Performance Interference of Virtual Machines: A Survey

WEIWEI LIN, South China University of Technology and Peng Cheng Laboratory
CHENNIAN XIONG*, South China University of Technology
WENTAI WU*, Peng Cheng Laboratory
FANG SHI, South China University of Technology
KEQIN LI, State University of New York
MINXIAN XU, Shenzhen Institute of Advanced Technology, Chinese Academy of Sciences

The rapid development of cloud computing with virtualization technology has benefited both the academia and the industry. For any cloud data center at scale, one of the primary challenges is how to effectively orchestrate a large number of virtual machines (VMs) in a performance-aware and cost-effective manner. A key problem here is that the performance interference between VMs can significantly undermine the efficiency of cloud data centers, leading to performance degradation and additional operation cost. To address this issue, extensive studies have been conducted to investigate the problem from different aspects. In this survey, we make a comprehensive investigation into the causes of VM interference and provide an in-depth review of existing research and solutions in the literature. We first categorize existing studies on interference models according to their modeling objectives, metrics used and modeling methods. Then we revisit interference-aware strategies for scheduling optimization as well as co-optimization based approaches. Finally, the survey identifies open challenges with respect to VM interference in data centers and discusses possible research directions to provide insights for future research in the area.

CCS Concepts: • **General and reference** → **Surveys and overviews**; • **Computer systems organization** → **Cloud computing**.

Additional Key Words and Phrases: Cloud data center, VM performance interference, measuring and modeling, scheduling optimization

1 INTRODUCTION

1.1 Background

Cloud computing has gained a great popularity due to its business-critical features like reliable performance, high scalability and on-demand. In the past decade, the cloud computing industry has been booming since many Internet giants launched their own cloud platforms, such as Amazon's AWS, Google's Google cloud, etc. Cloud computing is recognized by the industry as an important technical driving force in the digital era, as well as a critical way to realize digital transformation for traditional enterprises. The market continues to expand with the ever-increasing demands of cloud computing services. According to Gartner's analysis, global cloud revenue to total $474 billion in 2022, up from $408 billion in 2021 [1]. In addition, more than half of enterprise IT spending in key market segments will shift to the cloud by 2025 [2].

Virtualization technology is the key technology to support cloud computing platform. By utilizing this technique, the computing resources of a single physical server (PM) can be virtualized into multiple isolated computing domains managed by a virtual machine monitor (VMM). Virtualization reduces the coupling among hardware resources and system or user software, leading to improving the utilization of computing resources [3]. Moreover, due to the dynamic and open nature of virtualization, it is easier to make cloud data center's management and maintenance in a better manner for load balancing and energy saving by online migration [4]. Despite its benefits, server virtualization also brings new problems, such as additional

*corresponding authors

Authors' addresses: Weiwei Lin, linww@scut.edu.cn, South China University of Technology, Guangzhou, China, 510006, Peng Cheng Laboratory, Shenzhen, China, 518000; Chennian Xiong, 1119432501@qq.com, South China University of Technology, Guangzhou, China, 510006; Wentai Wu, nnwtwu@pcl.ac.cn, Peng Cheng Laboratory, Shenzhen, China, 518000; Fang Shi, csshifang@mail.scut.edu.cn, South China University of Technology, Guangzhou, China, 510006; Keqin Li, lik@newpaltz.edu, State University of New York, New Paltz, New York, 12561; Minxian Xu, mx.xu@siat.ac.cn, Shenzhen Institute of Advanced Technology, Chinese Academy of Sciences, Shenzhen, China.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.
© 2022 Association for Computing Machinery.
0360-0300/2022/11-ART $15.00
https://doi.org/10.1145/3573009

Just Accepted

resource consumption and potential resource contention which inevitably leads to performance interference between VMs [5]. Any virtualization technology cannot guarantee perfect resource and performance isolation. As a result, the applications running on multiple VMs compete for the underlying resources of the host. Particularly, when the VM instances hosted by a same physical server tend to have a strong conflict of resource demands, the performances of the VMs are likely to degrade seriously.

With the increasing number and density of virtual machines within a single data center, performance degradation caused by the mutual interference of VM instances has been a major concern. Analysis on Amazon EC2 platform shows that, due to VM interference, disk IO bandwidth of EC2 server can drop by as much as 50% [6], whilst the network IO bandwidth can decrease by as much as 60% [7]. Studies also found that 90% running time of MapReduce applications is largely affected by the interference between CPU-intensive and IO-intensive operations [8]. In the meantime, performance interference has a certain impact on the energy efficiency of cloud data centers [9]. In light of this, we argue that how to effectively detect and measure performance interference and how to prevent or mitigate it through VM management are the key to improve cloud servers' quality. Nonetheless, there are many challenges [10–12] in the detection, measurement and alleviation of VM interference, which can be summarized as follows:

(1) The complexity in the causes of performance interference: performance interference is mainly caused by resource contention, however, it requires a comprehensive understanding on VM's feedbacks to the contention of different resources and dynamics of workloads. The causes and consequences of performance interference can significantly differ with different workload types.
(2) Real-time constraints: for performance-sensitive cloud applications, prompt decisions have to be made once the performance interference occurs. This means that detection, prediction and scheduling methods are required to achieve real-time scheduling.
(3) Limited information access: there is a semantic gap between VMs and the guest OS, which makes it difficult to accurately measure and allocate the resources needed by VM. Out of security and portability concerns, the information acquisition of internal applications inside the VM is very much limited.
(4) Low-overhead intervention: for minimal intrusion to the application runtime environment and ensuring smooth operations, the overheads of detecting, measuring and mitigating VM interference need to be as low as possible.

We detail the contributions of our work and draw comparison to existing surveys on performance interference in Section 1.3.

1.2 Source of Cited Articles

We review a broad range of research articles indexed by mainstream academic databases including IEEEXplore, Springer, Elsevier, ACM Digital Library. We chose "Performance interference", "VM interference", "resource contention" , "interference in cloud", "interference-aware", etc. as the keywords in titles and abstracts and filtered the results by the relevance of topic. In addition, we also extend our search to include articles that do not feature these keywords but have specific content discussing performance interference in the context. In terms of the year of publication, 64% of the cited articles were published in or before 2017 and 36% were published in or after 2018. In terms of the article sources, journal papers account for 44% of the whole collection, 53% are in conference proceedings, and the rest comes from other sources of publication such as books and dissertations. The distribution of article sources is shown in Fig 1.

1.3 Comparison with Existing Surveys

This section summarizes relevant surveys on performance interference. It is worth noticing that a number of surveys on VM scheduling optimization [13–16] provide analysis related to the interference between VMs, but most of the discussions are conceptual or limited to one or two aspects. In-depth analysis of the problem is not included in these papers. We retrieved four surveys specifically focused on VM interference. In Table 1 we compare our survey with these works in multiple aspects such target scenarios and interference metrics analyzed.

Surveys [17, 18] focus on performance interference during the process of VM migration. Bloch et al. [17] analyzed the interference in the process of VM migration in several categories including co-located VM Interference, network interference and application performance interference. They summarize the articles with focus on the classification of performance interference in the VM migration process. Similarly, relevant studies on how to minimize the resource interference during

Just Accepted

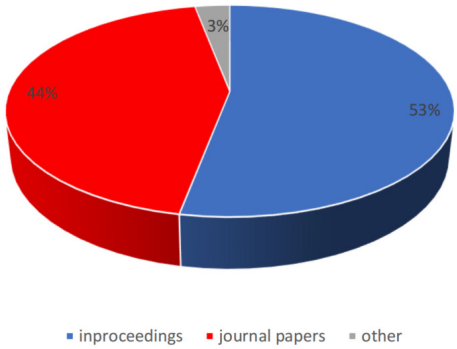


Fig. 1. Distribution of article sources

Table 1. Comparison with the Existing Surveys

Survey	Studied scenarios	Metrics analyzed		Summarize modeling and optimization respectively	Focuses of content	Year
		Independent metrics	Derived metrics			
[17]	VM migration	no	no	no	Classification of interference in VM migration such as co-located interference, network interference and application interference	2014
[18]	VM migration	no	no	no	The mechanism of performance interference in the process of VM migration	2018
[19]	All (VM migration, VM placement, etc)	yes	yes	no	Performance interference optimization: learning based approaches and queuing based approaches	2017
[20]	All (VM migration, VM placement, etc)	no	yes	yes	Categorization of studies by scenarios from the single-server virtualization to geo-distributed data centers	2014
This survey	All (VM migration, VM placement, etc)	yes	yes	yes	Interference modeling reviewed from three aspects: modeling objectives, modeling metrics and modeling methods. Interference-aware optimization from two aspects: VM placement optimization and VM consolidation optimization	2022

Just Accepted

live migration of VMs are summarized in [18]. These two surveys focus their scope of investigation on the interference caused by VM migration while in-depth analysis on interference detection and modeling are missing.

It is necessary to extend the discussion to more scenarios on the cloud. For example, the placement of VMs will also interfere with the co-located VMs while the change of workload can be another factor leading to performance interference. In this regard, Amri et al. [19] summarized existing performance interference optimization methods into learning based approaches and queuing based approaches. However, this paper does not distinguish between modeling methods and optimization methods in the literature. The two aspects are connected to each other but the methodologies and principles behind can be very much different. Separate views of interference modeling and optimization are important for better understanding of the problem and the solutions. Xu et al. [20] presented a comprehensive review of the modeling methods and optimization

methods, but the majority of the papers surveyed in their work were published before 2014. Analysis of interference related metrics is the fundamental part of interference modeling. There has been much discussion on which metrics are affected by interference and which metrics are suitable to measure the degree of interference or performance loss. However, most of the existing surveys [17] and [18] do not provide sufficient insights to address these concerns. Some of the commonly-used interference-indicating metrics are introduced in [20] and [19]. Our work extends the discussion covering more aspects from metric selection to data collection.

Compared with the existing surveys, we in this paper present a more comprehensive review covering the modeling and the optimization for VM interference in data centers. The scope of survey by Xu et al. [20] is the closest to that of our paper. They summarize the related works based on diverse scenarios including single-server virtualization, mega data center, and geo-distributed data centers. From a different perspective, we categorize relevant studies on interference modeling by their modeling objectives, the modeling metrics, and the modeling methods. Within each category, we provide in-depth analysis, comparison of studies as well as concrete examples. For interference-aware optimization, we summarize the latest approaches by grouping them into placement optimization and consolidation optimization.

1.4 Our Contributions

Existing studies on performance interference mainly focuses on one of the following aspects: i) Analysis of the causes of performance interference, ii) methods to assess performance interference qualitatively or quantitatively, and iii) approaches to mitigating or avoiding performance interference. In this paper, we make a comprehensive survey that covers all the three aspects. The overall structure of this survey study is shown in Fig. 2.

Our main contributions are summarized as follows:

(1) We present a formal definition of VM performance interference followed by an analysis of the causes of performance interference.
(2) We summarize and compare existing studies on the detection and assessment of performance interference by their objectives of modeling, the metrics used for modeling and the modeling methods.
(3) We review existing interference-aware VM scheduling optimization approaches and categorize them as placement strategies and consolidation strategies. We illustrate the corresponding workflow and analyze where existing studies can be applied.
(4) We outline several research issues that remain to be resolved and possible directions for further study on VM performance interference.

1.5 Article Structure

The rest of this article is organized as follows: Section 2 gives the definition of performance interference and analyzes the causes. Section 3 summarizes the methods of measuring or modeling performance interference. Section 4 reviews existing interference-aware VM scheduling optimization strategies. In Section 5, we discuss open issues and possible future directions for future research. We conclude the article in Section 6.

