In the simulation experiment, the dataset we use for tasks' resource requirements comes from Google Cluster Data [30]. In the real platform experiment, the dataset we use comes from AMS experimental scientific computing tasks' resource requirements, which are produced in Southeast University's cloud datacenter [31].

A. COMPARISONS AND INDICATORS

In the experiment, we compare the performance of **HirePool** with five mechanisms: NRR, NMRS, Markov, PCC, and PPR.

- **NRR:** (No resource reuse). In this scenario, all VMs are deployed on PMs without using resource overcommitment technology, and the VM's peak resource requirement is reserved for the entire execution time. That is, NRR allocates physical resources to VMs only once, based on static VM resource demands.
- **NMRS:** (No migration resource sharing). This approach adopts resource overcommitment technology to realize resource reuse and maximize resource usage during VM deployment, similar to previous work [7]. However,

the VM migration mechanism is not applied to further optimize NMRS's resource reuse performance.

- **Markov:** This method also uses resource overcommitment technology to reuse physical resources during VM deployment, and adopts the Markov decision mechanism used in [12] and [13] to plan the VM migration by considering variations in VM resource requirements.
- **PCC:** (Predictive control for consolidation). This mechanism uses resource overcommitment technology, too, and the main idea of PCC can refer to [21]. It handles the variations in VM resource requirements via live VM migrations, by formulating the problem as a control optimization problem based on integer programming, and then uses a Monte Carlo optimization method to approximate a solution.
- **PPR:** (Performance-to-power ratio aware). This method adopts our hybrid resource pool's architecture. The difference between this method and HirePool is that PPR uses the VM migration scheme presented by [15], which migrates VMs by using a greedy algorithm. But this method cannot optimize the number of VM migrations and may significantly degrade the service response time of some or all applications.

In the process of conducting the experiments, we measured four performance indicators.

- **ANP:** (Average number of PMs used). This is the average number of PMs required for each period time.
- **ARU:** (Average resource usage). This is the average resource use of all PMs in the datacenter for each time period.
- **ANM:** (Average number of VM migrations). This is the average number of migrations for each VM for each time period.
- **ACT:** (Average computing time). This is the average computing time required for each resource-reuse optimization.

B. SIMULATION EXPERIMENTS

1) SIMULATION SETTINGS

In simulation experiments, we used CloudSim [35] to stimulate a cloud datacenter environment. Each PM in the datacenter provided 24 physical CPUs, 24GB RAM and 1TB hard disk storage. We divided the VMs deployed on the PM into three types with different virtual CPU cores, and the CPU cores are 4, 6, and 8, respectively, but the same with 2GB memory and 100GB hard disk storage. Resource overcommitment technology is not supported by PMs in `norePool`; that is, 24 physical CPUs of each PM can only be virtualized into 24 virtual CPUs. However, the 24 physical CPUs of each PM in `rePool` can be virtualized into several virtual CPUs according to its `OR` (for example, if `OR`=2, there are 48 virtual CPU cores that can be virtualized and provided by such a PM). After the tasks are submitted, they are scheduled to a free VM (if there is one) by the task scheduler, or else they are arranged into a waiting queue until a VM is released.

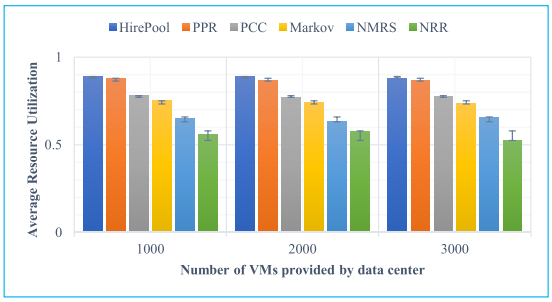


FIGURE 8. Comparison of PMs' average resource usage.

Resource requirements loads are generated based on the data in the Google Cluster dataset [30] when tasks are running. Using the method proposed in [7], the future resource requirements of each task are obtained according to its type. We simulated 100 hours of resource usage in a cloud datacenter, where the length of each time period is set as 10 minutes for each VM migration planning.

