Joint Optimization of Operational Cost and Performance Interference in Cloud Data Centers

Xibo Jin[1,4], Fa Zhang[2], Lin Wang[1,4], Songlin Hu[2], Biyu Zhou[1,4] and Zhiyong Liu[1,3]
[1]Beijing Key Laboratory of Mobile Computing and Pervasive Device, ICT, Beijing, China
[2]Key Laboratory of Intelligent Information Processing, ICT, Beijing, China
[3]State Key Laboratory for Computer Architecture, ICT, Beijing, China
[4]University of Chinese Academy of Sciences, Beijing, China
Email: {jinxibo, zhangfa, wangling, husonglin, zhoubiyu, zyliu}@ict.ac.cn

***Abstract*—Virtual machine (VM) scheduling is an important technique for the efficient operation of the computing resources in a data center. Previous work has mainly focused on consolidating VMs to improve resource utilization and to optimize energy consumption. However, the interference between collocated VMs is usually ignored, which can result in much worse performance degradation of the applications running on the VMs due to the contention of the shared resources. Based on this observation, we aim at designing efficient VM assignment and scheduling strategies in which we consider optimizing both the operational cost of the data center and the performance degradation of the running applications. We then propose a general model that captures the tradeoff between the two contradictory objectives. We present offline and online solutions for this problem by exploiting the spatial and temporal information of performance interference of VM collocation, where VM scheduling is performed by jointly considering the combinations and the life-cycle overlap of the VMs. Evaluation results show that the proposed methods can generate efficient schedules for VMs, achieving low operational cost while significantly reducing the performance degradation of applications in cloud data centers.**

***Keywords*-Virtual machine scheduling; Operational cost; Performance degradation; Data centers**

I. Introduction

Cloud computing has become a promising option for modern computing platforms and will most likely continue to be the dominant service model in the future [1], [2], [3], [4], [5], [6]. Cloud computing takes advantage of virtualization technologies such as VMware [7] and Xen [8] to encapsulate applications into virtual machines (VMs) and allow independent applications to execute on the same physical server simultaneously. Furthermore, cloud computing affords users the abilities to obtain, configure, and deploy cloud services themselves using cloud service catalogs, without requiring the assistance of IT (Infrastructure Technology) [9], [10], [11]. The feasibility of VM consolidation and on-demand resource allocation offers an opportunity for cloud operators to multiplex resources among users and thus improve operational costs [12], [13], e.g., reduce energy consumption [14].

However, although it introduces better utilization to the cloud system, this type of resource multiplexing is not always beneficial. When VMs are consolidated together, the performance interference between the VMs introduced by the contention of shared resources, such as last-level-cache, memory bus, network and disk bandwidth, cannot be ignored [15], [16], [17], [18], [19], [20]. Compared to running on a dedicated server, a VM has to compete for shared resources with other VMs that are collocated with it, and thus, the performance will be degraded even with the same resource reservation. Whereas previous work has focused on the analysis of application-level interference in single servers, we aim to study the assignment and scheduling of VMs to physical servers, mitigating performance interference while optimizing the operational cost. We tackle this combinatorial problem of joint optimization by leveraging the specific structures of VM collocation.

A. Performance Interference inside a Cloud Data Center

It is necessary to provide efficient management of performance interference to guarantee quality of service for tenants in a cloud data center. In general, the performance interference between VMs can be affected by the following two factors.

VM combination. Recent studies have analyzed resource contention for possible VM combinations and have suggested the collocation of those VMs that exhibit less competition between shared resources [15], [16], [17], [18], [19], [20]. To quantify the overall performance interference between collocated VMs, we evaluated the performance degradation of VMs using the SPECcpu 2006 benchmark [21], where we assumed that each application executes in a VM and runs on a physical core. We define the *Performance Degradation Ratio* (PDR) of a VM as the increment of running time divided by the time used for the VM to be executed in a dedicated server. The statistical results are demonstrated in Table I. As shown in the table, the PDR of `429.mcf` when being collocated with `470.lbm` is 62.08%, whereas it is 11.90% when being collocated with `403.gcc`. This finding reveals that different VM combinations lead to a variable level of performance interference. As a result, VM placement can be achieved in an intelligent manner such that the performance interference between VMs is minimized. Another observation is that the PDRs of VMs become larger with the increase in the number of collocated VMs. For example, in Table I, the PDRs of `470.lbm` and `403.gcc` are 10.06% and 44.53% respectively, whereas these values increase to 16.29% and 67.55% when a third VM for `429.mcf` is launched simultaneously on the same physical server.

TABLE I: Running times (and stretches in percentile) of applications (VMs) collocated in the same physical server. (For example, the 1st row means each application runs on a dedicated server and the 3rd row means 403.gcc and 429.mcf collocate in a server.)

	401.bzip2	403.gcc	429.mcf	453.povray	470.lbm
401.bzip2/403.gcc/429.mcf/453.povray/470.lbm	498	265	269	186	318
401.bzip2 + 470.lbm	642 (28.92)	–	–	–	358 (12.58)
403.gcc + 429.mcf	–	**299 (12.83)**	**301 (11.90)**	–	–
429.mcf + 470.lbm	–	–	**436 (62.08)**	–	**366 (15.09)**
403.gcc + 453.povray	–	270 (1.89)	–	193 (3.76)	–
453.povray + 470.lbm	–	–	–	201 (8.06)	326 (2.52)
403.gcc + 470.lbm	–	**383 (44.53)**	–	–	**350 (10.06)**
403.gcc + 429.mcf + 470.lbm	–	**444 (67.55)**	**487 (81.04)**	–	**407 (16.29)**
401.bzip2 + 429.mcf + 470.lbm	725 (45.58)	–	482 (79.18)	–	404 (27.04)
401.bzip2 + 403.gcc + 429.mcf + 470.lbm	778 (56.22)	495 (86.79)	538 (100.00)	–	466 (46.54)

TABLE II: Running times (and stretches in percentile) of applications (VMs) collocated in the same physical server with different overlap times. (For example, the 1st grid means 429.mcf and 403.gcc collocate in a physical server with different overlap times, i.e., $+60$ means that 403.gcc starts after 429.mcf has run 60 unit times.)

Apps	0	+60	+120	+180
429.mcf	301 (11.90)	**294 (9.29)**	283 (5.20)	277 (2.97)
403.gcc	299 (12.83)	**292 (10.19)**	286 (7.92)	278 (4.91)
470.lbm	404 (27.04)	377 (18.55)	354 (11.32)	336 (5.66)
429.mcf	482 (79.18)	440 (63.57)	401 (49.07)	360 (33.83)
401.bzip2	725 (45.58)	643 (29.12)	578 (16.06)	523 (5.02)
470.lbm	466 (46.54)	407 (27.99)	357 (12.26)	337 (5.97)
429.mcf	538 (100.0)	476 (76.95)	418 (55.39)	365 (35.69)
403.gcc	**495 (86.79)**	**424 (60.00)**	**353 (33.21)**	**293 (10.57)**
401.bzip2	778 (56.22)	658 (32.13)	550 (10.44)	510 (2.41)

Life-cycle overlapping. It is a challenging problem to take into account the life cycles of VMs. On the one hand, overlapping the execution of VMs can improve the resource utilization of the system and thus reduce the marginal cost.[1] On the other hand, due to performance interference, reducing the overlap of the executions of VMs can mitigate performance degradation, thus shortening the completion times of VMs. This effect can be verified by the results shown in Table II. For example, when collocated with 470.lbm, 429.mcf, and 401.bzip2, 403.gcc experiences a considerable reduction in PDR from 86.79% to 10.57% with the decrease in the execution overlaps. Due to performance interference, the neglect of life-cycle overlap can result in more serious problems, such as resource-reservation violation caused by the extension of the execution duration of VMs. Moreover, the performance of some VMs will become unacceptably worse when the execution is always overlapped with other mutual-interference VMs; therefore, their performance will always be degraded by collocated VMs.

B. Tradeoff between Operational Cost and Performance Interference

In general, operational cost refers to the daily expenditure caused by the operation of a cloud computing system, including electricity cost and system maintenance expenses. Among these expenses, the cost of electricity constitutes a major proportion [22], [23]. As a consequence, achieving energy efficiency on servers can result in a significant reduction in the operational cost of a data center. For this reason, we will use the term *energy consumption* to refer to operational cost. Throughout the paper, we use both terms interchangeably. There is a large body of work focused on improving the energy efficiency of single servers, such as Dynamic Voltage Frequency Scaling (DVFS) and powering down [24]. Based on these two fundamental mechanisms, studies have investigated the reduction in the energy consumption of a cloud system using virtualization techniques such as VM consolidation to improve hardware utilization. However, although these methods can help attain the goal of energy conservation elegantly, very little attention has been paid to the accompanying side effect, i.e., performance interference. Moreover, to the best of our knowledge, a quantitative analysis of the tradeoff between energy consumption and performance interference is practically non-existent in the literature, which is highly desired by cloud operators.

We study the VM assignment and scheduling problem for arbitrating between energy consumption and performance interference, i.e., reducing energy consumption while maintaining low performance degradation for VMs. On the one hand, ideally, the energy consumption is minimized when a minimum number of servers is used. This minimization can be achieved by consolidating VMs and then switching idle servers to a power-saving mode (sleeping or power-off). The set of active servers is managed dynamically according to the workload. Consequently, the energy consumed by under-utilized servers can be saved, as can the corresponding cost incurred by power delivery and cooling infrastructure.

On the other hand, VM consolidation can result in undesirable performance interference between VMs due to the contention in shared resources. This performance interference can extend the execution durations of VMs to a large extent, which may introduce unacceptable performance losses to user applications (and further result in Service-Level-Agreement violations). A simple example is illustrated in Fig. 1. It can be observed that the assignment shown on the right side of the figure is better than the one shown on the left in two respects:

[1]This refers to the static cost irrespective of the load of a server incurred by always-on components, such as idle energy.