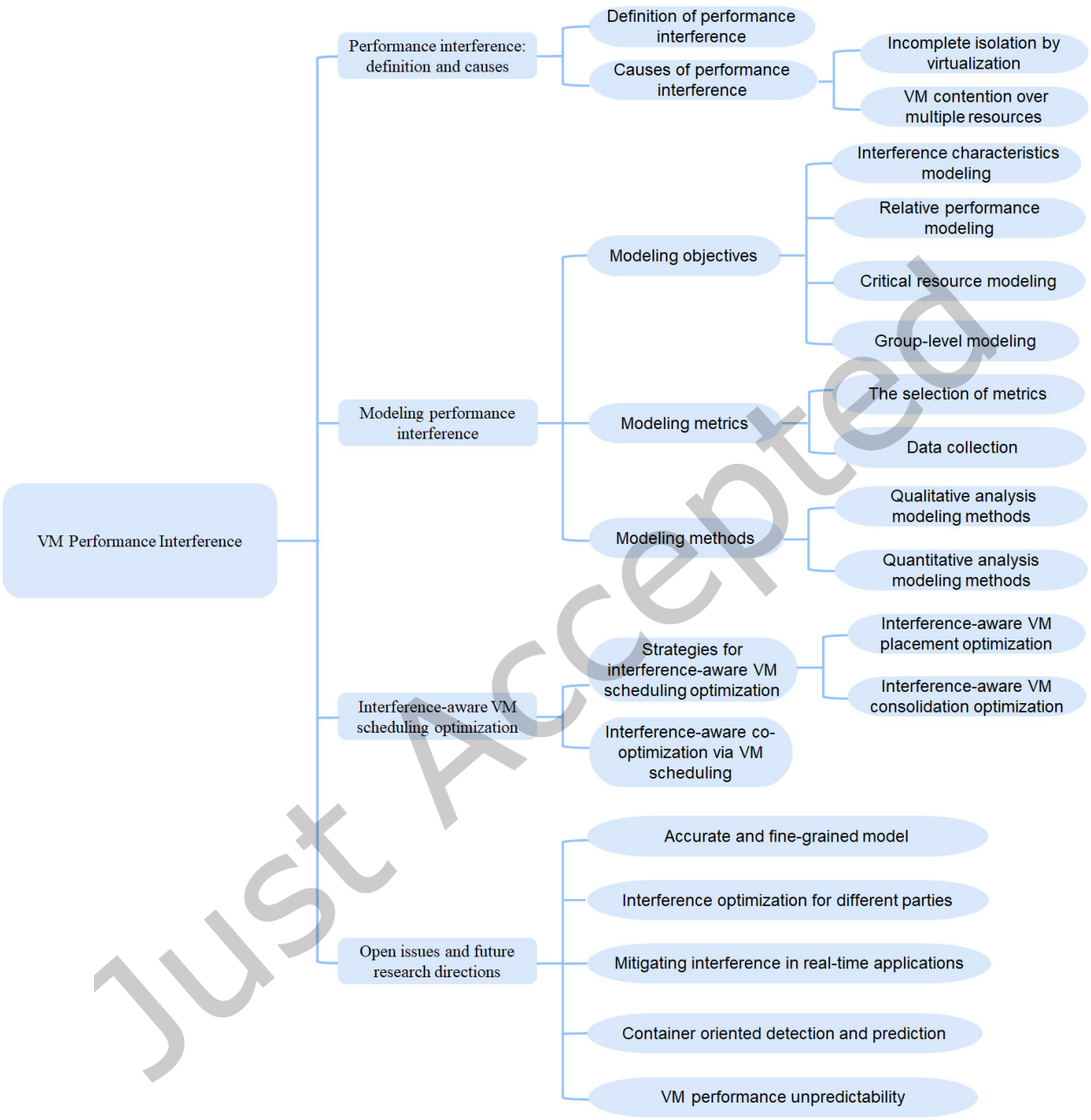
2 PERFORMANCE INTERFERENCE: DEFINITION AND CAUSE

In this section, we provide the definition of performance interference and analyze the possible causes of performance interference between VMs.

2.1 Definition of Performance Interference

The term 'performance' in this paper refers to the ability of VMs or applications to maintain the intended service level. A number of metrics can be used to represent performance. For example, task completion time can be used to measure the performance for batch applications while request response time can be used to measure the performance for interactive application. For IO-intensive VMs, their performance can be measured by IO throughput. Given ideal conditions, the performance can be predictable. However, in some practical scenarios, the performance of a task or a VM is unable to reach the expected level predicted based on the resource capacity of the host. Some studies define this fluctuation as performance unpredictability [21] and there are many reasons leading to the problem. The main reason behind performance unpredictability is the resource contention between co-located VMs due to incomplete resource isolation by the virtualization technology.

Just Accepted



Just Accepted

Fig. 2. The structure of this survey on VM performance interference and scheduling-based optimization in cloud data centers

VMs compete with each other for the underlying resources, which leads to the shortage of the actual resources provided by the VM to load and consequently the fluctuation of actual performance [5].

Since interference is mutual, the performance of VM may be affected by other co-located VMs. The measurement of the performance loss of one of the co-located VMs due to interference is given by

$$PD(A@B) = \frac{P_B^A - P_{alone}^A}{P_{alone}^A} \tag{1}$$

where A and B represents two co-located VMs, P_B^A represents the actual performance of VM A under mutual interference with VM B, P_{alone}^A represents the ideal performance of VM A when A runs alone, $PD(A@B)$ evaluates to what extent VM A's performance is influenced by the presence of VM B.

The selection of the performance metric is the key to accurately reflecting the intensity of interference (which will be introduced in details in Section 3.2). Depending on the system and applications, some metrics cannot serve as a good indicator to quantify the change of performance shown in Eq. (1), yet one can assess whether the performance interference occurs by observing and comparing the values of multiple metrics under ideal and actual conditions. For example, we can use a set of metric to classify the VMs working in an ideal condition without interference, and then by clustering we can assess whether performance interference occurs given new observations of these metrics [22]. Also, we can make use of outlier detection methods, where the VM showing abnormal values of the relevant metrics can be considered being interfered by other VM instances [23].

2.2 Causes of Performance Interference

Virtualization technology is the key supporting technology of cloud data center, playing an important role in the performance improvement and resource optimization of cloud data center. However, the incomplete performance isolation leads to resource contention between co-hosted VMs, resulting in the problem of performance interference.

2.2.1 Incomplete isolation by virtualization. Virtualization offers critical features such as dynamic scaling and on-demand resource provisioning [24]. With the advantage of virtualization technology, live migration and quick restart of VM make the management of large data center more convenient. In addition, virtualization provides strong isolation among virtual domains. For example, security isolation can prevent malicious attacks from other domains; error isolation can prevent abnormal applications from damaging the entire system; and environmental isolation allows multiple operating systems to run on the same computer. However, performance isolation is never 100 percent guaranteed. Under ideal conditions, running application inside a VM is like running that on a separate PM from the user's perspective because each VM runs a separate operating system which is called guest OS. However, the incomplete resource isolation in the shared use of resources in the system hinders its implementation.

Fundamentally, VMs share the physical resources provided by its underlying host. If a VM runs short of resources, it will compete for the underlying resources, which results in the performance of other VMs being affected. As a result, the performance of the application inside a VM can be susceptible to constant deviation from that it is expected to have and dependent on not only the capacity of some specific types of resources but the number of co-located VMs that compete for them. At present, the mainstream VM monitoring managers (XenServer, vSphere, hypervisor) can achieve certain degree of performance isolation by configuring a dedicated physical CPU core and non-overlapping memory and disk space resources for VMs [3]. However, the improvement is usually marginal because it is difficult to isolate some shared hardware resources, such as CPU cache and network or disk IO bandwidth [25]. In general, contention on shared resources can be eliminated through explicit resource isolation such as cache allocation technology (CAT) and memory bandwidth allocation (MBA). However, explicit allocation results in poor system-wide performance if contention for resources is only minor. Additionally, there is no interface for some resources to explicitly manage such as bus locks [26]. Therefore, the contention of these hardware resources is the main cause of performance interference. Above all, under the current technical conditions, the contention of the resources such as computing, network and memory will inevitably affect all VMs in the same host node to a certain extent.

2.2.2 VM contention over multiple resources. Contentious resources sharing among VMs is the major cause of performance interference. The more similar two VMs are in resource demand, the stronger the performance interference is. Therefore, an intuitive approach is to use the similarity of resource demands to judge whether the VMs are suitable for co-location operation [27]. In addition, cloud data center generally adopts over-subscription to maximize the utilization of resources, which results in the resource demand by applications deployed on the server exceeds the resources actually owned by the server [28]. For example, allocating more vCPU cores to the VMs than the actual number of cores can create a deficit in cores available, which intensifies the resource contention between applications and leads to potential performance interference between them. The contentions over different types of resources (shown in Fig. 3) can have different impacts on VM performance interference and the mechanisms behind can be quite different.

Just Accepted

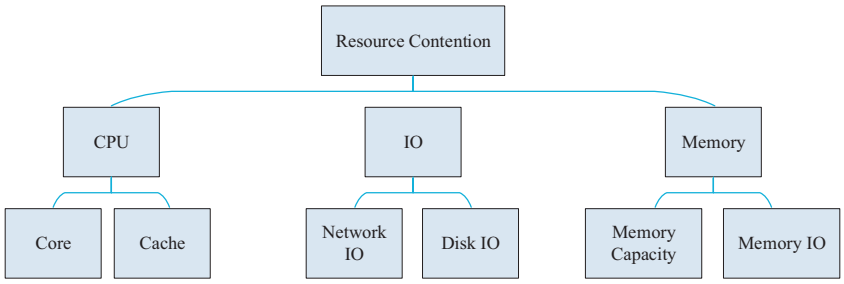


Fig. 3. Resource contention factors of performance interference

The contention for CPU resources. CPU-intensive applications account for a large proportion of cloud tasks. They mainly consume computing resources but occupy little IO resources. The main factor of performance interference between CPU-intensive applications is the contention for cores and cache space. In a virtualized environment, each VM has its own vCPU, but only when they occupy the physical CPU can they have computing ability. However, each physical CPU can only be used by one vCPU at any time. If the time slice allocated to the vCPU of the VM is insufficient, the normal execution of the task can not be guaranteed, and even the task may be delayed. As for caching, current VM architecture does not provide isolated CPU cache space, and the applications deployed on the host will share the last level cache, resulting in that the miss rate of the last level cache in the situation of co-located operation is much higher than that of running alone. An increase in the value of cache miss ratio reflects that most of the data in the CPU is read out of memory, which leads to longer task run time, resulting in performance degradation [29]. In addition, the priority of tasks accessing the CPU also has an impact on performance. For example, if a high priority application and a low priority application are deployed together, the high priority application can always obtain CPU resources whilst the low priority application needs to wait for the high priority application to finish running to obtain CPU resources, which will lead to greater performance interference.

The contention for IO resources. Contention can happen over local IO resources (e.g., disk access) and remote IO resources (e.g., sockets). Network IO is responsible for communicating with remote devices, and disk IO is responsible for accessing external storage devices. Multiple applications running on the server will inevitably share the disk bandwidth and network bandwidth, and the actual occupied bandwidth of VMs which co-locate with other VMs will inevitably be less than the bandwidth that they can occupy when running alone. This will inevitably slow down the speed of data exchange, and inevitably affect the speed of task execution, and even task requests will starve to death due to lack of sufficient bandwidth resources [30].

The contention for memory resources. Although memories are typically empowered by non-overlapping technology, contention inevitably happens between co-located VMs when there is insufficient memory capacity on the host. Besides, the memory bandwidth responsible for communication with CPU will also cause performance interference due to contention. The principle is similar to the that of IO devices, and memory bandwidth is the main reason of memory resource interference. Memory capacity and bandwidth are determined by hardware, yet memory contention can be reduced to a certain extent through the management and configuration of memory resources [31, 32], which makes a low-cost approach to the reduction of performance interference intensity and frequency.