2) EXPERIMENTAL RESULTS AND ANALYSIS

a: RESOURCE OPTIMIZATION PERFORMANCE

We evaluated the ANP and ARU of the six resource optimization mechanisms for comparison (described in Section VI-A) in a datacenter containing 1,000, 2,000, and 3,000 VMs, respectively, and the `OR` of PMs in `rePool` was set as 2. Fig.8 shows the experimental results, where the horizontal coordinates represent the number of VMs provided by the datacenter, the vertical coordinates represent the ARU of all PMs in the datacenter in each time period, and the error bars show the standard deviation with 90% confidence. According to Fig.8's results, we see that four resource-reuse mechanisms can achieve better resource optimization compared to NRR without resource reuse, that is because the cloud's physical resource usage can be maximized by over-commit technology. Then, compared to NMRS (without VM migration), we find that the resource-reuse mechanisms (Markov, PCC, PPR, and HirePool) that support VM migration effectively improve resource-reuse efficiency. Finally, we observe that HirePool offers the best resource optimization performance, and it improves the ARU by 13% and decreases the ANP by 12% (compared to the Markov and PCC approaches that also consider resource and migration optimization). So, the results verify that HirePool effectively improves resource usage and optimizes resource-reuse efficiency. Furthermore, although these results show that PPR also achieves better resource optimization performance, it may lead to longer VM migration times due to lack of migration optimization (as confirmed in the following tests).

b: VM MIGRATION COST

Using the same settings as the aforementioned experiments, we also tested VM migration costs that is, the number of VM migrations caused by using resource overcommitment for HirePool, PPR, Markov, and PCC. Fig.9 shows the results,

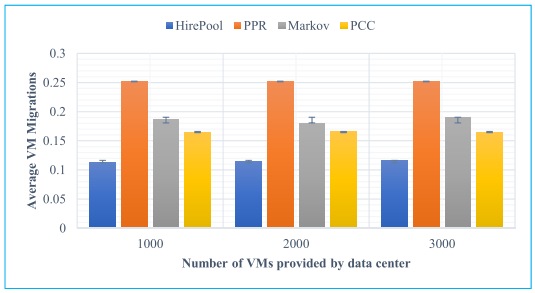


FIGURE 9. Comparison of average VM migrations.

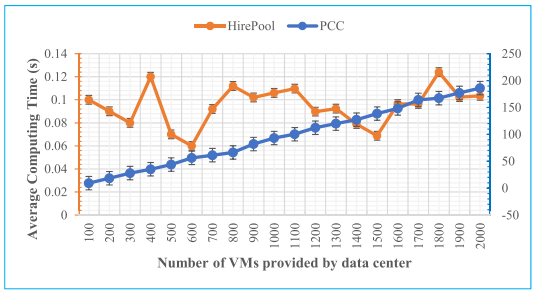


FIGURE 10. Comparison of computing time for planning.

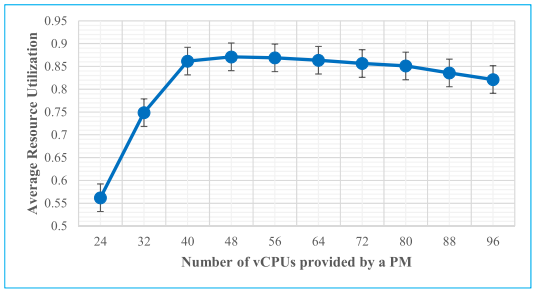


FIGURE 11. Influence of varying resource overcommitment ratios on the average resource usage (ARU).

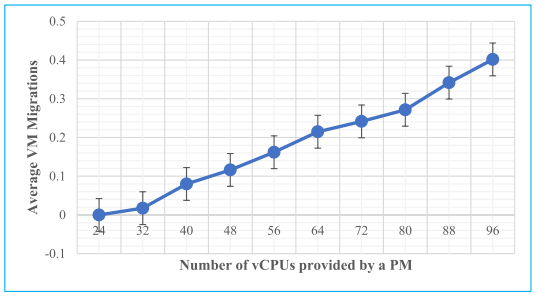


FIGURE 12. Influence of varying resource overcommitment ratios on the average number of VM migrations (ANM).

where the horizontal coordinates represent the number of VMs provided by the datacenter, the vertical coordinates represent the ANM of each VM for each time period, and the error bars show the standard deviation with 95% confidence. From these results, we see that HirePool effectively reduces the number of VM migrations while simultaneously ensuring the resource-reuse effects. Specifically, compared to PCC, HirePool reduces the ANM by 31% and has little impact on the user experience. This is mainly due to the VM migration planning carried out by HirePool, which reduces the number of VM migration as much as possible.