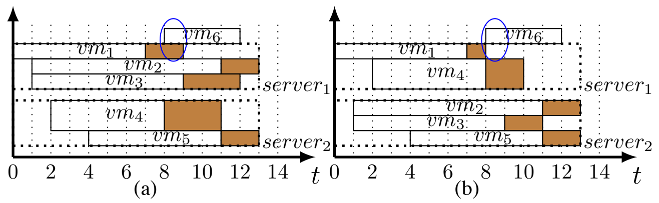

Fig. 1: Two methods of VM allocation. Assume the two servers have the same resource capacity of $\{\mathbf{1}\}$ unit and each VM from $\{vm_1, vm_2, vm_3, vm_5, vm_6\}$ demands $\{\frac{1}{3}\}$ unit while vm_4 requires $\{\frac{2}{3}\}$ unit. The rectangles in color represent the extension of the execution time due to resource contention, which represent the same mean in the following figures. (a): An inappropriate scheduling. (b): A better scheduling.

i) the performance of most VMs, such as vm_4, is degraded to a lower extent, and ii) the real-time accommodation of vm_6 without involving another server becomes possible. (As shown in ellipses.) The figure also reveals that the two factors, VM combination and life-cycle overlapping, are coupled and mutually affected. Therefore, to arbitrate between energy consumption and performance interference, it is necessary to provide a careful design of VM consolidation in which VMs are allocated in appropriate combinations and collocated VMs are scheduled with the most favorable life-cycle overlapping.

C. Overview of the Paper

In this work, we seek to formulate efficient solutions for reducing energy consumption while minimizing performance interference among VMs. Our main contributions are summarized as follows:

1) We characterize the energy consumption and the performance interference in a unified model and formulate the challenge of VM assignment and scheduling into an optimization problem. We also prove the NP completeness of the problem.
2) We propose efficient algorithms for both offline VM assignment and scheduling and online cases where dynamical arrival of the VMs can be managed.
3) We further reduce the energy consumption and improve the performance by using the batch arrival and resource reservation information. We also provide a distributed implementation of the algorithms for large-scale data centers to reduce the time required for information collection.
4) We evaluate the efficiency of the proposed algorithms through comprehensive simulations, showing that the proposed technique can obtain desirable performance while reducing energy consumption of the system.

The rest of this paper is organized as follows. In Section II, we summarize studies relevant to ours. Section III discusses the modeling of the problem. Section IV presents our algorithms for VM assignment and scheduling, in which both offline and online cases are considered, while a distributed solution is also provided. Section V validates the performance of the algorithms by extensive simulations. We finally conclude the paper in Section VI.

II. Related Work

With cloud computing becoming increasingly popular, researchers have conducted studies on executing traditional applications (e.g., *HPC and scientific computing*) in cloud environments. This section summarizes the research efforts undertaken in VM assignment and scheduling that are relevant to our work in terms of operational energy management and application performance interference in data centers.

Energy consumption management. It is known that the most efficient way to reduce energy consumption is by consolidating applications (VMs) into a set of active servers such that the utilization of the data center is maintained at a high level. An early study [25] extended virtualization solutions to support rich and effective policies for active power management, which had not been done before. The authors integrated "hard" and "soft" power states to provide high power savings and showed that substantial benefits could be derived from the coordination of online methods for server consolidation with their proposed management techniques. Kusic *et al.* [26] considered the problem of consolidating services into a smaller number of computing resources. The authors implemented a dynamic resource provisioning framework for virtualized server environments, which was tackled as a problem of sequential optimization and solved using a lookahead control scheme. Beloglazov *et al.* [27] investigated scheduling algorithms that consolidate VMs into the minimum number of servers. The authors proposed a policy known as Modified Best Fit Decreasing (MBFD) for energy-efficient management of cloud computing environments. In another representative work [28], the authors investigated the energy-saving problem by dynamically "right-sizing" the data center in both offline and online cases. Liu *et al.* [29] studied the problem of arbitrating the power-performance tradeoff in clouds. They provided a probabilistic framework in which online decisions were made on request for admission control, routing, and VM allocation. The authors modeled a unified objective to balance the power consumption of servers and the system throughput of application requests. Chen *et al.* [30] proposed consolidating complementary VMs into a fewer number of VMs to increase energy efficiency. In their work, spatial/temporal awareness represented the resource requirements in different resource dimensions over a given period. The authors considered this VM consolidation a classical d-dimensional vector bin-packing problem and proposed a dimension-aware heuristic algorithm to solve the problem. In contrast, we consider the spatial and temporal information of performance interference of VM collocation.

Thus, the aforementioned studies mainly focus on optimizing energy consumption or jointly considering the system throughput. However, in this study, we consider the performance interference of VM collocation and energy consumption to model the unified objective for VM assignment and scheduling.

Performance interference optimization. Several works [15], [16], [17], [18], [19], [20], [31] have taken into account performance interference when exploiting VM consolidation to improve resource utilization. Govindan *et al.* [15] presented a technique for predicting performance interference due to processor cache sharing. The authors showed that their technique can be used to achieve the most efficient consolidation because the prediction of performance degradation for any possible application placement requires only a linear number of measurements. Chiang *et al.* [16] considered the problem of interference-aware scheduling for data-intensive applications in virtualized environments. The authors presented a task and resource allocation control framework that can mitigate interference effects due to concurrent data-intensive applications and improve overall system performance. Mars *et al.* [17] presented a characterization methodology, named "Bubble-Up", that enables the accurate prediction of the performance degradation that results from contention for shared resources in memory subsystems. The authors showed that their methodology could predict the performance interference between collocated applications with an accuracy of 1% to 2% for the actual performance degradation. Roytman *et al.* [18] proposed a system that consolidates VMs to minimize unused resources and guarantees that the performance degradation is within a tunable bound. Their system employed a method for suitable VM combination that was demonstrated to perform almost optimally, and the system involved another technique that maximizes performance while not leaving any resource unused. Kim *et al.* [19] suggested a performance model that considers interferences in the shared last-level cache and memory bus. The authors claimed that the model could be used to estimate the performance degradation, among various applications. Based on the interference model, they also presented a VM consolidation method. Verboven *et al.* [20] addressed performance degradation prediction models and proposed a novel approach using both the classification and regression capabilities of support vector machines. A recent survey [31] discussed the state of the art of some of these solutions for managing the performance overhead in different cloud scenarios.

Compared with these previous works considering performance interference optimization, our model provides a unified characterization of both energy consumption and performance interference, and our solution for VM assignment and scheduling considers both VM combination and life-cycle overlapping. We explore the tradeoff between the performance degradation overhead of VMs and resource provision of cloud data centers on a high level, which is raised as an open research issue in [32]. Moreover, our work can be regarded as a complement to previous studies because the solutions provided by these studies can be integrated into our optimization framework to reduce the overall cost of a cloud system.

III. Model and Problem Description

In this section, we describe the proposed model and formulate an optimization problem of VM scheduling that aims at arbitrating between energy consumption cost and performance degradation penalty.

A. Resource Allocation and Energy Cost

We model a cloud data center as an undirected graph and denote it by $G = (\mathcal{M}, \mathcal{L})$, where $\mathcal{M}$ ($|\mathcal{M}| = M$) is the set of physical servers and $\mathcal{L}$ is the set of physical links between servers. Each server $server_i \in \mathcal{M}$ is associated with s types of resources, e.g., CPU, memory, and storage space. The resources of $server_i$ are available in $\boldsymbol{R}_i = \{C_{ik}\}$ ($k = 1, 2, ..., s$) units.

Recent studies [26], [33] have shown that the power consumption $P[u(t)]$ and the CPU utilization $u(t)$ of a server have a linear relationship

$$P[u(t)] = P_{idle} + (P_{peak} - P_{idle}) * u(t). \tag{1}$$

The P_{idle} and P_{peak} represent the power consumption of a server at CPU utilization rates of 0% and 100%, respectively. Clearly, the energy consumption of a server is its power integrated over the duration, i.e., $\int_t P[u(t)]\,\mathrm{d}t$.

B. VM Request and Interference

Cloud computing provides users with scalable, elastic and on-demand resources [34]. Users submit their VM requests to a cloud data center scheduler. Each VM request vm_j is specified by an instance vector $\boldsymbol{I}_j = [a_j, p_j, \boldsymbol{R}_j]$, where a_j is the arrival time, and p_j is the duration of processing when vm_j runs alone. The duration time p_j can be extracted from the instance information manager [35]. Then it is assumed that p_j is known in advance [35], [36], which is common in the research of resource scheduling (E.g., scheduling high-performance computing or scientific computing applications [37]). Note that the VMs should start at the arrival time. The capacity vector $\boldsymbol{R}_j = \{R_{jk}\}$ ($k = 1, 2, ..., s$) represents the resources that vm_j requires for processing its work. For example, an instance type of VM in Amazon EC2 [38] specifics its resource capacity {CPU:2 vcpu/8 EC2 units, memory:7 GB, storage:1680 GB}.

For each pair of vm_j and $vm_{j'}$, the system defines the performance degradation factor $d_{jj'} \geq 0$ as the percentage increase in the execution time of vm_j when the VMs collocate on the same physical server. The concept of the performance degradation factor was proposed by researchers in [15], [17], [18], [20], [39]. It is reasonable to use the performance degradation factor caused by the interference among the VMs collocated on a physical server as a factor that indicates the increase of the execution time of a VM, since the collocated VMs will have contention on the resources and need additional management work for their executions on the server. The degradation factor can be determined by using the methods provided in [15], [20], [39]. Briefly, a VM is profiled individually to generate an interference profile, and then the degradation factor is estimated by degradation prediction model. (Detailed description of the methods to determine the performance degradation factor can be found in [15], [20], [39].) The degradation factor obtained in this way indicates the performance interference between the VMs collocated on the same physical server. Thus they can be used for the design of VM assignment and scheduling algorithms to reduce performance degradation. We focus on the VM assignment and scheduling given that these factors are known. Note that $d_{jj'}$ may not be equal to $d_{j'j}$ because two VMs will experience different extents of degradation. It is also showed that adding VMs to the server to concurrently run with existing VMs will increase the degradation factor of previous VMs [18].