3 MODELING PERFORMANCE INTERFERENCE

There are several ways to model the impact of performance on a given instance of VM by its co-located VMs. The model reflects the intensity of resource contention at different levels of the system from the application to the entire cloud service system. It is important for the service provider to model and measure the actual performance of their virtual servers before planning and making any scheduling or scaling decisions in a cloud data center.

We introduce the typically ways to model performance interference and summarize existing studies in three aspects: modeling objectives, modeling metrics, and modeling methods. In Section 3.1, we classify the studies based on modeling objectives which are associated to the characteristics of a single VM, a group of VMs, and interference-related resources. In Section 3.2, we introduce the process of building an interference assessment model from the selection of indicative metrics to

Just Accepted

the collection of data. In Section 3.3, we summarize commonly-used models and methods for both qualitative and quantitative analysis in the assessment of performance interference.

3.1 Modeling Objectives

Performance interference can be evaluated from different perspectives at different levels. Through reviewing existing studies, we found four types of objectives. The first objective is to model the interference characteristics of a single application or VM, and the second is to model the actual performance or the performance loss with the presence of interference. The first two objectives are useful for interference awareness optimization based on the metrics of a single application or VM. The third kind of modeling objectives is group-level modeling, wherein performance interference is examined over a group of VMs during migration or a group of resource-competing applications. The fourth model objectives aims to model interference from the perspective of resources. Because performance interference is caused by resource contention, resource-based models can be used to solve the problem of performance interference by re-allocating the critical resource. We list some of the works mentioned in this subsection in Table 2.

Table 2. Comparison of Studies based on Modeling Objectives

Ref.	Modeling objectives	Application scenarios	Use of models
[33]	Interference characteristics	For co-located applications that are memory-intensive and share last-level cache	Quantify how susceptible an application is to be interfered by other applications, and quantify how likely one application is to interfere with another
[34]	Interference characteristics	For co-located VMs that are cache intensive	Quantify how susceptible a VM is to be interfered by other applications, and quantify how likely a VM is to interfere with another
[35]	Relative performance	For MapReduce clusters	Predict the actual performance of MapReduce application under interferences
[5]	Relative performance	For the application *analyzer* operating under interferences	Predict the actual performance of application *analyzer* under interference conditions
[36]	Critical resources	For any types of VMs	Determine the interference caused by competing for the four resources according to VM usage
[22]	Critical resources	For any servers	Determine the main interference source of the server
[37]	Group-level modeling	For VM migration process	Quantify the degree of interference in the VM migration process
[38]	Group-level modeling	For co-located interactive applications with the delay-insensitive application	Quantify the overall performance interference degree of the two types of application under

3.1.1 *Interference Characteristics Modeling.*

Concepts of interference intensity and interference sensitivity. From the perspective of applications and VMs, performance interference is mutual but asymmetric. Different applications or VMs have different interference characteristics, some are easy to be interfered, and some are easy to interfere with co-located instances. In order to differentiate them, studies [33, 34] defined two concepts: interference sensitivity and interference intensity. Interference sensitivity is a measure of how much an application or a VM will suffer when co-located with others and interference intensity is a measure of how aggressively of an application or a VM when occupying resources. These two concepts can be expressed by a rough estimation and can be used for scheduling. For example, if the value of interference intensity of application APP_A to application APP_B is 3, and that to application APP_C is 4, obviously, APP_C is subject to the greatest performance interference of APP_B evidently. Therefore, the node which APP_A is running in should first schedule APP_B instead of APP_C. The value of interference sensitivity is used in the opposite way.

Examples of interference characteristics. There is a intricate relation between interference intensity and interference sensitivity. For example, Kim et al. [33] modeled the interference characteristics based on the shared last-level cache(LLC) and memory bus of applications in the cloud. For interference intensity, the authors found that the interference intensity is related to two metrics including the LLC miss rate (the number of misses per second) and the LLC miss ratio (the number of

Just Accepted

LLC misses per LLC reference). The metric positively correlated with the interference intensity is taken as the numerator, and the metric negatively correlated with the interference intensity is taken as the denominator. Therefore, the interference intensity of application A to co-located application B is given by

$$\Delta R_{LLC-miss,A|B} \propto I = r_{LLC-miss,B} \times \sqrt{\frac{1}{R_{LLC-miss,B}}}, \tag{2}$$

where $R_{LLC-miss,A|B} \propto I$ represents the interference intensity of application A to application B, $r_{LLC-miss,B}$ represents the LLC miss rate of application B and $R_{LLC-miss,B}$ represents the LLC miss ratio of application B. As for the interference sensitivity of application A itself, authors think that it is determined by two metrics, the LLC miss ratio $R_{LLC-miss,A}$ of the application itself and the proportion of LLC miss in the whole running $R_{stall,A}$. The equation of the sensitivity S of application A is given by

$$S = (1 - R_{LLC-miss,A}) \times R_{stall,A}. \tag{3}$$

Chen et al. [34] studied the VM interference sensitivity and intensity on cache resources. Authors think that the higher the proportion of the total access number of least recently used (LRU) cache of VM A to all co-located VMs, the higher the cache interference sensitivity of VM A. Besides, the higher the number of cache misses of VM A, the higher the impact of VM A on other co-located VMs which means the cache intensity is higher.

3.1.2 *Relative Performance Modeling.*

Concept of normalized score. Many works proposed to measure interference by comparing the performance under ideal conditions and interference conditions by introducing the concept of normalized score or normalized performance. Two common forms of scores exist in the literature. The first one is the ratio of the performance running in the actual environment to the performance running alone under ideal conditions. For example, Koh et al. [5] proposed a normalized performance equation based on task completion time which is shown in Eq. (4). Another form of normalized performance is the ratio of the difference between the performance running in the actual environment and the performance running alone under ideal conditions and the latter, such as the normalized performance formula based on CPI metric proposed by Chen et al. [39] shown in Eq. (5).

$$NS(F@B) = \frac{Performance(F@B)}{Performance(F@Idle)}, \tag{4}$$

$$NS(F@B) = \frac{Performance(F@B) - Performance(F@Idle)}{Performance(F@Idle)}, \tag{5}$$

where $NS(F@B)$ represents the normalized score, $Performance(F@B)$ represents the performance in the actual environment and $Performance(F@Idle)$ represents the performance running alone under ideal conditions. The performance metrics can be selected by a variety of metrics which can intuitively reflect the actual operating condition, such as: running time [5, 40, 41], application throughput [42], online application request response time [43] and QoS [44], etc. However, due to the black-box nature of tasks and privacy terms, it is not easy for researchers to obtain these metrics in some cloud environments. Thus, some metrics of the system bottom layer[22, 39, 45], such as CPI (cycle per instructions) and MIPS (Million Instructions Per Second), are also widely used in performance interference models to represent performance because of their versatility and easy access.

Examples of relative performance models. Modeling performance considering interference factor consists of two ways. One is to model the left part of Eq. 4 or Eq. 5. The other is to model the actual performance under interference, that is, to model the numerator on the right-hand site of Eq. 4 or Eq. 5. After investigation, we think that the first way accounts for the majority. Whether the first method or the second method, the process of modeling is to establish the relationship between the selected independent metrics and derived metrics.

For example, Bu et al. [35] built the normalized score model of MapReduce. Because MapReduce is a CPU-intensive and IO-intensive application, authors selected CPU-related metrics and IO-related metrics as independent variables to predict the normalized score of MapReduce $\hat{S}$. $\hat{S}$ is defined on the task's actual completion time (T_{real}) over the run time without interference (T), $\hat{S} = T_{real}/T$. The modeling equation is given by

$$\hat{S} = \alpha\hat{S}_{cpu} + \beta\hat{S}_{io} + C, \tag{6}$$

Just Accepted

where $\hat{S}_{cpu}$ is a formula composed of CPU-related metrics, $\hat{S}_{io}$ is a formula composed of IO-related metrics and C is the parament.

As another example, Koh et al. [5] selected 10 corresponding metrics as independent variables for the normalized score model of the application '*analyzer*' when co-located with other load B, and then used the linear regression method to establish the relationship between these metrics and the normalized score. The selected metrics and coefficients are shown in Table 6, and the normalized score is given by

$$\overline{NS(analyser@B)} = a_0 + a_1 \cdot X_1 + a_2 \cdot X_2 + \cdots a_{10} \cdot X_{10}, \tag{7}$$

where X_1 to X_2 represent the metrics selected and $a_i (i = 1, 2, \ldots, 10)$ are parameters.

3.1.3 Critical Resource Modeling. Resource contention usually occurs when multiple co-hosted VMs have intensive usage on a specific type of resource or a group of resources. Hence, a general approach is to detect VM performance interference through monitoring the resource usage and quantifying its association with interference. Peng et al. [36] established a profile for VM which contains CPU, MEM, BW, IO and other resources. Each type of resource has three metrics, namely, the average resource utilization avg of the load running on the VM, and the proportion of the load resource utilization exceeding the threshold value. With these metrics, authors can judge the resource usage of the VM, and then judge whether the load on the VM has caused performance interference due to contention for a certain resource. Authors divide the interference into three categories: no interference, medium interference and interference which are shown in Table 3. $P_{warning}$ represents the threshold values of the selected metrics and *avg* represents the average values of the selected metrics.

Table 3. Resource Contention Grouping

	CPU	MEM	BW	IO
Intensive	$P_{warning} > 10$ %	$P_{warning} > 90\%$	$avg > 2\%$	$avg > 2\%$
Medium-intensive	$0 < P_{warning} \leq 10\%$	$10\% < P_{warning} \leq 90\%$	$0.6\% < avg \leq 2\%$	$0.4\% < avg \leq 2\%$
No-intensive	$P_{warning} = 0$	$P_{warning} \leq 10\%$	avg $\leq 0.6\%$	avg $\leq 0.4\%$

By analyzing the resource usage, the critical resource of contention that causes interference can be identified. For example, based on the percentage of time that different resources were accessed during task execution, Novakavic et al. [22] analyzed the server's current contention for the most intense resources. The running time of the whole server is divided into four parts, which are the time of the CPU core running instructions T_{core}, the time waiting for memory access operations $T_{off_{core}}$, the time waiting for disk reads T_{disk}, and the network-related occupation time T_{net}. The first two can be obtained by CPI analysis, and the latter two can be obtained by system data. The whole running time is shown as

$$T_{overall} = T_{core} + T_{off_{core}} + T_{disk} + T_{net}. \tag{8}$$

The authors further proposed a resource contribution model to identify the type of resource most likely to be the main source of interference. The model is given by

$$Factor_R = \frac{T_R^{production} - T_R^{isolation}}{T_{overall}^{production}}, \tag{9}$$

where $T_R^{production}$ represents the time occupied by resource R in the actual environment, the $T_R^{isolution}$ represents the time occupied by resource R in the ideal environment and $T_{overall}^{production}$ represents the time occupied by all resources in the actual environment. The model compares the time occupied by resource R in the actual environment with in the ideal situation. The larger the ratio is, the more intense the contention for resource R is. In addition, Javadi et al. [46] used a decision tree based classifier to find the main interference source. They conducted controlled interference experiment by using micro benchmark and trained with the monitored data in each case. After training, the decision tree can classify interference sources according to the observed measurements, which are easy to observe, such as CPU utilization, IO waiting time.

The abovementioned models work by monitoring the system and cannot predict the future resources usage. Towards a proactive solution, Barve et al. [47] used random forest regression to predict the resource utilization of applications during execution. They calculated the actual execution time of the application based on the resource usage predicted by the model. Besides, Chen et al. [48] proposed a method to predict each resource contention statue based on Markov chain which is introduced in Section 3.3.5. In addition, Tseng et al. [49] designed a dynamic prediction model for VM resource utilization

Just Accepted

in cloud data center, which used a population-based multi-objective genetic algorithm (GA) to predict the CPU utilization, memory utilization and other resource utilization of VM at the future time t. Mehmood et al. [50] combined KNN and DT to predict the memory and CPU usage of workload. The above two references are only for reference. However, the latter two models did not take the interference factors into account. So these two studies are for your reference only.