c: COMPUTATIONAL TIME COST

To validate whether HirePool can effectively reduce the complexity of VM migration planning, we evaluated the ACT of making decisions for VM migration planning between HirePool and PCC. Specifically, we compared the calculation time spent by these two mechanisms on migration planning decision-making when the number of VMs in the datacenter increases from 100 to 2,000 in simulation experiments, and each PM can deploy 10 VMs. Fig.10 shows the results, where the abscissa represents the number of VMs provided by the datacenter, the ordinate represents the average computing time required for each migration plan made by the two mechanisms on different scales of the datacenter, and the error bars show the standard deviation with 95% confidence. Because the ACT spent by the two mechanisms, respectively, has a relatively large gap, the ordinate is marked with a logarithmic scale. Overall, the ACT spent by the proposed mechanism seems quite low approximately 0.1 second and does not change with the expansion of VMs' scale. The reason why is that HirePool only concerns itself with VM migration planning on a single PM, and the scope of migration is limited to two PMs belonging to `rePool` and `norePool`, respectively, so HirePool can solve the migration-planning decision-making problem quickly. As Fig.10 shows, the computing time of PCC increases as the datacenter's scale increases. This is because PCC's migration plan services all datacenter's VMs, so when with the datacenter's size increases, the scale for VM migration-planning decision making also increases although an approximation algorithm based on Monte Carlo optimization was used.

d: RESOURCE OVERCOMMITMENT RATE EFFECT

Here, we attempt to validate the influence of `OR` on the performance of resource reuse (such as ARU and ANM). As defined, a PM can provide a different number of virtual CPUs (vCPUs) with different `OR`, and we did simulation experiments on a PM with 24, 32, . . . , 80, 88, and 96 vCPUs, respectively. Fig.11 and Fig.12 show the experimental results, where the abscissa represents the number of vCPUs that can be provided by a single PM at different `OR`, and the ordinate of Fig.11 represents the ARU while the ordinate of Fig.12 represents the ANM. All the error bars show the standard deviation with 90% confidence.

In Fig.11, the ARU shows a trend with a large increase at the beginning and then a small decrease when `OR` increases. We think this upward trend occurs when $\texttt{OR} > 1$, because the more that VMs reuse physical resources, the better, particularly if there are enough physical resources leftover. The downward trend occurs when the exaggerated `OR` leads to

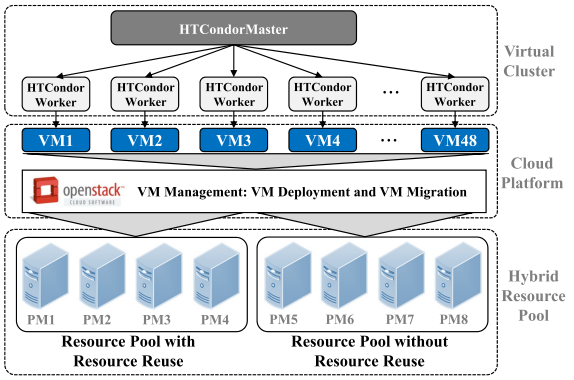


FIGURE 13. Test cloud platform's architecture.

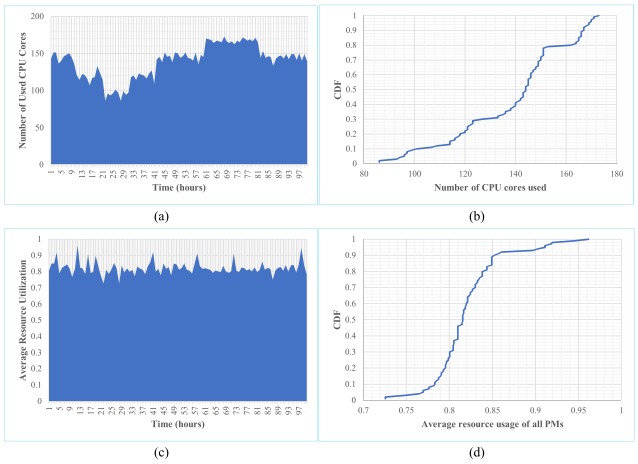


FIGURE 14. (a) Number of CPU cores used by all VMs; (b) Empirical CDF of number of CPU cores used; (c) Average resource usage of all PMs used; (d) Empirical CDF of average resource usage.