A method is given in [40] to estimate the execution time of a VM when it is collocated with other VMs on a server. Given

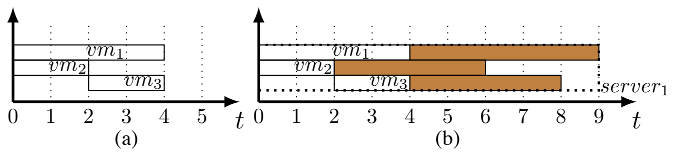

Fig. 2: The execution of VM instance collocation in a server. Assume the server has a resource capacity of $\{\mathbf{1}\}$ unit. Each VM from $\{vm_1, vm_2, vm_3\}$ demands $\{\mathbf{\frac{1}{3}}\}$ unit, and the processing times for the three VMs are given by $p_1 = 4$, $p_2 = 2$, and $p_3 = 2$. The performance degradation factors among them are all $\{d_{..}\} = 1$. (a): VM instance configuration. (b): Running on a server.

a pair of two VMs, vm_j and $vm_{j'}$, collocated on a server and suppose that the performance degradation factor is $d_{jj'} = d$, then if the execution time of vm_j in isolation is T, the time needed to execute the two multiplexed VMs is $T{\cdot}(1+d)$. The execution time of i VMs is $T{\cdot}(1+d)^{i-1}$, and the performance degradation factor of a VM is $\frac{T\cdot(1+d)^{i-1}-T}{T} = (1+d)^{i-1}-1$, where i is the number of VMs that collocated on the same server. It is assumed here that performance degradation factors are different for different VM pairs, which is the same as in [15], [20], [39]. Suppose that vm_j is collocated with a set $\mathcal{J}$ of VMs on a server, and the performance degradation factor of vm_j with the set $\mathcal{J}$ of k VMs is $d_{j\mathcal{J}}$. Then $d_{j\mathcal{J}} = (1+d_{jj_1})(1+d_{jj_2}){\cdots}(1+d_{jj_{k-1}})-1$. Thus we have

$$d_{j\mathcal{J}} = \Pi_{j'\in\mathcal{J}}(1+d_{jj'})-1. \tag{2}$$

Notice that the degradation factor $d_{jj'}$ may be different when the two VMs have different combinations with other VMs, and they are estimated using the methods provided in [15], [20], [39]. Then, the degradation factor is used to estimate the processing time work; i.e., when vm_j concurrently runs with a set $\mathcal{J}$ of VMs for duration time $\widetilde{p}_j$, it completes $\frac{1}{1+d_{j\mathcal{J}}}\widetilde{p}_j$ work of processing time. To illustrate the behavior of this interference model between VMs, consider the example shown in Fig. 2. During the first 2 units of time, vm_1 is collocated with vm_2. Each VM processes 1 unit of work because $\frac{2}{1+d_{12}} = \frac{2}{1+d_{21}} = 1$. In the next 2 units of time, as the vm_3 joins in, all the VMs process 0.5 units of work because $\frac{2}{(1+d_{12})(1+d_{13})} = \frac{2}{(1+d_{21})(1+d_{23})} = \frac{2}{(1+d_{31})(1+d_{32})} = 0.5$. From time 4 to 6, the situation is the same as that between time 2 and 4: The VMs process 0.5 units of work. vm_2 leaves the server at time 6 when it completes its processing work. From time 6 to 8, vm_1 and vm_3 process 1 unit of work. vm_3 leaves at time 8 as it completes its processing work. Finally, vm_1 will process 1 more unit of time to complete its work if it runs alone or if the server is assigned VMs that do not cause performance degradation to vm_1.

C. Scheduling Problem Description

There are two issues concerning the allocation of VMs that need to be addressed. Cloud infrastructure providers offer specific types of VMs, which tend to reserve resources, such as CPU, memory and storage space. The providers' goal is to reduce the operational cost, i.e., minimize energy consumption or capacity cost. In contrast, cloud users seek to reduce the running time of their requests to save on bills for renting the resources.

In our scheduling, time is divided into discrete periods, $t = 1, 2, ..., T$. For example, the interval τ can be one or five minute(s). The binary decision variable $x_{ij}(t)$ indicates whether vm_j is allocated to $server_i$ at the time slot t. The variable defines $c_i(t)$ as the energy consumption cost of $server_i$ running during time slot t, i.e., $c_i(t) = P[u(t)]\tau$. Thus, the total operational cost during time slot t is the sum of all running servers, which is calculated as

$$C_i(t) = \sum_i c_i(t). \tag{3}$$

Let $Q(t)$ denote the set of VMs that run at time slot t. Define $D(t)$ as the set of VMs that complete their execution and leave at time slot t. Hence, the actual execution time t_j of vm_j $(j \in D(t))$ is

$$t_j = t - a_j, \quad t_j{\geq}p_j. \tag{4}$$

The performance degradation penalty is modeled by a convex function $f(\cdot)$. One natural model is $f[(\frac{t_j-p_j}{p_j})] = \alpha^{(\frac{t_j-p_j}{p_j})} - 1$ (α is a parameter. Note that $t_j{\geq}p_j$), which penalizes the delay cost due to exceeding the processing time p_j. Therefore, the VM scheduling problem is defined as the following optimization:

$$min \qquad \sum_{t=1}^{T}\sum_{i=1}^{M}c_i(t) + \beta\sum_{t=1}^{T}\sum_{j\in D(t)}f[(\frac{t_j-p_j}{p_j})] \tag{5}$$

$$s.t. \qquad \sum_{j\in Q(t)}x_{ij}(t)R_{jk}{\leq}R_{ik} \quad \forall i, \forall t, \forall k, \tag{6}$$

$$\sum_{i=1}^{M}x_{ij}(t) = 1 \quad \forall j, \forall t, \tag{7}$$

$$x_{ij}(t){\leq}x_{ij}(t+1) \quad \forall t, j{\notin}D(t), \tag{8}$$

$$x_{ij}(t){\in}\{0,1\} \quad \forall i, \forall j, \forall t. \tag{9}$$

The objective function (5) minimizes the total operational server costs and performance degradation penalties, and $\beta > 0$ is a constant that represents the relative importance between two objectives. Constraint (6) ensures that the aggregated resource demand of multiple VMs does not exceed a server's capacity for all resource types and at all time slots. Constraint (7) ensures that each VM is allocated to one of the servers at any point in time. Constraint (8) ensures that if a VM has been assigned on a server it will not be assigned to other servers. Recall that $D(t)$ is the set of VMs that complete their execution at time slot t. More specifically, if vm_j has not completed the execution at time slot t (i.e., $j{\notin}D(t)$, $x_{ij}(t{+}1) = 1$), we have $x_{ij}(t{+}1){\geq}x_{ij}(t)$ because $x_{ij}(t) = 0$ or $x_{ij}(t) = 1$. Constraint (9) ensure that $x_{ij}(t)$ is set to one if vm_j is allocated to $server_i$ at time slot t. We first present the computational complexity of this problem:

Theorem 1. *The problem of finding an optimal VM schedule for arbitrating between operational cost and performance degradation penalty is NP-complete.*

Proof: First, we transform the optimization problem into an associated decision problem: Given the instance vectors of

VMs, the performance degradation factors, and a bound on the sum of energy consumption and performance degradation penalty, is there a schedule such that the bound on the sum of the cost and penalty is satisfied? Clearly, the problem is in $\mathcal{NP}$ because we can compute and verify in polynomial time that a proposed schedule satisfies the given bound on the sum of the operational cost and performance degradation penalty. We next prove that finding an optimal VM schedule for arbitrating between energy consumption and performance degradation penalty is NP-complete via the reduction from the 3-Dimensional Matching Problem [41], [42].

Consider an instance of 3-Dimensional Matching: Let $A = \{a_1, a_2, .., a_q\}$, $B = \{b_1, b_2, ..., b_q\}$, and $C = \{c_1, c_2, ..., c_q\}$ be three disjoint sets containing q elements each. Let $Z = \{z_1, z_2, ..., z_{l'}, ..., z_l\}$ be a set of triples such that each z_i consists of one element from A, one element from B, and one element from C. Is there a subset $Z^{'} \subseteq Z$ such that every element in A, B, and C appears in exactly one triple in $Z^{'}$? We construct an instance of the VM scheduling problem as follows. Let there be $3q$ VMs and $M_q(\geq q)$ servers. The VMs correspond to the elements in A, B and C. For each $1 \leq i \leq M_q$, $server_i$ has resource vector $\boldsymbol{R_i} = \{R_{ik}\} = \{\mathbf{1}\}$ $(k = 1, 2, ..., s)$. For each $1 \leq j \leq 3q$, vm_j has instance vector $[0, 1, \{\mathbf{\frac{1}{3}}\}]$. VMs have no interference with each other in the triples $z_{l'} \in Z^{'}$; otherwise, they have a performance degradation factor of 1 between each other. The sum of the cost and penalty is q. The energy consumption cost of a server is 1 per unit of time slot when it runs at full utilization (suppose it to be 0.5 at idle). The sum of the cost and penalty is equal to q if and only if the $3q$ VMs are scheduled on q servers and do not cause performance degradation, i.e., $\bigcup\limits_{server_i} z_{l'} = A \cup B \cup C$. Thus, there is an optimal VM schedule if and only if there exists a 3-Dimensional Matching. It is clear that the abovementioned reduction is a pseudo-polynomial reduction. Thus, we can conclude that the problem is NP-complete by this pseudo-polynomial time reduction from the 3-Dimensional Matching Problem, which has been proven to be NP-complete. ■

IV. VM Scheduling Design

Because it is an NP-hard combinatorial optimization problem and there is no computationally efficient solution, we exploit the unique problem structure of VM scheduling in cloud data centers to develop the solutions. We first study the offline problem. Then, we extend the solutions to the dynamic environment of the problem (i.e., online version).

A. Offline Scheduling Design

In this condition, the information of VMs that will be scheduled is known at the outset. The multiple user requests for VMs can be a typical offline scenario. We propose offline algorithms for VM scheduling and analyze the performance.

Bin Packing Variant Algorithm (BPV). From the perspective of single energy consumption criterion optimization, various packing algorithms have become the reserve choices. It is an obvious advantage to reduce energy consumption when decreasing the number of active servers [30]. Therefore, an algorithm derived from First-Fit bin packing is considered. The algorithm keeps the VMs in a list sorted in increasing order of the arrival time. Each VM is allocated to the first possible accommodated server according to the list order and invokes a server when a capacity violation occurs. The difference from the First Fit algorithm is that VMs will depart from the servers when they complete their work and the relevant resources will be recovered. Note that such an approach is widely used in current cloud services, and it also allows an implicit comparison with prior work [20], [30].

Minimum Increasing Cost Algorithm (MIC). Another natural algorithm is the greedy differential of increasing costs of energy consumption and performance degradation penalty. The VMs are also kept in increasing order of their arrival time. The algorithm assigns the next VM to the server that minimizes the increment in total cost. There are two choices for the allocation of the next VM. The increment in total cost is the sum of energy consumption and the performance degradation penalty when the VM is allocated to an active server running existing VMs. The other choice is a currently unused server that involves paying for more static energy consumption, supposing the VM process runs alone.

Theorem 2. *Let I_{max} denote the maximum number of VMs that can be simultaneously accommodated by a server. The approximation ratio of the MIC algorithm is I_{max}.*

Proof: Note that for the minimization problem, an algorithm achieves a δ-approximation factor if, for all instances, it returns a solution at most δ times the optimal value.