3.1.4 Group-level Modeling. A branch of existing studies focus on measuring the overall performance interference at the level of the entire cloud data center across multiple entities. A group-level interference model can provide some holistic information for interference-aware VM optimization from the standpoint of overall interests of all participants.

Grouping VMs in Migration. In the process of VM migration, interference can happen due to interplay between source PM, target PM, and the VM to migrate. Thus, the performance interference analysis for VM migration should not only take into account the optimization of a VM, a PM or a task, but also all the participants in the migration event. For example, Xu et al. [37] formulate the migration interference as

$$M = f_M(\omega_s, \omega_d, t_i) \approx \omega_s + \omega_d + \kappa_t \times t_i. \tag{10}$$

where M quantifies the migration interference, ω_s represents the migration interference on source PM, ω_d represents the migration interference on target PM and t_i represents migration time. We can observe that this formula consists of three parts and the first two parts represent the interference of source PM and destination PM respectively because the VMs on the source and destination PMs will suffer serious performance fluctuations due to the extra resource consumption in the progress of VM migration. With this model, the performance interference of all parties involved in the VM migration process can be considered and reduce migration loss is more favorable.

Grouping Specific Types of Workload. In most interference awareness optimization, the optimization object is not limited to a single application and VM. Many works aggregated the individual models of applications or VMs to calculate overall interference, but some studies directly model the overall interference measurement. For example, in some cloud scenarios, application types are limited to specific types. In this case, some works considered all running applications as a group to measure performance interference. R. Shaw et al. [38] modeled the co-location interference of interactive load and delay-insensitive load. Authors think that the number of SLA violation of interactive load increases exponentially with the increase of load allocated to all PMs and the number of PMs used because interactive load needs faster response time. While the delay-insensitive load is often more flexible to performance interference, so the number of SLA violation only increases linearly with the increase of the number of loads running on VMs. The total performance interference degree imposed by two kinds of loads on the data center is given by

$$W_j = \frac{1}{M} \sum_{j=1}^{M} (\sum_{k=1}^{N} v_{kij})^x + (\sum_{k=1}^{N} v_{kdj})^x, \tag{11}$$

where M represents the number of active PMs in the data center, n represents the total number of VMs, v_{ki} represents the number of interactive VM loads, v_{kd} represents the number of delay-insensitive loads, k represents the index of VMs, j represents the index of PMs and is the index value corresponding to different types of VMs. With the help of this model, the author proposed a VM-PM mapping algorithm for two different types of load to avoid interactive load violating SLA due to delay.

3.2 Metrics of Interference

3.2.1 The selection of metrics.

Common metrics. There are two types of metrics that can be used to detect and measure the performance interference between VMs: independent metrics and derived metrics. The independent metrics refer to the variables directly sampled as observations and can serve as indirect indicators for performance interference. The derived metrics are variables that can directly measure the intensity of interference or the level of performance loss caused by interference.

The selection of independent metrics is usually associated to the critical resources. Existing studies typically follow two paths. One is to directly select the metrics based on the main resources demand. For example, for CPU-intensive applications, CPU-related metrics should be selected, such as CPU cycle, cache miss rate, vCPU utilization rate, etc. [51]. And as for IO-intensive applications, the common selected metrics are IO throughput, bandwidth, read /write or transmission rate [52],

Just Accepted

etc. The other path is to trace the access process of the mainly consumed resources [48] and we will take the Ref. [48] as an example of the second path to introduce below. The commonly selected metrics for different resource intensive applications or VMs are shown in Table 4. The selected metrics can be PM-level data or VM-level data. For example, the utilization of vCPU and the utilization of pCPU can be selected to reflect 'CPU utilization' in Table 4.

Table 4. Common Metrics that Reflect the Usage of Critical Resources

Resource	Utilization indicators
CPU	CPU utilization, CPU cache, CPU cycle
Memory	memory utilization, idle amount and memory bandwidth
Disk IO	disk utilization, partition utilization, disk read / write IO data
Network IO	in and out traffic, packet loss rate and network bandwidth

Derived metrics are results of calculation based on some performance metrics, as shown in Eq. (4) and Eq. (5). Some works used application performance as a performance metric such as running time [5, 40, 41], application throughput [42], online application request response time [43] and QoS [44] because these metrics can directly display the impact of performance interference on the performance of cloud system . Another works used some metrics of the system bottom layer [22, 39, 45] as a performance metric such as CPI (cycle per instructions) and MIPS (Million Instructions Per Second). These bottom layer metrics are not read from application-level monitoring, but from the relatively bottom level. However, it is these metrics such as MIPS that determine the application performance, so these metrics can be used as performance metrics.

Examples of metric selection. Bu et al. [35] built a specialized model for MapReduce applications. Because Mapreduce mainly consumes CPU resources and IO resources, the selected independent variable metrics are shown in Table 5. The authors also used a derived metric based on the task completion time.

Table 5. The Selected Metrics of the Model Given by Eq. (6) in reference [35]

	Parameters
System CPU	c_u: Local CPU usage in *Domu* c_a: Aggregated CPU usage of co-located VMS c_o: CPU usage in *Dom*0
System IO	r_u: Local read rate in *Domu* w_u: Local write rate in *Domu* r_a: Aggregated read rate of co-hosted VMs w_a: Aggregated write rate of co-hosted VMs io_u: IO utilization of physical server
Task	t_c: Average CPU demand t_r: Average read rate t_w: Average write rate

Koh et al. [5] built model for the application *analyzer*. The path to select the independent variable metrics is the second path. Because the application *analyzer* mainly consumes memory resources, the selection process should track the process of accessing memory. First, because the application *analyzer* mainly uses memory resources, the metrics related to memory reading and writing should be directly selected. Second, because memory read-write requests need to be sent to the CPU, so some CPU-related metrics should also be selected. Third, since memory access will only occur when the requested content can not be found in cache, the cache-related metrics should also be considered. Besides, because the state of TLB also has a significant impact on memory read and write, authors selected the performance counter 'blocks' as one of the independent variable. To sum up, Koh et al. [5] select 10 corresponding metrics as independent variables for the normalized performance model of the *analyzer* when co-located with the other load *B*. The selected metrics and coefficients are shown in Table 6.

3.2.2 Data collection. From the literature, we summarize that there are two ways to collect data for interference modeling: data collection based on interference injection and data collection based on historical records.

Just Accepted

Table 6. The Independent Metrics and Coefficients in [5]

X	Coefficient	X	Coefficient
cputime	6.60E-01	*Reads_issued*	5.70E-04
cachehits	-3.98E-01	*Time_reading*	-4.41E-05
cachemisses	-6.62E-10	*Writes_issued*	1.95E-05
vmswitches	-4.99E-04	*Time_reading*	-7.13E-06
novmswitches	-1.06E-03	*blocks*	8.19E-04
a_0	4.33E-01		

Interference injection. The basic idea of interference injection is to manually increase the pressure on specific resources in the host environment where the target VM or application under test is running. By controlling the strength S of the resource pressure, the actual performance of application A under different levels of interference can be obtained. This characterizes its sensitivity to interference. In this way one can also calculate the normalized performance of A given the resource pressure S [53]. The formula is shown in Eq. (12)

$$NS(A,S) = \frac{performance(A,S)}{performance(A,0)}. \tag{12}$$

where $performance(A,0)$ represents the performance of application A under no resource pressure condition while $performance(A,S)$ represents the performance of application A under resource pressure S. There are two ways to apply resource pressure. One can simply increase workload on the host machine, or simulate resource contention by manually controlling the resource capacity.

Two kinds of workload injection can be applied to increase resource contention. For the first one, the workload injected can accurately control its own usage of resources. Examples of such benchmarking workloads include iBench [54], CISBench [55] and Cuanta [56]. Because their occupancy of related resources changes regularly with time, the resource pressure on the load to be tested will also changes regularly over time. By doing so, the actual performance of the application to be tested under different pressures can be obtained. Another of the injected load is that the injected load cannot accurately control its own occupation of resources. Therefore, researchers obtain the actual performance of the load to be tested under different interference situation by running the load to be tested co-located respectively with a large number of different applications, such as Fecbench [47], and they also classified these applications into different categories and then summarizing the general rule of aggregated performance of the loads to be tested co-located with different kinds of tasks [9]. The injection load can not only exert pressure, but also feel the pressure exerted by the load to be tested, and then calculate the interference intensity and even resource utilization of the load to be tested [55, 57].

The second way to increase resource contention is to manually control the capacity of the specific resource without introducing additional workload. Experiments [58] proved that it is completely reasonable to simulate the phenomenon of "resource contention" by deliberately limiting the resource supply to the VM. This method does not need to run other loads and saves the resource cost. However, there are few existing studies resort to the second method.

In theory, interference injection method can obtain the performance data for any levels of performance interference, but it is difficult to obtain comprehensive performance interference data due to many factors. For example, because the operating system and advanced programming language shield the underlying hardware details, it is difficult to achieve desired interference accurately on specific resources.

Historical data. The data required for establishing the performance interference model can be extracted directly from the logs of a cloud system. Due to the dynamic nature of cloud environment, the performance of the running tasks and VMs will be more or less disturbed in a long time, and these data are recorded in the monitoring data. Researchers can use tools to collect historical data. For example, Masouros et al. [59] used PCM API [60] to collect underlying performance counters, and Buchaca et al. [61] used system calls to collect relevant data. Of course, researchers can also directly extract the data required by the model from the open dataset and by this way they does not need to run the VM, lead to saving resource consumption and time overhead.

Data collected from historical records of a production system can be more valuable than those acquired in test environment, but there are also some shortcomings. Firstly, the flexibility is poor. If the monitoring dimension needs to be updated, the historical data need to be accumulated again. Secondly, its application scope is only suitable for clusters with large resource

Just Accepted

contention, while for the cluster with stable operating environment, it is unable to obtain comprehensive performance interference data due to the lack of sufficient performance interference in the environment.

3.3 Modeling Methods

This section introduces the methods of establishing a model for detecting or assessing performance interference between VMs. Existing modeling methods presented in the literature typically fall into two categories: qualitative analysis modeling and quantitative analysis modeling. The output of qualitative analysis modeling is the level of performance interference or the interference-related characteristic for an application or VM. The output of quantitative analysis is a value which measures the degree of performance interference or the performance loss due to interference.

3.3.1 Qualitative analysis modeling methods. Most of the existing works used classification or clustering methods for qualitative analysis. Qualitative analysis can be applied to roughly assess performance interference (in levels) or identify the relevant characteristic of the application or VM. We summarize and compare these works in Table 7.

Table 7. Comparison of the Studies based on Qualitative Analysis

Ref.	Classification	Clustering	Method	Qualitative Results
[62]	✓	✓	SVM,K-means	Categorize the task as: Memory-intensive, CPU-intensive, Disk-intensive and Cache-intensive, and categorize the interference as: Absent, Low, Moderate, and High
[38]	✓		ANN	Categorize the application as: interactive or delay-insensitive
[63]	✓		static classification	Categorize the interference level as: Absent,Low,Moderate,andHigh
[5]		✓	hierarchical clustering	Classify the known applications that are similar to the interference characteristics of unknown applications
[64]		✓	hierarchical clustering	Group the applications with similar interference performance into clusters

Classification and clustering. Both classification and clustering offer the simplicity for modeling performance interference in the case when our primary concern is whether or why it happens. Classification-based methods requires supervised learning over labeled, which means one needs to train the detection model with prior knowledge about the presence of interference in the training samples. Clustering-based methods can group the applications by similarity in an unsupervised learning manner. The result of clustering offers insights on whether or why a VM is interfered. However, additional information (e.g, number of interference levels) is required with clustering so as to determine the presence of interference qualitatively.