too many VMs competing for physical resources, so that many VMs end up migrating to `norePool`, and consequently the whole ARU decreases. Additionally, we see in Fig.11 that when a single PM provides 40-56 vCPUs (that is, $1.66 \leq \texttt{OR} \leq 2.33$), the ARU reaches the maximum. This is because the aggregate resource requirements of all VMs deployed on a single PM are consistent with the PM's physical resource capacity, which is an optimal equilibrium for HirePool. Moreover, we see in Fig.12 that the ANM increases as `OR` increases. The reason why is that when the resource overcommitment rate increases, the possibility increases (during a given time period) for the sum of all the VMs' resource requirements to become greater than the PM's capacity, which may lead to an increase in the number of VMs that must migrate to the `norePool`.

C. REAL PLATFORM EXPERIMENTS

1) CLOUD PLATFORM SETTINGS

Except for the aforementioned simulation experiments, HirePool is also validated by tests on Southeast University's real cloud datacenter platform (SEU) [32]. The platform is comprised of eight PMs, and each PM is IBM H22 Blade Server with 2*Six-Core Xeon® 5650 CPUs, 24GB of DDR3 RAM, and 146GB hard disk storage. These physical resources are managed by OpenStack as infrastructure as a service (IaaS) for applications. Each VM in this cloud platform has four vCPUs, 2GB memory and 12GB HD. There are four PMs in `rePool` and `norePool`, respectively, and each PM in `rePool` can provide 12 VMs and those in `norePool` can provide six VMs. We adopted HTCondor [24] to construct a VM cluster with 48 VMs, which all belong to `rePool`. Fig.13 shows the real cloud platform's architecture. The tasks executed in the VM cluster were an AMS-02 Monte Carlo simulation in Southeast University's Science Operation Center [31], and we used the HirePool mechanism to optimize resource reuse. In the experiments, we tested the ANP, ARU, and ANM changes when tasks ran on the VM cluster, and we also verified the influence of VM migration on the tasks' performance.

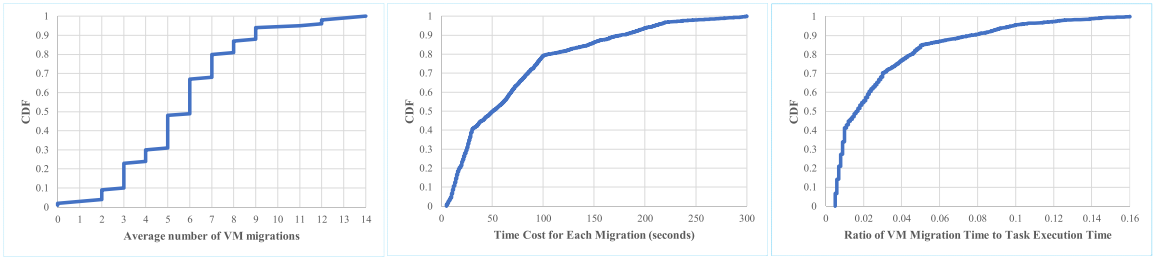
2) EXPERIMENTAL RESULTS AND ANALYSIS

a: RESOURCE OPTIMIZATION PERFORMANCE

During the real tests, the VM cluster ran with a consistently busy workload of 48 AMS Monte Carlo simulation production tasks. Once a task completed, another new task was submitted immediately. Fig.14-(a) and (b) show the change in the number of CPU cores used by all VMs in the cloud platform over 100 hours and its distribution. From them, we can see that the CPU cores requirements varied dynamically during the entire runtime, although the number of VMs did not change. And the number of CPU cores used is about the same probability in each range, and does not focus on a particular range. Fig.14-(c) and (d) show the average resource usage of all PMs running in the virtual cluster over 100 hours and its distribution; we see that no matter how the resource requirements change, the whole system ensures adequate resource usage, and the probability that ARU is above 0.8 is very high, indicating that the resource utilization rate of most PM is very high. So, from Fig.14, we can conclude that HirePool is an effective mechanism for resource-reuse optimization to dynamically adjust the number of PMs according to the changes in resource requirements. The reason is that HirePool can reuse physical resources in `rePool` when the load is low, and conversely, it can migrate VMs into `norePool` when the load increases.