We decompose the power of a $server_i$ at time t into the VMs according to the proportion of their CPU resources. For example, at time t, there are n_t VMs with a CPU resource of R_{j1} $(j = 1, ..., n_t)$ (here, we set the resource type 1 as the CPU resource) on $server_i$, which has CPU resource C_{i1}. Then, vm_j consumes $\frac{R_{j1} P_{idle}}{\sum_{j=1}^{n_t} R_{j1}} + \frac{R_{j1}(P_{peak} - P_{idle})}{C_{i1}}$ power, where the first term corresponds to the proportion of the static power and the second term is the dynamic power this VM consumes. Without loss of generality, we consider the VM vm_j. The power of this VM during its execution time is at least $\frac{R_{j1} P_{idle}}{C_{i1}} + \frac{R_{j1}(P_{peak} - P_{idle})}{C_{i1}}$ because $\sum_{j=1}^{n_t} R_{j1} \leq C_{i1}$. Again, we decompose the total cost of energy consumption and the performance degradation penalty to the cost of each VM when it is allocated. According to the MIC algorithm, the cost of inserting vm_j is no more than $[P_{idle} + \frac{R_{j1}(P_{peak} - P_{idle})}{C_{j1}}] * p_j$. Thus, the approximation ratio is

$$\frac{cost(MIC)}{cost(OPT)} \leq \textcircled{1} \frac{\sum\limits_j [P_{idle} + \frac{R_{j1}(P_{peak} - P_{idle})}{C_{j1}}] * p_j}{\sum\limits_j [\frac{R_{j1} P_{idle}}{C_{i1}} + \frac{R_{j1}(P_{peak} - P_{idle})}{C_{i1}}] * p_j}$$

$$= \frac{\sum\limits_j P_{idle} * p_j + \sum\limits_j \frac{R_{j1}(P_{peak} - P_{idle})}{C_{j1}} * p_j}{\sum\limits_j \frac{R_{j1} P_{idle}}{C_{i1}} * p_j + \sum\limits_j \frac{R_{j1}(P_{peak} - P_{idle})}{C_{i1}} * p_j}$$

$$\leq \textcircled{2} \frac{\sum\limits_j P_{idle} * p_j}{\sum\limits_j \frac{R_{j1} P_{idle}}{C_{i1}} * p_j} \leq \textcircled{3} \frac{\sum\limits_j P_{idle} * p_j}{\sum\limits_j \frac{P_{idle}}{I_{max}} * p_j} = I_{max},$$

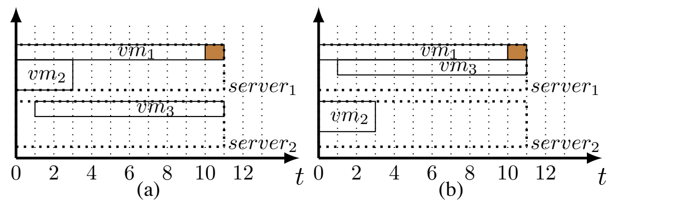

Fig. 3: BPV+MIC and an improved schedule for three VMs. Assume both physical servers have a resource capacity of $\{\mathbf{1}\}$ unit and vm_1, vm_2 and vm_3 have resource capacities of $\{\frac{1}{3}\}$, $\{\frac{2}{3}\}$, and $\{\frac{1}{3}\}$ unit, respectively. The performance degradation factors among the VMs are $\{d_{12} = d_{13} = 0,\ d_{21} = d_{31} = 0.1,\ d_{23} = d_{32} = 0.2\}$, whereas the processing times for the three VMs are given by $p_1 = 10$, $p_2 = 3$, and $p_3 = 9$. (a): BPV or MIC scheduling. (b): A better scheduling.

where the second inequality follows from $\sum_j P_{idle} * p_j \geq \sum_j \frac{R_{j1} P_{idle}}{C_{i1}} * p_j$ as $R_{j1} \leq C_{i1}$ and from the mathematical inequality $\frac{a+c}{b+c} \leq \frac{a}{b}$ as $a \geq b, c \geq 0$. The third inequality results from $I_{max} * R_{j1} \geq C_{i1}$, i.e., $\frac{R_{j1} P_{idle}}{C_{i1}} \geq \frac{P_{idle}}{I_{max}}$. This concludes the theorem. ■

Remark: The BPV algorithm only considers accepting the next VM and does not take into account performance degradation. Both of the abovementioned algorithms sort the VMs by their arrival time and then depend only on the information that is available to the algorithms at the scheduling time. Thus, they can be modified as online algorithms when excluding the sorting phase. Note that when a VM is allocated to a server, the VM's own duration time and the duration times of other VMs that are interfered by it must be updated. It is observed that these algorithms do not consider the life cycle overlapping of VMs. For example, there are three VMs to be scheduled, which are configured as shown Fig. 3. In Fig. 3(a), both of the abovementioned algorithms cause two servers to be active from time 0 to 11 and from time 1 to 11. A better scheduling (Fig. 3(b)) involves assigning vm_1 and vm_3 to $server_1$ and vm_2 to $server_2$. Thus, we can put $server_2$ in power-saving mode or turn-off mode from time 3 to 11.

Maximum Decreasing Cost Algorithm (MDC). Instead of sorting the VMs by the arrival time, the MDC algorithm considers the information of all VMs and operates like a clustering algorithm. We pursue the minimum cost of energy consumption and the minimum performance degradation penalty iteratively. Initially, each VM is allocated to a dedicated server. Next, we decide to repeatedly merge servers together by the formation of pairs. The goal of merging (MergeServers(·,·)) is to collocate the VMs on one server. The duration time of the VMs that cause interference among them must also be updated. We define the gain function of merging two servers, $\mathcal{S}_u$ and $\mathcal{S}_v$, as follows:

$$Gain(\mathcal{S}_u, \mathcal{S}_v) = Cost(\mathcal{S}_u) + Cost(\mathcal{S}_v) - Cost(\mathcal{S}_u \cup \mathcal{S}_v), \tag{10}$$

where $Cost(\cdot)$ denotes the total cost of the server according to the cost model defined in Section III-C. With respect to the merger that causes the violation of server capacity, we define the gain as a negative number. At each step, we choose the merger of two servers that results in the maximum decrease in the total cost. The algorithm ends when the merger of any two servers produces a negative gain. The pseudo-code for the MDC algorithm is summarized in ***Algorithm. 1***.

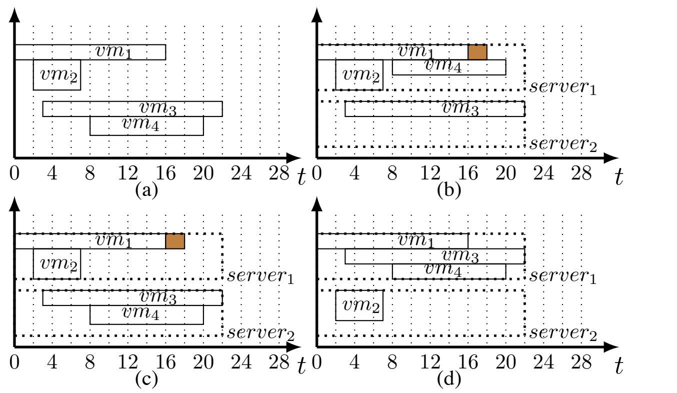

Fig. 4: Scheduling of four VMs. Assume each physical server has a resource capacity of $\{\mathbf{1}\}$ unit. Each VM from $\{vm_1, vm_3, vm_4\}$ demands $\{\frac{1}{3}\}$ unit, whereas vm_2 requires $\{\frac{2}{3}\}$ unit. The processing times for the VMs are given by $p_1 = 16$, $p_2 = 5$, $p_3 = 19$, and $p_4 = 12$. The performance degradation factor d_{21} is 0.25, and other factors among the VMs are 0. (a): VM instance. (b): BPV scheduling. (c): MIC scheduling. (d): MDC scheduling.

Algorithm 1: Maximum Decreasing Cost Algorithm

```
input  : the set of VMs $\mathcal{N}$
output: the scheduling result of VMs
begin
    Initial Servers $\mathcal{S}$={$\mathcal{S}_1 = \{vm_1\}, ..., \mathcal{S}_n = \{vm_n\}$};
    $maxGain = \max_{u,v} Gain(\mathcal{S}_u, \mathcal{S}_v)$;
    while $maxGain \geq 0$ do
        MergeServers($\mathcal{S}_u, \mathcal{S}_v$);
        Set $\mathcal{S}_u = \mathcal{S}_u \cup \mathcal{S}_v$, $\mathcal{S} = \mathcal{S} \backslash \mathcal{S}_v$;
        $maxGain = \max_{u,v} Gain(\mathcal{S}_u, \mathcal{S}_v)$;
    end
    Return the set of servers $\mathcal{S}$ and their accommodated VMs.
end
```

To illustrate the different behavior of these three scheduling strategies, we present an example in Fig. 4. BPV aggregates the VMs in parts of servers and leads to some servers as low-utilization servers. MIC is more likely to assign the subsequent VMs to be included by the anterior ones during its execution when the performance degradation factor is low. As a result, the algorithm seeks to balance the VMs between the servers. MDC prefers to collocate the VMs that share long life cycles and have low performance degradation factors between them. In summary, these scheduling algorithms exhibit different performances, and we will evaluate them in Section V.

B. Online Algorithm for Dynamic Problem

In this section, we introduce the online version of the VM scheduling, which is an extension of the previous algorithms. In this case, a sequence $\mathcal{N}$ of VMs arrives over time, where $\mathcal{N} = \{vm_0, vm_1, ..., vm_j, ...\}$. Each VM vm_j must be assigned to a server upon its arrival, without information about the future set of VMs ($\{vm_{j'} | j' > j\}$). We provide the algorithm that schedules each incoming VM by dispatching the VMs to current active servers or a new server that will be activated.

Recall that the BPV and MIC algorithms first sort the VMs in a list in increasing order of the VMs' arrival time, and then schedule VMs one by one according to the sequential order. So they perform the scheduling upon the arrival of a VM vm_j at time t without the knowledge of future arrival VMs. That is, the algorithms depend only on the information that is available to the algorithms at the moment. Thus, the algorithms can also support the scheduling in the online version without changing their procedure, and they are denoted by OBPV and OMIC, respectively. When a new VM arrives, the OBPV algorithm will allocate the VM to the first possible accommodated server. In the OMIC algorithm, the VM is allocated to the server that minimizes the increment in the total cost. Note that the algorithms operate similarly to BPV and MIC, respectively, in allocating the next VM in the sorted list. We then derive a competitive ratio for the OMIC algorithm. We say that for the minimization problem, an algorithm is γ-*competitive* if for all the problem instances, it returns the cost at most γ times the cost of the optimal offline solution. From the Theorem 2, we have the following competitive ratio:

Theorem 3. *The competitive ratio of the OMIC algorithm for VM scheduling, which aims to arbitrate between operational cost and performance degradation, is at most I_{max}, where I_{max} has the same meaning as in Theorem 2.*

C. Incorporating VM Batch Arrival and VM Reservation

In the previous section, VMs were considered to arrive one by one and there was no information about future arriving VMs. To match the cloud data centers, we incorporate the following two properties into the scheduling design:

- There is a set $\mathcal{N}_t$ of VMs to be scheduled at time t because many users submit their VMs to the cloud data center at the same time.
- There is a set $\mathcal{N}_t^f$ of reserved VMs at time t because users reserve for lower costs and reserving capacity in the cloud data center.