Some examples in existing works. For classification methods, a variety of classification-based methods are explored for modeling performance interference. For example, on the characteristic types related to interference, Meyer et al. used SVM (Support Vector Machine) to classify applications that mainly consume the same resources into one category [62]. Shaw et al. used ANN (Artificial Neural Network) to divide workloads into interactive and delay-insensitive workloads [38]. In addition, the classification method can also be used to classify the interference level. For example, Ludwig et al. designed formulated static classification rules to divided the performance interference degree into four categories: Absent, Low, Moderate,and High [63]. The classification standard in this article was fixed. We think that researchers can change the static standard to the dynamic floating standard to adapt to the rapid change of the operating environment.

Clustering-based methods can be used to assess the intensity level of performance interference. For example, Meyer et al. used K-means to divide the interference data into different interference levels [62]. In addition, some works used this method to compare the similarity of interference characteristics between applications. For example, Koh et al. [64] used the hierarchical clustering method to compare the similarity between unknown applications and known applications, and then infer the interference performance of unknown applications based on the values of known applications [5]. And Li et al. used clustering method to divide the applications with similar interference performance into a set [64].

Just Accepted

Qualitative analysis can be combined with quantitative analysis for modeling. For example, Li et al. built quantitative model for each interference type of applications [64]. Koh et al. [5] take the distance of hierarchical clustering as the value of similarity to model the performance interference of unknown applications:

$$NS(U@Bn) = \sum_{N} wi \times NS(i). \tag{13}$$

where $NS(U@Bn)$ represents the interference of unknown application U suffering from co-located application Bn, N is the number of known applications similar to U, wi represents the ratio of similarity of application i to all similar applications and $NS(i)$ represents the degree of performance interference of known application i.

3.3.2 Quantitative analysis modeling methods. The purpose of building quantitative analysis models is to measure how strong the interference is. This is usually done using interference-related metrics. The output of a quantitative analysis model is the values quantifying the degree of performance interference or the performance loss due to interference. The process of modeling is to establish the functional relationship between the selected independent metrics and performance interference measurement metrics. We summarize some of the works mentioned in this subsection in Table 8.

Table 8. Comparison of Studies on Quantitative Analysis Modeling

Ref.	Method	Target Scenarios	Use of Models
[65]	statistical analysis	Network-intensive and CPU moderate	Quantify the overall performance degradation due to VM migration interference in the process of VM migration
[66]	statistical analysis	Network-intensive, CPU-intensive	Quantify the performance degradation due to interference of the co-location of CPU-intensive VMs or the co-location of network-intensive VMs
[67]	function regression	CPU-intensive, RAM-intensive, Disk-intensive	Establish the relationship between the number of co-located VMs and the degree of performance interference
[42]	function regression	CPU-intensive, IO-intensive	Quantify the performance degradation due to interference based on four metrics shown in Table 10
[48]	markov chain	Network-intensive, CPU-intensive, IO-intensive	Predict the performance degradation due to interference based on the probability of entering the next stage of access resources
[68]	neural network	Any type	Predict the impact of two co-located applications interference based on RNN
[59]	neural network	Any type	Predict the impact of two co-located applications interference based on LSTM

Modeling method based on statistical analysis. The methods of modeling based on statistical analysis refers to the method that study the correlation between independent variables and interference measures, such as proportional and inverse. This method is simple and easy to configure, but because the statistical data is usually only suitable for the server that meets the specific hardware configuration or only considers the specific running environment, the model or parameters obtained in many cases can nott have good generalization ability.

Anu et.al [65] studied the correlation between some resource metrics and performance interference measurement in the process of VM migration. Authors thought that the performance interference in the migration process is mainly caused by CPU resources and network resources. By analyzing the corresponding relationship between resource metrics and performance interference, the interference caused by CPU resources is given by

$$M_c = \frac{C_d}{C_l}, \tag{14}$$

where M_c represents the contention degree of CPU resources, C_d is the number of CPUs required by the VMs in the system, and C_l is the total number of CPUs that can be allocated. The above formula reflect the correlation between the two CPU-related metrics and the interference measurement. In the correlation, C_d is directly proportional to the interference and C_l is inversely proportional to the interference.

Wang et al. [66] studied the performance degradation due to interference of the co-location of CPU-intensive VMs and network-intensive VMs. For CPU intensive VMs, the higher the LLC miss rate, the higher the performance degradation. For

Just Accepted

network intensive VMs, the higher the LLC miss rate of other co-located VMs, the higher the performance degradation. The performance degradation formulas of the two VM types are given separately by

$$d_{vm_i} = \partial_{vm_i} \times R_{vm_i}, \tag{15}$$

$$P_{vm_i} = \sum_{j \neq i}^{n} p_{vm_i} = \sum_{j \neq i}^{n} \alpha_{vm_j} \times R_{vm_j}, \tag{16}$$

where d_{vm_i} represents the performance degradation due to interference of CPU-intensive VM, R_{vm_i} represents the LLC miss rates of the CPU-intensive VM, P_{vm_i} represents the the performance degradation due to interference of network-intensive VM, n represents the number of co-located VMs with the VM, R_{vm_j} represents the LLC miss rates of one of the co-located VMs with the network-intensive VM.

Modeling method based on function regression. A regression model can be univariate or multivariate. The essence of performance interference model is to describe the mapping relationship between one or more performance indicating variables and performance interference measurement. Therefore, regression analysis is one of the common methods to determine the parameters of performance interference model.

Jersak et al. [67] used regression method to establish the relationship between the number of co-located VMs and the degree of performance interference for the co-location operation of the same resource intensive loads. The equations for three kinds of resource intensive VMs are shown in Table 9, where y represents the degree of performance interference, and x represents the number of co-located VMs. This study only focuses on co-located interference under the same resource intensive, only focuses on the number of VMs and does not consider the diversity of load types on VMs although the types of the loads running on VMs of a physical host are different in general.

Table 9. The Models for Co-location Interference in [67]

Resource	Interference Model
CPU	$y = 6.5346 \ln(x) - 4.4983,$
RAM	$y = 34.398 \ln(x) - 4.7183,$
Disk	$y = 35.347 \ln(x) + 7.2785,$

Chiang et al. [42] selected four metrics as independent variables and established the functional relationship between these metrics and performance interference measurement after sufficient VM co-location experiments, which is given by

$$\hat{Y} = c + \sum_{i=1}^{4} \partial_i \times X_{VM_{1,i}} + \sum_{i=1}^{4} \beta_i \times X_{VM_{2,i}}, \tag{17}$$

where $X_{VM_{1,i}}$ are the values of the i-th parameter of VM VM_1, i values are 1 to 4, and the order is shown in Table 10. The representation of VM VM_2 is the same. If we combine $X_{VM_{1,i}}$ and $X_{VM_{2,i}}$ into the power formula, such as $(1+\sum_{i=1}^{4} X_{VM_{1,i}} + \sum_{i=1}^{4} X_{VM_{2,i}})^2$, primary non-linear formula can be built. Because the formula is complex, we do not list it here. After author's experiments, the test results show that the nonlinear model has the best performance. Besides this article, the idea of the article of Sun et al. [51] is very similar.

Table 10. The Selected Performance Metrics in [42]

CPU	IO
Local utilization in DomU	Read requests per second
Global utilization in Dom0	Write requests per second

Just Accepted

Modeling method based on Markov chains. The quantitative prediction of performance interference can also be probabilistic. Markov chain is a stochastic process following Markov property based on probability theory and is commonly used to model system behaviors which are linked to a certain space of states. The possibility of entering the next stage is calculated through probability. Chen et al. [48] inferred the actual time under interference condition taken by VM to access resources based on the probability of entering different stages when accessing resources. Votke et al. [69] selected the number of VMs on all servers, service rate under interference or non-interference state, request rate and other parameters to predict the number of requests suffered from performance interference executed in cloud data center.

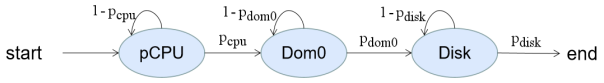


Fig. 4. The three states in the process of accessing disk

We take [48] as an example to illustrate the practice of this method in performance interference modeling. In this article, the model of VM performance was built by predicting the probability of entering the next state transition when accessing network, CPU and disk resource. For example, because accessing disk resources is consists of three states such as pCPU, Dom0 and Disk, authors designed a state transition diagram for these steps, which is shown in Fig 4. The ρ in the Fig 4 represents the probability of entering the next state. The first state *pCPU* represents the step that a job completes service at physical CPU to place the IO request in the shared memory area. The second state *Dom*0 represents the step that interrupts are sent to privileged VM Dom0. The third state *Disk* represents the step that the job obtain the disk response in the shared memory area. The probability of the state transition from pCPU to Dom0 is calculated by the metrics include the capacity of physical CPU, the capacity of virtual CPU and the CPU utilization due to relevant VM. The probability of the state transition from Dom0 to Disk is calculated by the metrics such as the utilization of disk channel. Authors proposed the equation of the predicted average time spent by accessing disk resource, which is given by

$$E(K_{disk}) = \frac{1}{pDom_0} + \frac{1}{p_{cpu}} + \frac{1}{p_{disk}}, \tag{18}$$

where $pDom_0$, p_{cpu} and p_{disk} respectively represent the probability of staying at the current stage which is calculated by 1 minus the probability of transition to the next stage.

Modeling method based on neural networks. Neural network, with a large number of neurons and dense connections between them, has proved very effective as a power tool to learn sophisticated patterns. In a cloud data center, the VM running environment is ever-changing. Thus, it is difficult to establish precise relationship between performance interference measurement and corresponding metrics with simple models. Precise models should be complex and nonlinear and the neural network has the ability to learn and build the model of nonlinear complex relationship. The advantages of neural networks are not limited to this. After learning from the initial input and its relationship, it can also infer the unknown relationship from the unknown dates, which makes the neural network has a wide application space in the complex problem of VM performance interference [59, 61, 68].

Buchaca et al. [68] proposed a workload-workload prediction method using a sequence-sequence model based on recurrent neural networks (RNNs) to predict the impact of two co-located applications interference. The model uses two gating loop units (GRU) as building blocks; one GRU processes the incoming trace signals of the two applications and passes the processed information to the other GRU, which outputs the predicted resources usage with time dimension of the two applications when co-located with each other. In addition to the work, Masouros et al. [59] established a similar pairwise co-located performance prediction model based on LSTM and David et al. [61] did it based on RNN. Although the above works can only predict the performance of two co-located applications, the proposed models consider the time dimension and more finely correlate the relationship between the corresponding metrics and performance interference measures.

3.4 Comparison of Studies on Modeling

After reviewing the relevant works (listed in Table 2, Table 7 and Table 8), we have the following conclusions:

(1) In the process of experiment, most of the studies focus on the problem of mutual interference between two VMs or tasks. There are few in-depth analysis on the problem of multiple VMs or tasks interfering with each other.

Just Accepted

(2) Most of the existing methods use the task execution time for measuring performance interference under CPU intensive workloads. For network IO intensive and disk IO intensive applications, more metrics are considered in addition to task execution time, including the throughput rate of the VM.
(3) Incremental training of the interference models are of great importance as we may see constant environmental changes in the cloud data center. However, techniques like reinforcement learning are rarely explored in modeling performance interference.
(4) In terms of the objectives, responses of different applications to mutual interference and their interference characteristics are hardly explored.

4 INTERFERENCE-AWARE VM SCHEDULING OPTIMIZATION

The VM scheduler in a virtualized cluster is responsible for both the placement of new VM instances and the consolidation of running instances through VM migration [70] or killing. With prevalent deployment of virtualized services, much effort has been paid to the development of interference-aware optimization methods. These VM scheduling optimization methods include the optimization of the new task placement strategy and the optimization of adjusting the current VM-PM mapping in the data center. Some optimization strategies proposed in the literature have multiple optimization objectives beside interference reduction. For example, they can be designed to be interference-aware while being effective in reducing energy consumption. We define these multi-objective optimization strategies as interference-aware co-optimization strategies.