b: MIGRATION OPTIMIZATION PERFORMANCE

Fig.15-(a) shows the distribution of average number of VM migrations in the real cloud platform when planning migration for each hour. The times of live VM migration can be limited to nine with 90% confidence, as depicted by the figure. We also tested the time cost for each VM migration. Fig.15-(b) shows the results, where the abscissa represents the time spent for VM migration, and the ordinate represents the distribution of the time cost for VM migrations. More than 80% time costs are within 100 seconds (similar to [33]



(a) (b) (c)

FIGURE 15. Empirical CDF of (a) average number of VM migrations; (b) time cost for VM migrations; (c) ratio of VM migration time to task execution time.

and [34]). Additionally, we tested the ratio of VM migration time to total execution time for each task; Fig.15-(c) shows the results, where the abscissa represents all possible ratios and the ordinate represents the distribution of the time ratio of VM migration to task execution. We see that the ratio of VM migration time to task execution time is less than 0.05 for more than 80% tasks, so we conclude that the impacts of VM migration during resource-reuse optimization on the performance of total tasks and the users' experience are inconsequential.

VII. CONCLUSIONS AND FUTURE WORK

Considering the dynamically changing resource requirements of VMs in cloud datacenters, this paper proposes an effective resource-reuse optimization mechanism based on a hybrid resource pool model, to improve the efficiency of resource reuse by maximizing resource utilization with fewer VM migrations. From this work, we can find out that for some problems which are difficult to be solved due to the large scale, the problem size can be reduced by limiting the constraint, so that the problem can be solved. As shown in this paper, if we consider the original migration planning problem directly, we will find the problem is very large (that is, $n^{m\times t}$) and difficult to solve. When we restrict that the VMs can only migrate between a `rePool` PM and a `norePool` PM, the problem scale will be greatly reduced to $2^{m\times t}$, so that the problem can be solved quickly.

Moreover, the virtual resource-reuse optimization mechanism proposed here is basic, functional, and effective. We plan to expand and improve upon this work, and continue research in three respects. First, changing the resource overcommitment ratio will influence the results of resource reuse and VM migration time, so we hope to propose a novel algorithm that determines the appropriate overcommitment ratio intelligently. Second, we only consider a one-dimensional CPU requirement in this paper, but we hope HirePool can adapt to multidimensional resource requirements (such as shown in [36]) in the future. Finally, we only consider the optimization of VM migration time in this work; in future work, we will consider migration communication costs, energy consumption, and other factors in migration-optimization modeling more fully.

REFERENCES

[1] M. A. Rodriguez and R. Buyya, "A responsive knapsack-based algorithm for resource provisioning and scheduling of scientific workflows in clouds," in *Proc. IEEE ICPP*, Beijing, China, Sep. 2015, pp. 839–848.

[2] C. Q. Wu, X. Lin, D. Yu, W. Xu, and L. Li, "End-to-end delay minimization for scientific workflows in clouds under budget constraint," *IEEE Trans. Cloud Comput.*, vol. 3, no. 2, pp. 169–181, Apr./Jun. 2015.

[3] Q. Jiang, Y. C. Lee, and A. Y. Zomaya, "Executing large scale scientific workflow ensembles in public clouds," in *Proc. IEEE ICPP*, Beijing, China, Sep. 2015, pp. 520–529.

[4] C. Delimitrou and C. Kozyrakis, "Quasar: Resource-efficient and QoS-aware cluster management," in *Proc. ACM ASPLOS*, New York, NY, USA, 2014, pp. 127–144.

[5] S. Mustafa, K. Bilal, S. U. R. Malik, and S. A. Madani, "SLA-aware energy efficient resource management for cloud environments," *IEEE Access*, vol. 6, pp. 15004–15020, 2018.

[6] S. Zhang, Z. Qian, Z. Luo, J. Wu, and S. Lu, "Burstiness-aware resource reservation for server consolidation in computing clouds," *IEEE Trans. Parallel Distrib. Syst.*, vol. 27, no. 4, pp. 964–977, Aug. 2016.

[7] L. Chen and H. Shen, "Consolidating complementary VMs with spatial/temporal-awareness in cloud datacenters," in *Proc. IEEE INFOCOM*, Toronto, ON, Canada, Apr./May 2014, pp. 1033–1041.