The algorithm defines the time t as the scheduling time only when there are some VMs that must be started at this time. According to the definition of the reserved VMs, we have the arrival time relationship $a_j > a_j^{'}, \forall vm_j \in \mathcal{N}_t^f, \forall vm_j^{'} \in \mathcal{N}_t$. In this situation, the problem is transformed into the scheduling of a set of VMs, i.e., $\mathcal{N}_t \cup \mathcal{N}_t^f$, to be allocated in the cloud data center. The difference from the offline scheduling is that at the beginning of the scheduling, there are some VMs that have been allocated in the cloud data center. Consider, for example, scheduling time $t = 2$ in previous Fig. 4; incoming VM vm_2 and reserved VMs $\{vm_3, vm_4\}$ must be scheduled at this time. We prefer the scheduling of Fig. 4(d) to that of Fig. 4(c) because it is known that a server can be put into power-saving mode or shut down only if there are no VMs active on it. Thus, the problem concerning which VM must be scheduled first if the OMIC algorithm is used to schedule the VMs one by one arises. The algorithm defines the alignment ratio of the VMs to the server as follows: the ratio of the VMs' completion time to the server's completion time. An obvious intuition is to allocate the VMs to maximize the alignment ratio if these VMs have weak performance interference, i.e., to align the VMs and their server. Thus, the objective is to allocate the VMs to their best candidate server. Based on the foregoing analysis, we present an algorithm from the servers' perspective. First, the algorithm supposes that each VM $vm_j \in (\mathcal{N}_t \cup \mathcal{N}_t^f)$ is allocated to a dedicate virtual server. Each server proposes the profits to other VMs that are allocated to virtual servers. To be specific, the profit metric is defined as follows:

$$Profit_{i,j} = Cost(vm_j) - AddCost(server_i, vm_j), \quad (11)$$

where $Cost(vm_j)$ is the total cost of a server to run vm_j alone, and $AddCost(server_i, vm_j)$ is the increment total cost of running vm_j on $server_i$. When vm_j cannot be allocated to the server $server_i$, the $Profit_{i,j}$ is simply set to a negative value. The algorithm allocates the VMs iteratively. In each round, the algorithm selects the maximum profit, i.e., $\{\max\limits_{i,j} Profit_{i,j} | Profit_{i,j} >= 0\}$. Then, the vm_j is allocated to $server_i$. The algorithm stops when $\max\limits_{i,j} Profit_{i,j} < 0$. In this situation, the algorithm indicates that the total cost cannot be improved. The VMs are allocated to their current servers, and virtual servers will be active to process the VMs. The process is summarized in ***Algorithm. 2*** (IVP).

Algorithm 2: Incorporating VM Plan Online Algorithm

```
input  : the set of VMs $\{\mathcal{N}_t \cup \mathcal{N}_t^f\}$, current active servers
         $\mathcal{S}_I$ at time $t$
output: the scheduling result of VMs
begin
    Set virtual servers
    $\mathcal{S}_V$=$\{\mathcal{S}_j$=$\{vm_j\} | vm_j \in \{\mathcal{N}_t \cup \mathcal{N}_t^f\}\}$;
    Set all servers $\mathcal{S} = \mathcal{S}_V \cup \mathcal{S}_I$;
    Set $maxProfit = \max\limits_{i,j} Profit_{i,j}$, $i \in \mathcal{S}$;
    while $maxProfit \geq 0$ do
        The configuration
        $Server_i(t) = \arg\max\limits_{i \in \mathcal{S}} Profit_{i,j}$;
        Update $server_i$ and Delete $vm_j$ from $\{\mathcal{N}_t \cup \mathcal{N}_t^f\}$;
        Set $maxProfit = \max\limits_{i,j} Profit_{i,j}$;
    end
    Return the plan of VMs.
end
```

We use an example to explain this planning algorithm, as illustrated in Fig. 5. Assume that at time $t = 2$, there is one active server $server_1$ with vm_1 running on it, and there is one VM vm_2 arriving and two reserved VMs $\{vm_3, vm_4\}$. Because $t = 2$ is the scheduling time, the planning algorithm is triggered. First, each VM $vm_j \in \{vm_2, vm_3, vm_4\}$ is allocated to a virtual server $server_i \in \{server_2, server_3, server_4\}$, respectively. Then, the profit $Profit_{i,j}$ is calculated according to Equation. 11. Because $Profit_{1,3}$ is the maximum profit, the algorithm allocates the vm_3 to $server_1$ in the first round. This procedure is repeated in the second round, and vm_4 is allocated to $server_1$. In the last round, vm_2 is allocated to $server_2$. Finally, the algorithm generates the allocation shown

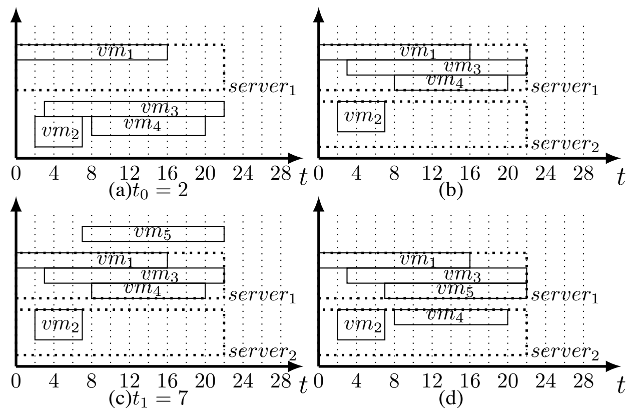

Fig. 5: Incorporating VMs' batch arrival and VMs' reservation scheduling. Assume each physical server has a resource capacity of $\{\mathbf{1}\}$ unit. vm_1, vm_2, vm_3, vm_4, and vm_5 have resource capacities of $\{\frac{1}{3}\}$, $\{\frac{2}{3}\}$, $\{\frac{1}{3}\}$, $\{\frac{1}{3}\}$, and $\{\frac{1}{3}\}$ unit, respectively. The processing times for the VMs are given by $p_1 = 16$, $p_2 = 5$, $p_3 = 19$, $p_4 = 12$, and $p_5 = 15$. The performance degradation factor d_{21} is 0.25, and other factors among the VMs are 0. (a): $t = 2$ Initial servers and VMs. (b): $t = 2$ Allocation result. (c): $t = 7$ Initial servers and VMs. (d): $t = 7$ Allocation result.

in Fig. 5(b). Assume that at time $t = 7$ there is one VM vm_5 arriving. Thus, the planning algorithm is triggered and generates the allocation shown in Fig. 5(d). Note that the allocation of the reserved VM vm_4 is changed from $server_1$ to $server_2$.

The time complexity of this algorithm is $O(mn^2)$, where m is the number of servers ($|\mathcal{S}_V \cup \mathcal{S}_I|$) and n is the number of VMs ($|\mathcal{N}_t \cup \mathcal{N}_t^f|$). It follows that each server $server_i$ proposes a profit to VM vm_j, and in each round, we fix a VM vm_j.

D. Distributed Design toward Data Center Scale

In a large data center, it is time-consuming to gather detailed information about each server and run the algorithm on a single server. We thus propose a distribution scheme that opposes the algorithm introduced in the previous section and that centralizes information and selects the best candidate server. Upon the arrival of each new VM to the data center, information regarding the VMs, which are waiting to be allocated, is passed to each active server. Each server $server_i$ maintains the information of the VMs. Next, the algorithm proceeds in stages and synchronizes using a common clock. In the first stage, each single client server $server_i$ proposes profits to the VMs and sends the maximum profit $maxProfit_{i,j}$ of vm_j to the distribution server (dispatcher). The distribution server collects the maximum profits from all client servers and chooses the $max\text{-}maxProfit$, i.e., the current maximum benefit from the allocation of $vm_{\widehat{j}}$ to $server_{\widehat{i}}$. Then, the allocation decision of stage o is broadcast to client servers. In the subsequent stages, the client server $server_i$ receives the decision message $\langle\widehat{i},\widehat{j}\rangle$ from the distribution server and proceeds to make the following two choices: 1) If the profit of the $(o-1)$ stage is chosen, the algorithm fixes $vm_{\widehat{j}}$ on it and removes $vm_{\widehat{j}}$ from the unscheduled VMs. 2) If the profit of the $(o-1)$ stage is not chosen, the algorithm simply removes $vm_{\widehat{j}}$ from the unscheduled VMs. The profit-proposing procedure is the same as the previous stage. Assume there are some VMs that are unscheduled, i.e., they do not benefit from being allocated to the client servers or cannot be allocated to current active servers. We run *Algorithm. 2* to schedule them with the input of these VMs and some current inactive servers. The pseudo-code of this algorithm is summarized in ***Algorithm. 3***.