4.1 Strategies for Interference-aware VM Scheduling

The majority of studies with the aim of avoiding or mitigating VM performance interference focusing on the optimization of VM placement and consolidation. The placement of VMs is combined with the design of queues and task priority. Recent studies also explore the possibility to perform optimized, interference-aware VM scheduling without prior knowledge about the applications or the environment. Besides, some studies investigated how to reduce the side-effects of optimization such as the performance fluctuation caused by VM consolidation in the process of migration.

4.1.1 Interference-aware VM Placement Optimization. Interference-aware placement strategies optimize the placement rules to avoid the potential impact of performance interference on the new VMs (and the associated tasks) in the initial stage of scheduling. The architecture of the strategy is shown in Fig. 5. Some works adopt task priority design to adjust the scheduling

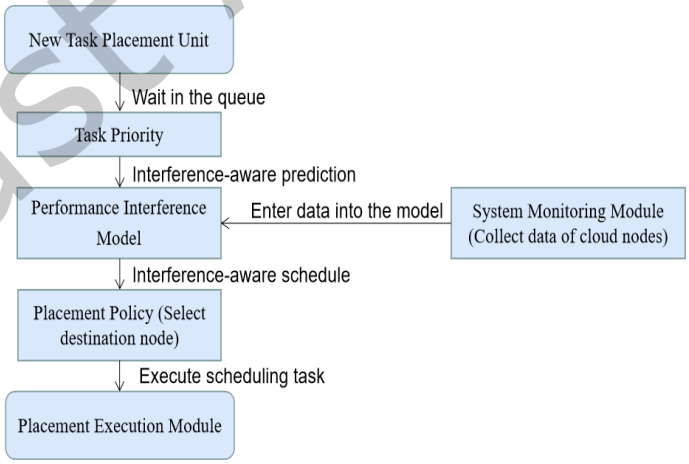


Fig. 5. The framework of placement strategy

order. The module "placement strategy" in Fig. 5 is responsible for the search of candidate nodes and the decision of the node that the task schedule to. In this module, the interference-related data of the task to be scheduled will be fed into the model to predict the performance interference after scheduled to the selected candidate nodes and then the task will be scheduled to one node according to the requirement of interference optimization. In addition, in recent years, there are more and more

Just Accepted

optimization schemes without prior knowledge. These schemes do not require any data to be given in advance, but obtain interference information and make decisions automatically.

The design of task priority. The design of task priority allows for more flexible control over the VMs waiting to be scheduled in some specific scenarios. For example, the priorities can be determined according to the interference characteristic or preferred operation guarantee of different kinds of tasks. We compare the design of the task priority in the literature in Table 11.

Table 11. Comparison of the Task Priority Design for Interference-aware VM Placement

Ref.	Higher priority	Lower Priority
[71]	CPU-intensive applications	Interactive applications
[72]	Network-intensive applications	CPU-intensive applications
[38]	Delayed-insensitive tasks	Interactive tasks
[33]	The descending order of the values of interference intensity	The ascending order of the values of interference sensitivity

For example, in order to guarantee the response time of interactive applications, the CPU-intensive applications which occupy CPU for a long time should be classified into ultra-low priority unless the applications are at risk of delay [71]. Similarly, Sampaio et al. [72] designed similar priority rules for CPU-intensive and network-intensive applications because they thought that guaranteeing the operation of network-intensive tasks is more important. Besides, in the work of Shaw et al. [38], the scheduling strategy first scheduled delayed-insensitive tasks, and then arranged the scheduling of interactive tasks according to the resource pressure of all nodes after delay-insensitive tasks have been scheduled, so as to prevent interactive tasks from performance degradation due to interference. In addition, Kim et al. [33] designed two priorities for tasks based on the magnitude of interference intensity and interference sensitivity separately. Based on the two priorities, the algorithm schedules a part of tasks one by one firstly according to the descending order of the values of interference intensity, and then allocates other tasks based on the ascending order of the values of interference sensitivity so as to avoid the mutual interference of several tasks that are not suitable for co-location operation.

Placement Policy. A diversity of VM placement strategies can be found in the literature. Most of them seek to avoid or alleviate performance interference by reduce source contention. They differ in terms of the adoption of inference metric, goal of placement, and the placement rules. We summarize them in Table 12.

In simple terms, a placement policy chooses the best fit for a VM among several candidate nodes. Some nodes are excluded from the set of candidate nodes at the beginning of placement scheduling. For example, Alves et al. [73] and Lin et al. [75] limited the range of candidate nodes with the constraints of integer linear programming and filtered out the nodes that do not satisfy the threshold requirements of resources capacity. In the node search, most existing works used non-global optimization methods such as heuristic algorithms [35, 42, 67, 72, 73] and meta- heuristic algorithms [74]. In the node search phase, the performance interference of the tasks or VMs after scheduled to candidate nodes were predicted and recorded. Generally speaking, in the prediction process, only the the relevant metrics of the task to be scheduled and the candidate node need to be input into the model to obtain the prediction results. However, some works proposed more fine-grained methods, in which the prediction results of surrounding environmental factors were also taken into account. For example, Reza et al. [76] considered the prediction results of the incoming rate of events in a future time interval and Swain et al. [77] considered the overlapping time of the task to be scheduled and the co-located tasks. Considering the surrounding environmental factors will increase the accuracy of performance interference prediction and pave the way for accurate and fine-grained interference-aware scheduling.

The choice of the node that the task schedule to depends on the objectives of placement optimization. The objectives can be divided into two types. One is to mitigate interference for the scheduled task and another is to ensure that of the overall system. For the first objective, some works compared the predicted interference in the process of node search, and selected the node with the optimal performance interference if the task was scheduled to [71, 78]. Besides, some works did not pursue the optimal interference, as long as the interference does not exceed the threshold [35, 67, 79] or the task does not exceed the

Just Accepted

Table 12. Comparison of Interference-aware VM Placement Strategies

Ref	Resources Considered	Interference Metric		Placement Goal	Placement Rule
		Single VM or Task	Group-level model		
[42]	Disk IO, CPU	✓		Schedule the incoming tasks to different VMs in a way that minimizes the interference effects from co-located applications	Heuristic (Minimum interference online scheduler, Minimum Interference Batch Scheduler, Minimum Interference mixed scheduler)
[67]	CPU, RAM, Disk IO	✓		The interference of tasks on the nodes is lower than the threshold and meets the resource requirements	Heuristic (The First-fit Decreasing, The Best-fit Decreasing, The Worst-fit Decreasing)
[72]	CPU, Network	✓		The task performance and resource capacity meet the requirements	Heuristic
[73]	CPU, Memory		✓	Optimize task completion time and save active PM	Integer programming, Meta-Heuristic (Iterative Local Search)
[74]	CPU, RAM, Disk IO		✓	Reduce the number of active PMs, improve resource utilization, and optimize the completion time	Integer programming,Meta-Heuristic (Gray Wolf algorithm)

deadline [41]. For the second objective, the algorithm should take into account the overall group-level interference of all related cloud entities. Some works combined the performance interference models of each single entities such as VMs or tasks to form an overall performance interference model [73–75] and some works directly used the overall model built in the modeling stage such as the work of R. Shaw et al. [38].

Prior knowledge-free optimization schemes. The cloud environment is complex and dynamic, which complicates the decision-making of VM placement strategy. As a result, the effectiveness of optimization strategies are sensitive to the changes of the cloud environment. In recent years, researchers have been exploring a new approach called prior knowledge-free optimization. For example, Rameshan et al. [80] proposed a scheme called *stay-away*, which continuously learns the state-space representation to distinguish the states of execution for multiple co-located VMs. Based on the timely updated representation, the system can mitigate the adverse impact of performance interference on sensitive applications when coexisting with other batch applications.

As an experience-driven learning paradigm, reinforcement learning does not require sufficient data provided in advance. The decision maker (agent) obtains learning information by receiving reward (feedback) from the environment and updates its policy accordingly. Deep learning offers strong perception ability but hardly provides end-to-end decision-making ability, while reinforcement learning can make up for it. Therefore, the combination of the two provides a solution to the perception decision-making problem of complex systems. This emerging approach is called deep reinforcement learning (DRL). Enabled by the characteristics of DRL, recent studies seek to integrate interference-related factors with the reward mechanism in the design of prior knowledge-free optimization schemes [81, 82].

4.1.2 Interference-aware VM Consolidation Optimization. Due to the dynamic environment of cloud data centers, the initial host of a VM instance may not be the optimal one for the task after a period of execution. Consolidation strategies are designed to dynamically adjust the locations of VMs. Interference-aware VM consolidation optimization aims to adjust the current VM-PM mapping to avoid or lower the impact of performance interference in the system. The architecture of consolidation strategy is shown in Fig. 6. The performance interference monitoring system is responsible to estimate whether there is a trend of interference in VMs by monitoring the resource pressure in the cloud system and observing the running situation of VMs. When there is a trend of interference in VMs, it delivers warning to consolidation strategy. In term of the

Just Accepted

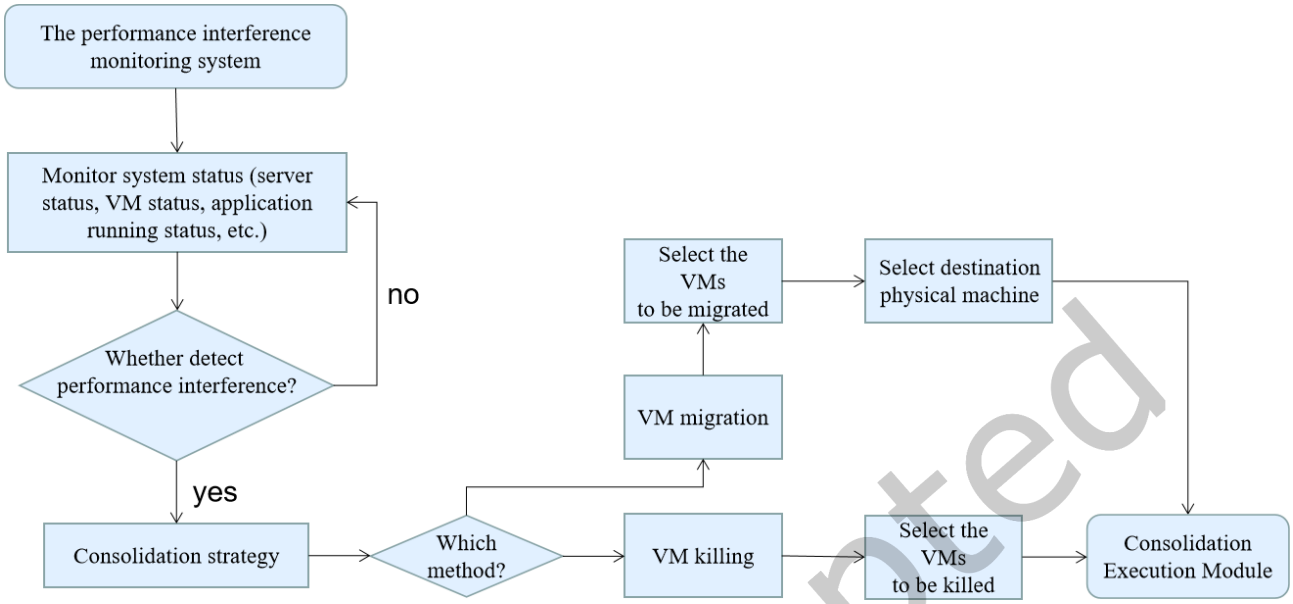


Fig. 6. The framework of VM consolidation strategy

consolidation strategy, it can alleviate the resource contention and optimize the VM operation by online migration or killing VM according to the interference-aware resource prediction model or interference-aware application performance prediction model. In the process of consolidation, we have to pay attention to the interference of VM migration process because the changes of the running environment can seriously affect the source server and destination server. In the following, we first introduce the performance interference monitoring system that decides if consolidation is necessary, then we discuss state-of-the-art consolidation strategies in the literature. We summarize the relevant works in Table 13.