[8] S. Albagli-Kim, H. Shachnai, and T. Tamir, "Scheduling jobs with dwindling resource requirements in clouds," in *Proc. IEEE INFOCOM*, Toronto, ON, Canada, Apr./May 2014, pp. 601–609.

[9] H. Lin, X. Qi, S. Yang, and S. Midkiff, "Workload-driven VM consolidation in cloud data centers," in *Proc. IEEE IPDPS*, Toronto, ON, Canada, May 2015, pp. 207–216.

[10] S. Xu, W. Lin, and J. Z. Wang, "Virtual machine placement algorithm based on peak workload characteristics," (in Chinese), *J. Softw.*, vol. 27, no. 7, pp. 1876–1887, 2016.

[11] M. Li, J. Bi, and Z. Li, "Resource-scheduling-waiting-aware virtual machine consolidation," (in Chinese), *J. Softw.*, vol. 25, no. 7, pp. 1388–1402, 2014.

[12] L. Chen, H. Shen, and K. Sapra, "Distributed autonomous virtual resource management in datacenters using finite-Markov decision process," in *Proc. ACM SOCC*, Seattle, WA, USA, 2014, pp. 1–13.

[13] Z. Han *et al.*, "Dynamic virtual machine management via approximate Markov decision process," in *Proc. IEEE INFOCOM*, San Francisco, CA, USA, Apr. 2016, pp. 1–9.

[14] M. Dabbagh, B. Hamdaoui, M. Guizani, and A. Rayes, "Efficient datacenter resource utilization through cloud resource overcommitment," in *Proc. IEEE INFOCOM Wkshps*, Hong Kong, Apr./May 2015, pp. 330–335.

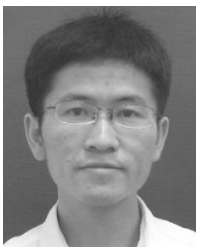
[15] X. Ruan and H. Chen, "Performance-to-power ratio aware virtual machine (VM) allocation in energy-efficient clouds," in *Proc. IEEE CLUSTER*, Chicago, USA, Sep. 2015, pp. 264–273.

[16] L. Chen, H. Shen, and K. Sapra, "RIAL: Resource intensity aware load balancing in clouds," in *Proc. IEEE INFOCOM*, Toronto, ON, Canada, Apr./May 2014, pp. 1294–1302.

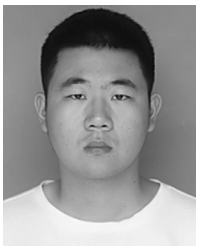
[17] L. Eyraud-Dubois and H. Larchevêque, "Optimizing resource allocation while handling SLA violations in cloud computing platforms," in *Proc. IEEE IPDPS*, Boston, MA, USA, May 2013, pp. 79–87.

[18] K. Ye *et al.*, "Profiling-based workload consolidation and migration in virtualized data centers," *IEEE Trans. Parallel Distrib. Syst.*, vol. 26, no. 3, pp. 878–890, Mar. 2015.