Algorithm 3: VM Profit Planning Algorithm

```
input : the set of VMs ${\mathcal{N}_t \cup \mathcal{N}_t^f}$ at time $t$
output: the allocation results of VMs
$DistributionServer$ :
begin
    Initialized round: Broadcast $\mathcal{J} = \langle\{\mathcal{N}_t \cup \mathcal{N}_t^f\}\rangle$;
    foreach round $o = 1, 2, ...$ do
        Receive message $\langle maxProfit_{i,j}\rangle$;
        if $\max_{i,j} maxProfit_{i,j} \geq 0$ then
            Pick $\langle i,j\rangle = \arg\max_{i,j} maxProfit_{i,j}$;
            Broadcast $\langle i,j\rangle$;
            Update $\mathcal{J} \backslash j$;
        end
    end
    if $\mathcal{J} \neq \emptyset$ then
        Run Algorithm. 2, i.e., $IVP(\mathcal{J}, \emptyset)$;
    end
    Return the allocation of VMs.
end
$ClientServers$ :
begin
    foreach server $server_i$ in parallel do
        Receive message $\mathcal{O}$ from $DistributionServer$;
        if $\mathcal{O} = \langle\{\mathcal{N}_t \cup \mathcal{N}_t^f\}\rangle$ then
            Save $\mathcal{J} = \mathcal{O}$;
            Set $\mathcal{S}_V = \{\mathcal{S}_j = \{vm_j\} | vm_j \in \{\mathcal{N}_t \cup \mathcal{N}_t^f\}\}$;
            Set $\mathcal{S} = \mathcal{S}_V \cup \{server_i\}$;
            Set $maxProfit_{i,j} = \max_{i,j} Profit_{i,j}$;
            Send $\langle maxProfit_{i,j}\rangle$ to the
            $DistributionServer$;
        else if $\mathcal{O} = \langle\widehat{i},\widehat{j}\rangle$ then
            if $\widehat{i} == server_i$ then
                Fix $\widehat{j}$ on $server_i$ and Update $server_i$;
            end
            Set $\mathcal{S}_V = \{\mathcal{S}_j = \{vm_j\} | vm_j \in \{\mathcal{J} \backslash \widehat{j}\}\}$;
            Set $\mathcal{S} = \mathcal{S}_V \cup \{server_i\}$;
            Set $maxProfit_{i,j} = \max_{i,j} Profit_{i,j}$;
            Send $\langle maxProfit_{i,j}\rangle$ to the
            $DistributionServer$;
    end
end
```

V. Performance Evaluation

In this section, we study the performance of the proposed algorithms through a small-scale experiment and a large-scale simulation.

A. Evaluation Setup

Setup Settings: The experiment and simulation are run in a data center that equips physical servers with a computing resource of 12 cores (e.g., *HP ProLiant DL385 G6*). For simplicity, we assume that the servers are homogeneous in the data center and that the VMs take up the total resources of their requested demand. The configuration of the VMs refers to the types of instances available in Amazon EC2 [38]. For example, a so-called *m1.small* VM features 1 core computing unit, 1.7 GB of memory and 160 GB of storage space. Because we focus on studying the arbitration between energy consumption and performance interference degradation, we omit other resource bounds, such as memory, in the simulations. Four different types of VMs are available to be chosen with computing resources of 1 core, 2 cores, 4 cores, and 8 cores. The simulations are conducted in different scenarios corresponding to offline and online VM scheduling. In the offline problem, a list of VMs have already been waiting to be processed. In the online scenario, the VMs arrive randomly over time.

Compared Baseline Algorithms: To provide benchmarks for our evaluations, we introduce three other algorithms:

- Random Strategy (RAND): a naive algorithm that randomly schedules the next VM on a physical server as long as the server has enough resources to host the VM.
- Round Robin (RR): an algorithm that allocates the next VM to physical servers in turn; the algorithm is used as a scheduling algorithm in Amazon EC2 [43].
- Minimum Increase Energy (MIE): an algorithm that assigns the next VM to the server that has the minimal increment in energy consumption. Note that this algorithm is different from MIC because the latter considers the increment in total cost.

All these algorithms first sort the VMs in a list in increasing order of the VMs' arrival time and a new server will open if the next VM cannot be allocated to current active servers.

Parameters: The power of a server is characterized by the three parameters indicated in Equation. 1. We use an HP ProLiant DL385 G6 with a 2 chip/12 core processor. According to the server power consumption parameters, we set $P_{idle} = 120W$ and $P_{busy} = 258W$. The performance degradation cost is characterized by the following parameters: α and β. The parameter α represents the intensity of the performance degradation penalty, and without loss of generality, it is set to 15. The tuning parameter β is used to adjust the energy consumption cost and performance degradation penalty and to represent the weight between the two costs.

Performance Metrics: To evaluate the performance of the proposed algorithms, we use the following four metrics:

- Normalized Total Energy Consumption: This metric demonstrates the quality of the solution produced by the proposed algorithms in terms of total energy consumption.
- Normalized Total Performance Degradation Penalty: This metric represents the penalties of the solution produced by the proposed algorithms in terms of performance degradation cost.
- Normalized Total Cost: This metric is defined as the sum of the energy consumption cost and performance degradation penalty.
- Normalized Worst Degradation Factor: This metric indicates the worst performance degradation factor of VMs caused by the scheduling algorithms.

All these performance metrics are normalized against the results of the BPV algorithm. In the offline problem, we also record the total number of physical servers used and the makespan of the VMs.

B. Experiment Results and Verification

We first conduct a small-scale experiment to evaluate the performance of the proposed algorithms. The performance degradation ratio is obtained from the statistics of SPECcpu 2006 benchmarks [19]. The properties (such as the arrival time and duration time) of the applications are drawn from a real OpenCloud Hadoop cluster trace [44]. Thus, this evaluation involves a small number of VMs, in which the workload trace and performance degradation ratio are obtained from the real case. The results are shown in Fig. 6. The figure indicates that the overall costs are reduced. More precisely, the MIC and MDC algorithms individually save approximately 37% and 41% on the total cost compared with the BPV algorithm. This result demonstrates the competitive advantage offered by the proposed algorithms versus methods that do not provide a unified consideration of both energy consumption and performance interference. The result also confirms that our solutions are effective and provide better performance in real-world situations. Next, we will use the simulation to evaluate the performance of our solutions on a much larger scale and for a wide range of workloads.

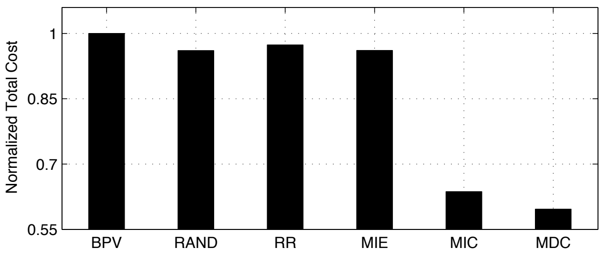

Fig. 6: Performance of algorithms in a real case. (The results are normalized against those obtained by the BPV algorithm.)

C. Large-Scale Simulation Results

In this section, we present the large-scale simulation results for the VM scheduling problem. The goal is to illustrate the impact of a wide variety of parameters and provide a better understanding of the joint optimization of operational cost and performance interference.

1) Evaluation of Offline Problem Solution: In the chosen configuration, the computing resource of a VM is randomly and uniformly chosen from the four types mentioned in the setup settings. We set each time slot to one minute. The duration time of the VMs is randomly generated from

$[30, 1000]$, and the arrival time of the VMs is randomly generated from $[0, 1000]$. The number of VMs varies from 100 to 1000 to correspondingly emulate the low workload and heavy workload in the data center. The algorithms consider two types of degradation factors between VMs. One is generated from the normal distribution, and the other is generated from the exponential distribution. We use the two different distributions for degradation factors to demonstrate that our approach is consistently effective. For each simulation, we run the proposed algorithms and compare the algorithms using the same list of VMs; each result of the randomized algorithm represents the average of 10 runs.

Resource Violation: In the first simulation of this evaluation, we examine the resource violation of algorithms when they do not take into account the performance degradation and do not update the duration time of the VMs. In Fig. 7, we show the resource violation of a server scheduled by the BPV algorithm. As shown, the computing resource surpasses the capacity during certain periods (e.g., $t = 653$ to $t = 1469$) due to the extension of the VM execution time.

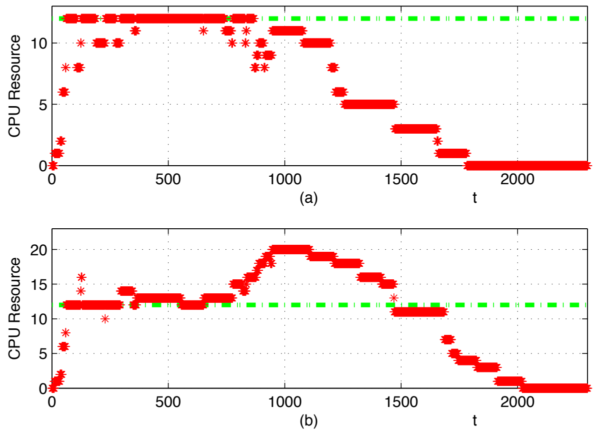

Fig. 7: Resource violation of a server in BPV scheduling. The green dashed line indicates the CPU capacity of a server, i.e., 12 cores. (a) CPU resource used by a server, omitting performance degradation. (b) The true CPU resource used when the algorithm considers that the performance extends the execution time of VMs.

Algorithm Performance Comparison: We now discuss the performance of all the aforementioned algorithms against that of the BPV algorithm with respect to four metrics, i.e., normalized energy consumption, performance degradation penalty, total cost and worst performance degradation factor. In this simulation, the degradation factor between VMs is generated from the normal distribution $N(0.0, 0.2)$ ($d_{..} = 0$ if $d_{..} < 0$). The results are depicted in Fig. 8. As demonstrated, the performance of the MDC algorithm is apparently better than that of other algorithms in reducing the performance degradation penalty. More precisely, the performance degradation penalty of MDC is 6% against BPV, whereas that of MIC is 24%. Fig. 8 also shows that MIC and MDC perform much better with respect to the worst performance degradation factor, which is important in data centers due to the SLA requirement. When considering energy consumption, we find that MIC and MDC also show a slight reduction compared with the other algorithms. This reduction is observed because MIC and MDC reduce the unnecessary execution time due to the performance degradation caused by interference. As a result, MIC and MDC reduce the total cost of energy consumption and the performance degradation penalty by up to 62% and 52%, respectively. It should be noted, however, that the total cost of MDC is reduced by 16% relative to that of MIC.

Impact of Performance Degradation Factor: We next investigate the impact of the performance degradation factor on the proposed algorithms. We keep the number of VMs fixed and execute the above algorithms with five different types of degradation factors between VMs. The factors are generated from the normal distributions $N(0.0, 0.2)$, $N(0.0, 0.4)$, $N(0.0, 0.6)$, $N(0.0, 0.8)$, and $N(0.0, 1.0)$ ($d_{..} = 0$ if $d_{..} < 0$). This setting suggests that the interference between VMs undergoes greater fluctuation as the variance is increased from 0.2 to 1.0. In each normal distribution, we generate five groups of degradation factors, and we generate three groups of VMs. The result is the average of the cross simulations, i.e., the average of 15 runs. Table III shows how the degradation factor affects the total cost. When the performance interference between VMs becomes more intensive, the total costs of the BPV, RAND, RR and MIE algorithms become much greater because they do not take into account the performance degradation penalty when making scheduling decisions. Furthermore, it can be observed that the MIC and MDC algorithms lead to a slight increase in total cost due to their intelligent scheduling. This rule also holds when we generate degradation factors between VMs from exponential distributions $E(100)$, $E(50)$, $E(20)$, $E(10)$, $E(5)$, and $E(2)$. (The results are listed in Table IV.)