Table 13. Comparison of the Studies on Interference-aware VM Consolidation

Ref.	Criteria	Operations	Strategy
[22]	Multiple metrics of VM significantly deviate from the normal value	Online migration	If only little VMs are affected, the affected VMs are migrated; if most VMs are affected, the interference source VMs are migrated
[66]	Normalized performance exceeds threshold	Online migration	Same as Ref. [22]
[83]	The LLC miss rates of VMs and PMs is very high	Online migration	Migrate a VM with a high LLC miss rate to a PM with a low LLC miss rate
[84]	The running VM obviously affects the operation of other VMs	Turn off the VM	Suspend the VM that causes high resource pressure
[45]	CPI metric fluctuates abnormally	Turn off the VM	Turn off the source VM that causes performance fluctuations
[85]	The state needs to be changed judged by Q-learning	Online migration	QoS reward in multi-agents Q-learing

The performance interference monitoring system. The performance interference monitoring system is responsible for activating the consolidation strategy. The activation condition determines the interference tolerance of the system and decides if certain operations should be triggered to alleviate the interference. A threshold is usually used as the criterion to decide whether to trigger the consolidation strategy. Some works set the threshold value based on the metric of task normalized performance which are shown in Eq. (4) and Eq. (5). If the threshold is exceeded, it will be considered that the performance degradation caused by interference is unacceptable, and then trigger the implementation of consolidation

Just Accepted

strategy [45, 86]. As described in section 3.1.2, performance evaluation criteria include application-level metrics such as task execution time and request response time, as well as PM-level and VM-level metrics such as CPI and MIPs. In addition to the criteria about performance,some works set the threshold value based on the metric of resource contention. For example, Chen et al. established a threshold equation based on the frequency of split locks to estimate whether the contention of memory bus lock is excessive [26]. Although Chen et al. studies resource allocation in that article, research can also be done on VM scheduling based on this detection method. Besides the single metric threshold, multiple metrics also can be applied to estimate whether the VM operation is abnormal by judging whether the selected multiple metrics of VMs seriously deviate from the normal values range by clustering or outlier detection [22, 23]. Some monitoring mechanisms not only estimate whether the running state of the task and VM is abnormal, but also identify the main interference source causing the interference [22, 48], which provides a more fine-grained reference for consolidation strategy.

Consolidation strategy. Existing studies generally implement VM migration strategies based on application performance and physical resource contention intensity. For example, Novakovic et al. [22] predicted the performance interference between VMs based on the underlying monitoring metrics. They identified the active VM that causes performance interference and then migrate it to a new host server. Ahn et al. [83] proposed to adjust the server-VM mapping based on the LLC miss rate of all servers and VMs to ease the memory access pressure. However, this method has a disadvantage. It may cause repeated migration of some VMs which always take up more resources than co-located VMs and frequent migration may cause performance jittering. In order to optimize this phenomenon, the number of migrated VMs and the times of migration should be taken into account when deciding the migration strategy. For example, in [66] the authors implemented different consolidation strategies based on the number of interfered VMs and that of the VMs in normal state. They pointed out that when the number of interfered VMs on a server is small, it is suggested to migrate the VMs being interfered. Otherwise, the interfering VMs should be migrated. The operation of VMs is not limited to the VM migration, and turning off the VMs is also an operation method [15]. For example, Salimi et al. [84] suspended the VMs that cause a significant increase in the resource pressure to the other VMs on the server. These VMs will be resumed when these will no longer cause high interference to the co-located VMs.

Similar to the development of placement strategies, researchers are exploring the application of reinforcement learning in the design of consolidation strategies. For example, based on Q-learing algorithm, Nishtala et al. [85] added a penalty mechanism for violating QoS into the reward mechanism to adjust the allocation state which is seriously affected by interference and avoid new interference caused by new allocation scheme.

Mitigate performance fluctuations in VM migration. The process of VM migration leads to changes in the system environment. In the process of migration, the additional consumption of CPU, network and memory resources can be significant. Hence, the threat to the stability of the system and the cost of migration should never be overlooked. In order to reduce the interference effect of resource pressure on all participants in the migration process, it is necessary to build a measurement model for the resource pressure or interference degree of participants. For example, Xu et al. [37] regarded all participants in the migration process — source PM, destination PM, and all VMs on related PMs as an integrated whole modeling object. The interference in this part was defined as migration interference. After the migration process is completed, the destination PM, the migrated VM and the VMs on the destination PM were regarded as another whole object and the interference in this part was defined as co-located interference. The interference effect of the two parts was measured by a fixed number. The lower the fixed number value is, the less impact the corresponding migration scheme has on the performance of the participants. In addition, Anu et al. [65] also modeled the migration process, but the selected metrics are all PM-level metrics not VM-level metrics compared with the previous paper. The above two mentioned models regard the migration process as a whole, and the evaluation of the migration strategy is only a fixed number, so we think these two models are as coarse-grained models. We expect that the more fine-grained model of online migration process will be proposed which considers the impact on different participants in different stages of the migration, rather than just a final number for all participants in the whole process. Due to the distribution of cloud data centers, the consideration of the cost of migration process is not only limited to resource contention, but also the data transmission time. Besides interference awareness, with topology awareness, the cost of migration can be better measured to formulate a more friendly migration strategy [87].

4.1.3 Comparison of the Studies on Interference Optimization. Through reviewing the existing studies on interference-aware scheduling optimization (summarized in Table 11-13), we have the following conclusions:

Just Accepted

(1) The research on interference-aware placement optimization is more extensive than that on interference-aware consolidation strategy. This may come down to the fact that VM placement is more fundamental. As suggested in [86], placement strategy should be enabled whenever new tasks arrive, and consolidation strategy should be enabled in daily management.
(2) In the design of interference-aware placement strategy, heuristic algorithms are commonly favored. This is mainly attributed to their interpretability and low overhead. Heuristic algorithms can be enhanced by applying proper queue priority based on the characteristics of tasks.
(3) In the design of interference-aware consolidation strategy, the majority of existing approaches [45, 66, 84, 86] are based on the performance metrics and current resource contention status. Besides, to minimize the impact on the system operating environment, the number of VMs to be migrated during each consolidation cycle should be limited.

4.2 Interference-aware Co-optimization via VM Scheduling

The mitigation of VM interference can come along with other optimization objectives. A number of existing VM placement and consolidation strategies seek to achieve load balancing [88, 89], save energy [90], increase resource utilization, reduce physical resource waste [91], or improve economic benefits [92]. Under many circumstances, we can reduce performance interference while improving other aspects of the cloud system by means of scheduling. Among these objectives, we found in the literature that it is very common to consider energy consumption and network topology in the design of interference-aware scheduling solutions. On the one hand, energy efficiency of data centers has become a broad concern. Many works set both energy consumption optimization and interference optimization as optimization objectives and developed VM scheduling strategies that are both interference-aware and energy-aware. On the other hand, though interference-aware optimization can alleviate the shared resources contention such as network resource in the data center, it is not enough to optimize the performance because the cost of data transmission is also closely associated to the topology. Thus, it is necessary to explore interference-aware and topology-aware VM scheduling optimization.

4.2.1 Co-optimization of interference and energy consumption. With the arising concern about the excessive energy consumption of data centers, energy efficiency has become an important factor in server management [93]. An interference-aware optimization strategy typically aims to optimize performance and avoid violating SLA, whilst an energy-aware optimization strategy typically seeks to reduce overall energy costs. It is natural to combine them when both objectives matter.

Interference-aware and energy-aware VM scheduling strategy can be regarded as multi-objective optimization. Some researches pay more attention to energy saving. For example, Nishtala et al. [85] use Q-learning reward mechanism to achieve optimal task management. Energy consumption-related factors accounts for most part of the reward such as load level, the number of cores, and DVFS. Only one metric related to performance inference, which is positive if QoS is reached, is considered in their work. Some works assign same importance to both and there are two methods to achieve that in the existing literature. One scheme of the first method is to combine the energy consumption model and performance disturbance model to form a new quantitative equation, and then use heuristic algorithm to select node for scheduling VMs based on the new equation [94]. The other one scheme of the first method is to use integer programming method to set up objective function of the weighted sum of the two models and then schedule VMs according to the mapping of the optimal solution [95, 96]. The second method is based on the research results of energy consumption problem to impose constraints on interference-aware scheduling optimization. For example, research shows that an idle server represents 60%-70% of the power consumed when it is fully utilized. Therefore, they suggest switch idle PMs to sleep mode. In addition, restrictions should also be applied to CPU power consumption to maximize the ratio of the amount of work performed to the consumed energy [72].

4.2.2 Co-optimization of interference and topology. In cloud data centers, the link distance between nodes and the network traffic are both important factors affecting the network transmission efficiency. To achieve optimized performance for network-intensive application and VM migration process, it is not enough to reduce the contention for network bandwidth. The uncertainty of network environment between nodes also brings uncertainty to task performance no less than the performance interference caused by resource contention. The number of some CPU and network IO-intensive hybrid operations, such as MapReduce, exist in cloud data centers are abundant. Therefore, it is significant to carry out the research on interference and topology awareness in this scenario.

There are two ways to optimize the cost of link. One is to assign the transmission cost between different nodes as the penalty value of scheduling, The objective function of scheduling optimization should not only include the performance of

Just Accepted

tasks, but also the penalty of data transmission in different nodes, so that the VM scheduling will consider the link cost [87]. Another method is similar to that of the first method, but the idea of this method is to increase the data locality, so that the nodes that transmit data intensively to each other are limited to a small subnetwork . This method sets different levels of locality such as VM-local, server-local, rack-local. The locality determines the network data transmission distance. If the resource conditions permit, the locality with the smallest network transmission range is preferred, and the VM is allocated to the nodes within the corresponding range [35].

5 OPEN ISSUES AND FUTURE DIRECTIONS

By reviewing existing approaches, we argue that there are still open challenges for the research on VM performance interference detection, prediction and optimization. In this section we discuss what can be further explored in the area. These topics are related to the improved interference modeling, interference-aware optimization for specific parties and applications, and extending the study to cover containerized infrastructure and more hardware-related factors. A summary is provided in Table 14.

5.1 Accurate and Fine-grained Interference Modeling

Existing approaches of interference modeling mostly resort to simple mathematical forms (e.g., linear models) and more importantly, overlook the interference between VMs that exhibit relatively different patterns of resource usage. In fact, VMs and applications have very complex characteristics in terms of causing or being affected by interference. For example, memory-intensive application may in some case compete intensively for CPU time slices, which makes it possible to interfere with CPU-intensive workload. Numerically, the selected load or performance metrics may not show a clear functional relation with the intensity of performance interference. Moreover, in different execution stages, the resource requirements of the VM and other co-located VMs can vary significantly. As a result, the interference degree of the same application running on the same VM at different times may be different and the single model cannot accurately show this relationship. A promising solution is to use deep neural network for models and reinforcement learning for decision-making.

In terms of granularity, most performance interference models assume a static interference pattern throughout the VM lifecycle or task execution phases. Most of the existing methods simplify the interference prediction problem into a regression problem. The model produces a single value global prediction estimate instead of a series of time-varying prediction values. In the implementation phase, many studies regard different task execution stages as a single process, ignoring the difference of resource requirements at different execution stages. The fine-grained performance interference models can provide more detail information for performance interference optimization, and is conducive to the development of a more refined optimization method, which promote the reasonable use of debris resources during operation so as to reduce resource contention and performance interference. For example, Wang et al [97] focused on the modeling of the first stage, that is, the stage of reading data from the storage device. Their method is dedicated to avoiding the interference between co-located applications in this specific stage.