[19] S. Homsi, S. Liu, G. A. Chaparro-Baquero, O. Bai, S. Ren, and G. Quan, "Workload consolidation for cloud data centers with guaranteed QoS using request reneging," *IEEE Trans. Parallel Distrib. Syst.*, vol. 28, no. 7, pp. 2103–2116, Jul. 2017.
[20] V. Armant, M. De Cauwer, K. N. Brown, and B. O'Sullivan, "Semi-online task assignment policies for workload consolidation in cloud computing systems," *Future Gener. Comput. Syst.*, vol. 82, pp. 89–103, May 2018.
[21] M. Gaggero and L. Caviglione, "Predictive control for energy-aware consolidation in cloud datacenters," *IEEE Trans. Control Syst. Technol.*, vol. 24, no. 2, pp. 461–474, Mar. 2016.
[22] (2016). *YARN*. [Online]. Available: https://hadoop.apache.org/docs/r2.7.2/hadoop-yarn/
[23] (2016). *Mesos*. [Online]. Available: http://mesos.apache.org/
[24] (2016). *HTCondor*. [Online]. Available: http://research.cs.wisc.edu/htcondor/
[25] R. Ghosh and V. K. Naik, "Biting off safely more than you can chew: Predictive analytics for resource over-commit in IaaS cloud," in *Proc. IEEE CLOUD*, Honolulu, HI, USA, Jun. 2012, pp. 25–32.
[26] D. Breitgand, Z. Dubitzky, A. Epstein, A. Glikson, and I. Shapira, "SLA-aware resource over-commit in an IaaS cloud," in *Proc. ACM CNSM*, Laxenburg, Austria, 2013, pp. 73–81.
[27] P. Toth, "Dynamic programming algorithms for the zero-one knapsack problem," *Computing*, vol. 25, no. 1, pp. 29–45, 1980.
[28] P. J. Kolesar, "A branch and bound algorithm for the knapsack problem," *Manage. Sci.*, vol. 13, no. 9, pp. 723–735, 1967.
[29] X. Tang, Y. Li, R. Ren, and W. Cai, "On first fit bin packing for online cloud server allocation," in *Proc. IEEE IPDPS*, Chicago, IL, USA, May 2016, pp. 323–332.
[30] (2016). *Google Cluster Data*. [Online]. Available: https://github.com/google/cluster-data
[31] (2016). *AMS-02 Monte Carlo Simulation in Science Operation Center at Southeast University*. [Online]. Available: https://indico.cern.ch/event/505613/contributions/2227427/
[32] F. Dong, J. Luo, J. Jin, Y. Wang, and Y. Zhu, "Performance evaluation and analysis of SEU cloud computing platform—Using general benchmarks and real world AMS application," in *Proc. IEEE SMC*, Seoul, South Korea, Oct. 2012, pp. 1455–1460.
[33] U. Deshpande, D. Chan, T.-Y. Guh, J. Edouard, K. Gopalan, and N. Bila, "Agile live migration of virtual machines," in *Proc. IEEE IPDPS*, Chicago, IL, USA, May 2016, pp. 1061–1070.
[34] H. Liu, H. Jin, X. Liao, L. Hu, and C. Yu, "Live migration of virtual machine based on full system trace and replay," in *Proc. ACM HPDC*, Garching, Germany, 2009, pp. 101–110.
[35] *CloudSim: A Framework for Modeling and Simulation of Cloud Computing Infrastructures and Services*. Accessed: Sep. 18, 2016. [Online]. Available: http://www.cloudbus.org/cloudsim
[36] X. Guan, B.-Y. Choi, and S. Song, "Energy efficient virtual network embedding for green data centers using data center topology and future migration," *Comput. Commun.*, vol. 69, pp. 50–59, Sep. 2015.



RUNQUN XIONG received the Ph.D. degree in computer science from Southeast University. He was with the European Organization for Nuclear Research as a Research Associate for the AMS-02 experiment from 2011 to 2012. He is currently a Lecturer with the School of Computer Science and Engineering, Southeast University, China, where he is involved in AMS-02 data processing at the AMS Science Operations Center. His current research interests include cloud computing, industrial Internet, and drone-based wireless communication systems. He is a member of the ACM and the China Computer Federation.



XIUYANG LI received the B.S. degree in computer science and engineering from the Ocean University of China, Qingdao, China. He is currently pursuing the M.S. degree with the School of Computer Science and Engineering, Southeast University, China. His current research interests include drone-assisted wireless communications and mobile edge computing.

JIYUAN SHI received the Ph.D. degree in computer science from Southeast University, China. He was a Visiting Doctoral Student at the European Organization for Nuclear Research, Switzerland, from 2012 to 2014. He is currently an Algorithm Research Engineer at Huawei Technologies Co., Ltd. His current research interests include network big data analysis, scientific workflow scheduling, and resource management in cloud systems.

ZHIANG WU (S'14–M'17) received the Ph.D. degree in computer science from Southeast University, China. He is currently a Full Professor with the School of Information Engineering, Nanjing University of Finance and Economics, China. He is also the Director of the Jiangsu Provincial Key Laboratory of E-Business. His recent research focuses on distributed computing, data mining, e-commerce intelligence, and social network analysis. He is a member of the ACM and a Senior Member of the China Computer Federation.

JIAHUI JIN received the Ph.D. degree in computer science from Southeast University, Nanjing, China, in 2015. He was a Visiting Ph.D. Student with the University of Massachusetts, Amherst, from 2012 to 2014. He is currently an Assistant Professor with the School of Computer Science and Engineering, Southeast University. His research interests include large-scale data processing, distributed systems, and parallel task scheduling.