TABLE III: Algorithm performance for different $N(0.0, V)$ degradation factors, with the results normalized against $V = 0.2$.

Algorithm	0.2	0.4	0.6	0.8	1.0
BPV	1	186.83	>200	>200	>200
RAND	1	1583.17	>2000	>2000	>2000
RR	1	10.60	23.87	>200	>200
MIE	1	2.78	8.87	14.48	188.06
MIC	1	1.0307	1.0333	1.0381	1.0406
MDC	1	1.0304	1.0480	1.0551	1.0567

TABLE IV: Algorithm performance for different $E(\lambda)$ degradation factors, with the results normalized against $\lambda = 100$.

Algo.	100	50	20	10	5	2
BPV	1	1.0549	1.2728	2.1169	>200	>200
RAND	1	1.0447	1.2130	1.7677	230	>300
RR	1	1.0415	1.1938	1.6070	13.5063	>200
MIE	1	1.0437	1.2089	1.6300	3.6698	263.4512
MIC	1	1.0471	1.1683	1.3006	1.4120	1.5360
MDC	1	1.0383	1.1179	1.2003	1.3072	1.4544

Impact of Workload Density: We compare the proposed algorithms on five data sets, 100, 200, 500, 800 and 1000 VMs, and in each of them, the arrival time of VMs is generated

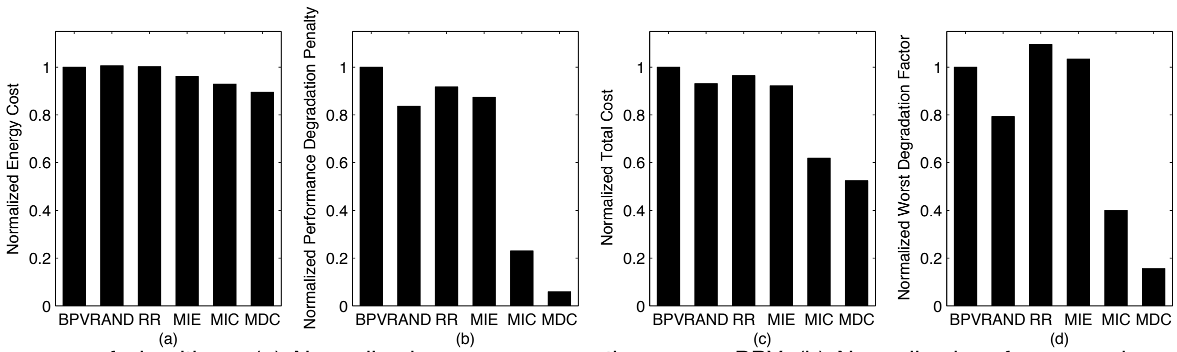

Fig. 8: Performance of algorithms. (a) Normalized energy consumption versus BPV. (b) Normalized performance degradation penalty versus BPV. (c) Normalized total cost versus BPV. (d) Normalized worst performance degradation factor versus BPV.

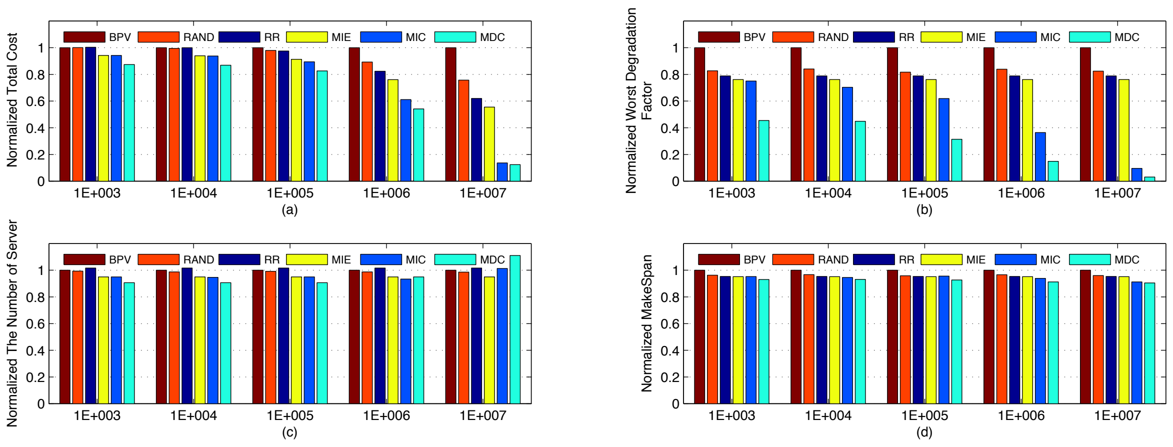

Fig. 10: Impact of different weights β between the operational cost and performance degradation penalty.

from the same range $[0, 1000]$, i.e., the number of data sets from 100 to 1000 represents the increment in workload density. The degradation factor between VMs is also generated from the normal distribution $N(0.0, 0.2)$ ($d_{..} = 0$ if $d_{..} < 0$). Fig. 9 presents the results. In all simulations with different intensities, the minimum total cost is achieved by MDC due to its more global view. Moreover, the improvement margin is stable with the increment in workload density. MIC and MDC perform better under a more intensive load, which is attributed to the fact that the four other algorithms would lead to greater performance degradation when they do not take into account the performance interference under heavy loads.

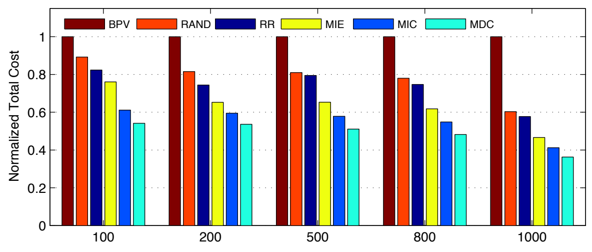

Fig. 9: Impact of different workload densities.

Impact of Weight of Performance Degradation Penalty: As mentioned previously with respect to the optimization model presented in Section III-C, the weight β is a constant that incorporates the normalization and the relative importance of the performance degradation penalty. Our VM scheduling exploits this weight, and now, we study the impact of this weight on the performance of the proposed algorithms. In this simulation, we focus on four metrics: total cost, worst performance degradation factor, the number of servers to be used and makespan. The results are depicted in Fig. 10. Fig. 10 shows that as the weight increases, the results of the worst performance degradation factor, the number of servers to be used and makespan mainly remain the same in BPV, RAND, RR and MIE. This observation is made because the scheduling of these four algorithms is not affected by the weight. The total cost increases only because the weight β increases. In addition, the worst performance degradation factor reduces quickly in MIC and MDC when the weight surpasses a certain value (in our simulation, e.g., $\beta = 10^6$). However, the number of servers to be used increases slightly when we give more weight to the performance degradation penalty, which leads to a decrease in the makespan metric. Clearly, the improvement in total cost in MIC and MDC increases with an increment in the weight.

In summary, the MIC algorithm provides the better performance with respect to joint optimization. Moreover, the MDC algorithm, which considers all input data, can further improve the performance.

2) Evaluation of Online Situation: We recall that the aforementioned evaluation algorithms, except for MDC, support scheduling in both the offline and online versions because they process the input VMs fed to them in sequential order (See

IV.B for more details). Thus, the corresponding evaluation results previously discussed also hold for the online situation, i.e., the online version algorithms (OMIC, OBPV) yield the same evaluation results. Therefore, we can also conclude that the online version algorithm that considers joint optimization (OMIC) provides better performance. Thus, we do not repeat the evaluations.

Here, we focus on evaluating the performance of *Algorithm. 2*, which incorporates the VM batch arrival and VM reservation in the cloud data center. In this simulation, we fix the total number of VMs and their performance degradation factors. We adopt different numbers of VMs to be revealed at each scheduling time to represent the VMs with batch arrival and reservation, i.e., there are n_r VMs revealed at each scheduling time if the number of VMs with batch arrival and reservation is n_r. We set the number $n_r = 1, 2, 3, 4, 5, 10, 20, 50,$ and 100. For example, the situation corresponds to one by one scheduling when the number is equal to 1 (the algorithm operates like OMIC), and there are 10 VMs revealed at each scheduling time when the number is 10. Note that the scheduling time indicates the trigger of *Algorithm. 2*. In other words, the *Algorithm. 2* is triggered to run when a certain number of *unscheduled* VMs arrive. For each n_r, We run the algorithm on five randomly generated instances (1#—5#) and one sequentially generated instance (6#) for each number n_r. More specifically, 6# corresponds to the case in which there are n_r VMs to be revealed in order of arrival time for each scheduling time. The results are depicted in Fig. 11. As shown, the performance improves as the number n_r increases. It can be concluded that the revelation of more information about VMs due to batch arrival and reservation can help improve the total cost. In addition, the total cost is much improved when the number n_r is set from value 1 to 2.

In summary, the *Algorithm. 2* exploits the properties of VM scheduling in cloud data centers and yields a greater improvement in the total cost than the OMIC algorithm, which does not consider these properties.

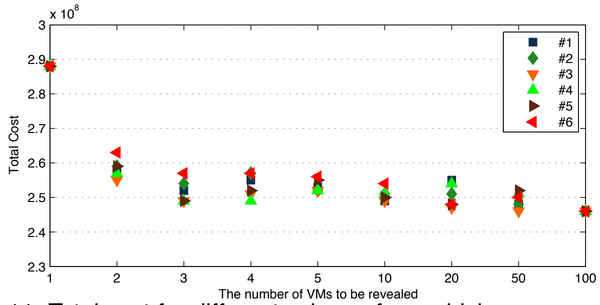

Fig. 11: Total cost for different values of n_r, which corresponds to different numbers of VMs to be revealed at each scheduling time.

VI. Conclusion

In this paper, we present VM assignment and scheduling algorithms to reduce the operational cost and improve the performance interference in cloud data centers. Whereas previous studies only address energy consumption management or performance interference optimization separately, we are among the first to build a joint model to capture the tradeoff between the two contradictory objectives. We also develop efficient scheduling algorithms for both offline and online cases and improve them by exploiting some properties of cloud environments, such as batch arrival and resource reservation. We evaluate the performance of the proposed algorithms by conducting a comprehensive set of simulations. Our results confirm that a joint optimization that takes into account both VM combination and life-cycle overlapping can significantly reduce the total cost as well as the performance interference in cloud data centers.

The performance of the system is an important factor for the quality of service of a cloud data center. How to efficiently utilize the resources while provide satisfiable QoS is a topic worth of further research.

References

[1] L. Wei, H. Zhu, Z. Cao, X. Dong, W. Jia, Y. Chen and A. V. Vasilakos. Security and privacy for storage and computation in cloud computing. *Information Sciences, 258*, pages 371-386, 2014.

[2] L. Wei, H. Zhu, Z. Cao, W. Jia and A. V. Vasilakos. SecCloud: Bridging secure storage and computation in cloud. In *Proceedings of the 30th International Conference on Distributed Computing Systems Workshops (ICDCSW'10)*, pages 52-61, 2010.

[3] M. R. Rahimi, N. Venkatasubramanian and A. V. Vasilakos. MuSIC: Mobility-aware optimal service allocation in mobile cloud computing. In *Proceedings of the 6th International Conference on Cloud Computing (CLOUD'13)*, pages 75-82, 2013.

[4] M. R. Rahimi, N. Venkatasubramanian, S. Mehrotra and A. V. Vasilakos. MAPCloud: Mobile applications on an elastic and scalable 2-tier cloud architecture. In *Proceedings of the 5th IEEE International Conference on Utility and Cloud Computing (UCC'12)*, pages 83-90, 2012.

[5] G. Wei, A. V. Vasilakos, Y. Zheng and N. Xiong. A game-theoretic method of fair resource allocation for cloud computing services. *Journal of Supercomputing, 54(2)*, pages 252-269, 2010.

[6] M. Aazam and E. Huh. Cloud broker service-oriented resource management model. *Transactions on Emerging Telecommunications Technologies*, 2015.

[7] VMware. http://www.vmware.com.

[8] Xen. http://xen.org.

[9] R. Buyya, C. Yeo, S. Venugopal, J. Broberg and I. Brandic. Cloud computing and emerging IT platforms: Vision, hype, and reality for delivering computing as the 5th utility. *Journal of Future Generation Computer Systems, 25(6)*, pages 599-616, 2009.

[10] D. Perera. The real obstacle to federal cloud computing. FiereceGovernmentIT. 2012.

[11] H. Zhang, B. Li, H. Jiang, F. Liu, A. V Vasilakos and J. Liu. A framework for truthful online auctions in cloud computing with heterogeneous user demands. In *Proceedings of the 32nd Annual IEEE International Conference on Computer Communications (INFOCOM'13)*, pages 1510-1518, 2013.

[12] L. Wang, F. Zhang, A. V. Vasilakos, C. Hou and Z. Liu. Joint virtual machine assignment and traffic engineering for green data center networks. *ACM SIGMETRICS performance evaluation review, 41(3)*, pages 107-112, 2013.

[13] D. Niyato, A. V. Vasilakos and Z. Kun. Resource and revenue sharing with coalition formation of cloud providers: game theoretic approach. In *Proceedings of the 11th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing, (CCGrid'11)*, pages 215-224, 2011.

[14] L. Wang, F. Zhang, J. A. Aroca, A. V. Vasilakos, K. Zheng, C. hou, D. Li and Z. Liu. GreenDCN: A general framework for achieving energy efficiency in data center networks. *IEEE Journal on Selected Areas in Communications, 32(1)*, pages 4-15, 2014.

[15] S. Govindan, J. Liu, A. Kansal and A. Sivasubramaniam. Cuanta: Quantifying effects of shared on-chip resource interference for consolidated virtual machines. In *Proceedings of the 2nd ACM Symposium on Cloud Computing (SOCC'11)*, No.22, 2011.

[16] R. C. Chiang, and H. H. Huang. TRACON: Interference-aware scheduling of data-intensive applications in virtualized environments. In *Proceedings of the 2011 International Conference for High Performance Computing, Networking, Storage and Analysis (SC'11)*, No.47, 2011.

[17] J. Mars, L. Tang, R. Hundt, K. Skadron and M. L. Soffa. Bubble-Up: Increasing utilization in modern warehouse scale computers via sensible co-locations. In *Proceedings of the 44th Annual IEEE/ACM International Symposium on Microarchitecture (MICRO'11)*, pages 248-259, 2011.

[18] A. Roytman, A. Kansal, S. Govindan, J. Liu and S. Nath. PACMan: Performance aware virtual machine consolidation. In *Proceedings of the 10th International Conference on Autonomic Computing (ICAC'13)*, 2013.

[19] S. Kim, H. Eom and H. Y. Yeom. Virtual machine consolidation based on interference modelling. *Journal of Supercomputing, 66(3)*, pages 1489-1506, 2013.

[20] S. Verboven, K. Vanmechelen and J. Broeckhove. Black box scheduling for resource intensive virtual machine workloads with interference models. *Journal of Future Generation Computer Systems, 29(8)*, pages 1871-1884, 2013.

[21] SPECcpu2006 Benchmark. http://www.spec.org/cpu2006/Docs.

[22] J. Hamilton. Cooperative expendable micro-slice servers (CEMS): Low cost, low power servers for internet-scale services. In *Proceedings of the 4th Biennial Conference on Innovative Data Systems Research (CIDR'09)*, 2009.

[23] A. Amokrane, M. F. Zhani, R. Langar, R. Boutaba and G. Pujolle. Greenhead: Virtual data center embedding across distributed infrastructures. *IEEE Transactions on Cloud Computing, 1(1)*, pages 36-49, 2013.

[24] S. Albers. Energy-efficient algorithms. *Communications of the ACM, 53(5)*, pages 86-96, 2010.

[25] R. Nathuji, K. Schwan. VirtualPower: Coordinated power management in virtualized enterprise systems. In *Proceedings of 21st ACM SIGOPS symposium on Operating systems principles (SOSP'07)*, pages 265-278, 2007.

[26] D. Kusic, J. O. kephart, J. E. Hanson, N. Kandasamy and G. Jiang. Power and performance management of virtualized computing environments via lookahead control. *Journal of Cluster Computing, 12(1)*, pages 1-15, 2009.

[27] A. Beloglazov, J. Abawajy, R. Buyya. Energy-aware resource allocation heuristics for efficient management of data centers for cloud computing. *Journal Future Generation Computing Systems, 28(5)*, pages 755-768, 2012.

[28] M. Lin, A. Wierman, L. Andrew and E. Thereska. Dynamic right-sizing for power-proportional data centers. In *Proceedings of the 30th Annual IEEE International Conference on Computer Communications (INFOCOM'11)*, pages 1098-1106, 2011.

[29] F. Liu, Z. Zhou, H. Jin, B. Li, B. Li and H. Jiang. On arbitrating the power-performance tradeoff in SaaS clouds. *IEEE Transactions on Parallel and Distributed Systems, Vol.99*, 2013.

[30] L. Chen and H. Shen. Consolidating complementary VMs with spatial/temporal-awareness in cloud datacenters. In *Proceedings of the 33rd Annual IEEE International Conference on Computer Communications (INFOCOM'14)*, pages 1033-1041, 2014.

[31] F. Xu, F, Liu, H. Jin and A. V. Vasilakos. Managing performance overhead of virtual machines in cloud computing: A survey, state of the art, and future directions. *Proceedings of the IEEE, 102(1)*, pages 11-31, 2014.

[32] J. Esch. Prolog to "Managing performance overhead of virtual machines in cloud computing: a survey, state of the art, and future directions". *Proceedings of the IEEE, 102(1)*, pages 7-10, 2014.

[33] X. Fan, W. Weber and L. A. Barroso. Power provisioning for a warehouse-size computer. In *Proceedings of the 34th Annual International Symposium on Computer Architecture (ISCA'07)*, pages 13-23, 2007.

[34] M. Aazam and E. Huh. QoS degradation based reimbursement for real-time cloud communication. In *Proceedings of the 1st Workshop on All-web Real-Time System (AWeS'15)*, ACM, 2015.

[35] D. Jung, J. Lim, J. Gil, E. Lee and H. Yu. Task balanced workflow scheduling technique considering task processing rate in spot market. *Journal of Applied Mathematics*, pages 1-10, 2014.

[36] Y. Azar, N. Ben-Aroya, N. R. Devanur and N. Jain. Cloud scheduling with setup cost. In *Proceedings of the 25th Annual ACM Symposium on Parallelism in Algorithms and Architectures (SPAA'13)*, pages 298-304, 2013.

[37] J. Ekanayake and G. Fox. High performance parallel computing with clouds and cloud technologies. In *Proceedings of the 1st International Conference on Cloud Computing (CloudComp'09)*, pages 20-38, 2009.

[38] Amazon EC2. http://aws.amazon.com/ec2.

[39] Y. Koh, R. Knauerhase, P. Brett, M. Bowman, Z. Wen and C. Pu. An analysis of performance interference effects in virtual environments. In *Proceedings of the IEEE International Symposium on Performance Analysis of Systems and Software (ISPASS'07)*, pages 200-209, 2007.

[40] D. Bruneo. A stochastic model to investigate data center performance and QoS in IaaS cloud computing systems. *IEEE Transactions on Parallel and Distributed Systems, 25(3)*, pages 560-569, 2014.

[41] M. R. Garey and D. S. Johnson. *Computers and intractability: A guide to the theory of NP-completeness*. W.H. Freeman, New York, 1979.

[42] J. Leung. *Handbook of scheduling*. CRC Press, Inc., Boca Raton, FL, USA, 2004.

[43] https://aws.amazon.com/articles/1639.

[44] K. Ren, Y. Kwon, M. Balazinska and B. Howe. Hadoop's adolescence: An analysis of Hadoop usage in scientific workloads. *Journal of VLDB Endowment, 6(10)*, pages 853-864, 2013.

Xibo Jin He is a PhD candidate of the Institute of Computing Technology, Chinese Academy of Sciences, and a student member of IEEE. His research interests include parallel computing, Cloud computing, scheduling algorithms, and energy-efficient computing.

Fa Zhang He is an associate professor of the Institute of Computing Technology, Chinese Academy of Sciences. His research interests include high performance algorithms, green computing and bioinformatics.

Lin Wang He is a PhD candidate of the Institute of Computing Technology, Chinese Academy of Sciences, and a student member of IEEE. His research interests include green network computing, routing algorithms and energy-efficient computing.

Songlin Hu He is an associate professor of the Institute of Computing Technology, Chinese Academy of Sciences. His research interests include distributed event based system, service computing and big data processing.

Biyu Zhou She is a PhD candidate of the Institute of Computing Technology, Chinese Academy of Sciences, and a student member of IEEE. Her research interests include green network computing, routing algorithms and energy-efficient computing.

Zhiyong Liu He is a professor of the Institute of Computing Technology, Chinese Academy of Sciences, and a senior member of China Computer Federation. His research interests include high performance algorithms and architecture, and parallel processing.