5.2 Performance Interference Optimization for Different Parties

For cloud service providers, profit is the priority. From this perspective, how to minimize performance interference without compromising the profit of service is non-trivial. For example, Lin et al [75] used the linear programming method to model the profit maximization problem considering multiple interference factors. Service providers can also arrange customized solutions or service quality for users according to their identities, so as to save resources and maximize benefits through differentiated services.

Performance interference-aware optimization can also benefit the customers. For cloud service users, they may also have demands for performance interference detection and prediction. For example, users need to know whether the quality of service they get satisfies their requirements. But from their point of view, many load characteristics are not easy or impossible to obtain because some data is not visible to the user or needs to be obtained from the service provider and other co-located applications users. Therefore, it is necessary to study the selection of load indicators that are visible and easily accessible to users for research. Generally, application-level load indicators, such as transaction response time and some VM-level indicators, are more appropriate for selection, because users are the main trigger of VM-level and application-level behavior changes and it's easy to obtain for users. For a instance, Amannejad et al. [98] based on the application transaction response time of web services on the cloud platform detects whether the application transaction has performance interference

Just Accepted

phenomenon. This detection model still has room for improvement, and more application-level load indicators can be added, such as transaction type, transaction load increment, IO throughput, etc. In a word, the optional load characteristics of user identity-oriented detection and measurement models are hardly explored.

5.3 Mitigating Interference for Real-time Applications

Some applications featuring real-time interactions are particularly sensitive to delay. Shortening response time is the priority of some hard real-time applications such as cloud gaming. In this case, we need prompt prediction and scheduling to prevent serious degradation in the application performance and the user experience [99]. The detection, prediction and scheduling mechanisms for real-time applications cannot be offline, and the time complexity of real-time scheduling algorithm must be low to ensure the timely execution of tasks. The cloud computing environment is dynamic, so the real-time detection, prediction and scheduling mechanism need to consider time-varying and unexpected situations, and the accuracy must be high, otherwise it will lead to continuous VM migration. From the above, we observe that it is particularly important to mitigate interference in real-time applications, but how to realize it is a very challenging and practical topic.

There are two possible ways to solve this problem. The first is to improve the design of scheduling algorithms. For example, Caglar et al. [100] proposed an interference-aware scheduling algorithm for soft real-time applications on cloud platform. Machine learning method is used to learn the best matching mode of these classified VMs. At the same time, the forward propagation and back propagation of neural network are used to optimize the previous steps. The second way is to mitigate interference from the perspective of hardware. To realize real-time prediction and scheduling on a cloud platform, one has to take into account the restriction on hardware access by real-time client software as well as the communication delay due to the geographic distribution of nodes [101].

5.4 Container-Oriented Interference Detection and Prediction

Cloud-native is a kind of new cloud technology product system, which is the future development direction of cloud computing. After using cloud native technology, developers do not need to consider the underlying technology implementation, and can make full ues of the flexibility and distributed advantages of the cloud platform to realize rapid deployment, on-demand scaling, non-stop delivery, etc. Container represents the emerging light-weight virtualization technology which is also called the next-generation virtualization technology. Containers encapsulate a relatively independent environment that is similar to but less isolated than that provided by virtual machines [102]. The main difference between them is the location of virtualization layer and the usage of operating system resources. Compared with VM, the architecture of container can reduce hardware cost, deploy development/test/production environment more quickly, maintain development/test/production environment more easily, and are more compatible with microservice architectures. In view of these advantages of container, it is expected to be more widely used than VMs in the future. Therefore, the performance interference analysis of co-located container load and the model building of performance interference detection and prediction for container are both worthwhile research directions. For example, Chen et al [103] used direct observation method to analyze the co-located interference of different container loads at the micro architecture level metrics (such as hardware counter, read-write rate and running time under different co-located environments). It also provided a list of recommended co-located loads and non-recommended co-located loads for different types of container applications. The idea is also adopted in [53]. It is expected that increasing attention will be to paid to the research on performance interference in containerized systems in the future.

5.5 Further Investigation in VM Performance Unpredictability

At present, the research on performance unpredictability of VMs in cloud data center mainly focuses on the performance interference caused by resource contention. This is the main factor accounting for the unpredictable performance of cloud data center. Nonetheless, we argue that there are still other factors, such as hardware heterogeneity of PMs in CDC and communication interaction between multiple modules of different task types, which can also contribute significantly to the interference between VMs.

Hardware resource heterogeneity is a common property of large-scale cloud data centers. It has been reported that heterogeneous hardware resources of physical host machines can cause VM performance fluctuations by up to 60% [104] for large virtual machine instances and 280% [105] for small virtual machine instances. In order to compensate for the performance uncertainty caused by heterogeneous hardware, it is necessary to compare and analyze the performance of different hardware types, and build a scientific performance interference model. In the scheduling phase, we can allocate tasks

Just Accepted

to the PM corresponding to the hardware type that can make it finish the fastest according to the hardware heterogeneous status [106]. In some situations, VMs have special mapping or isolation requirements for hardware types and we can schedule VMs according to different requirements of the same hardware type [107].

In data centers at scale, the cost of communication interaction should also be considered as the impact can be no smaller than the performance interference caused by resource contention. For example, some user-oriented and delay-sensitive cloud applications have multiple interaction modules and complex interaction patterns. These interaction behaviors and resource contention among VMs jointly make the task performance prediction agnostic [63]. For these VMs, the benefits of two VMs running on two servers separately are not necessarily more than the benefits of the co-location operation on one server because the communication interaction gain of co-location operation on a server is greater than the performance interference loss of that. Kejiang Ye et al. [108] proposed a scheduling method that comprehensively considers performance interference and communication interaction characteristic. However, the method in this article needs to traverse all possible placement situations, resulting in high time complexity. In order to remedy this shortcoming, we can model the impact of different interaction modes on the performance of VM and then replace exhaustive search with the a heuristic strategy based on the output of the model to improve the practicability of the scheme.

Table 14. A Summary of Future Directions

Future directions	Perspectives	Possible solutions
Accurate and fine-grained interference modeling	Model accuracy	Explore reinforcement learning and neural networks for model design
	Fine-grained modeling	Study time-series modeling and modeling for different operation stages
Performance interference optimization for different parties	For cloud service providers	Explore the connection between VM performance interference and profiting
	For cloud service users	Enable the users to be interference-aware with differentiated quality of service
Mitigating interference for real-time applications	Software-level optimization	Build interference model for real-time applications and design fast and accurate real-time scheduling algorithm
	Hardware-level optimization	Explore the optimization of hardware resource access mechanism
Container-oriented interference detection and prediction	Containerized systems	Model the performance interference characteristics of containers
Further investigation in VM performance unpredictability	Hardware resource heterogeneity	Build performance interference models that can adapt to heterogeneous hardware
	Communication and interaction	Explore how to balance the gain in interaction and the performance loss due to co-location interference

Just Accepted

6 SUMMARY AND CONCLUSION

The interference between VMs is one of the main causes of performance degradation and also complicates the management of virtualized data centers. The problem has a strong impact on the performance of the entire system as well as the cost of operation. In this survey, we first summarize the causes of VM performance interference. Then we provide a comprehensive review of the methodologies in modeling performance interference between co-located VMs and categorize them according to their modeling objectives, adopted metrics and modeling methods. Then we review a wide range of existing interference-aware VM scheduling schemes in two parts, including the interference-aware optimization through VM placement and consolidation and co-optimization schemes that take multiple objectives into account.

Through analyzing existing interference-related models, we found that most of the studies are limited to the problem of mutual interference between a pair of VMs. Also, incremental training of the interference models is of great importance in terms of adapting to environmental changes, but it is hardly explored. By reviewing studies on interference-aware scheduling optimization, we observe that the research on placement optimization is more extensive than that on VM consolidation

strategy. Particularly, heuristic algorithms is a very popular option when it comes to the design of placement strategy. It is expected that an increasing number of new approaches will adopt deep learning and reinforcement learning methods for interference-related modeling and decision-making. Meanwhile, it is also critical to ensure that the detection and scheduling operation are light enough for real-time applications. Another open challenge comes from the fact that interference is not the only factor that leads to unpredictable performance. This motivates us to investigate the association between VM interference and the data center infrastructure such as hardware heterogeneity and the interaction of applications. This research work will help researchers find the important characteristics of VM interference and select the most suitable techniques to address the VM interference issues with the specific requirement. Along with a summary of these open issues, we also provide insights for the possible solutions accordingly, aiming to inspire further study on the performance interference in cloud data centers.

ACKNOWLEDGMENTS

This work is supported by National Natural Science Foundation of China (62072187, 62102408), Guangdong Major Project of Basic and Applied Basic Research (2019B030302002), the Major Key Project of PCL (PCL2021A09), and Guangzhou Development Zone Science and Technology Project (2020GH10, 2021GH10).

Just Accepted

REFERENCES

[1] Laurence Goasduff. 2021. Gartner Says Cloud Will Be the Centerpiece of New Digital Experiences. https://www.gartner.com/en/newsroom/press-releases/2021-11-10-gartner-says-cloud-will-be-the-centerpiece-of-new-digital-experiences. (2021).

[2] Susan Moore. 2022. Gartner Says More Than Half of Enterprise IT Spending in Key Market Segments Will Shift to the Cloud by 2025. https://www.gartner.com/en/newsroom/press-releases/2022-02-09-gartner-says-more-than-half-of-enterprise-it-spending. (2022).

[3] Paul Barham, Boris Dragovic, Keir Fraser, Steven Hand, Tim Harris, Alex Ho, Rolf Neugebauer, Ian Pratt, and Andrew Warfield. 2003. Xen and the art of virtualization. *ACM SIGOPS Operating Systems Review* 37, 5 (2003), 164–177.

[4] Christopher Clark, Keir Fraser, Steven Hand, Jacob Gorm Hansen, Eric Jul, Christian Limpach, Ian Pratt, and Andrew Warfield. 2005. Live migration of virtual machines. In *Proceedings of the 2nd conference on Symposium on Networked Systems Design & Implementation-Volume 2*. 273–286.

[5] Younggyun Koh, Rob Knauerhase, Paul Brett, Mic Bowman, Zhihua Wen, and Calton Pu. 2007. An analysis of performance interference effects in virtual environments. In *2007 IEEE International Symposium on Performance Analysis of Systems & Software*. IEEE, 200–209.

[6] Sean Kenneth Barker and Prashant Shenoy. 2010. Empirical evaluation of latency-sensitive application performance in the cloud. In *Proceedings of the first annual ACM SIGMM conference on Multimedia systems*. 35–46.

[7] Xing Pu, Ling Liu, Yiduo Mei, Sankaran Sivathanu, Younggyun Koh, Calton Pu, and Yuanda Cao. 2012. Who is your neighbor: Net i/o performance interference in virtualized clouds. *IEEE Transactions on Services Computing* 6, 3 (2012), 314–329.

[8] Yi Yuan, Haiyang Wang, Dan Wang, and Jiangchuan Liu. 2013. On interference-aware provisioning for cloud-based big data processing. In *2013 IEEE/ACM 21st International Symposium on Quality of Service (IWQoS)*. IEEE, 1–6.

[9] Renyu Yang, Ismael Solis Moreno, Jie Xu, and Tianyu Wo. 2013. An analysis of performance interference effects on energy-efficiency of virtualized cloud environments. In *2013 IEEE 5th International Conference on Cloud Computing Technology and Science*, Vol. 1. IEEE, 112–119.

[10] Sa.Wang, W.Zhang, H.Heng, Y.Song, J.Wei, H.Zhong, and T.Huang. 2015. Approach of Quantifying Virtual Machine Performance Interference Based on Hardware Performance Counter(In Chinese). *Journal of Software* 26, 8 (2015), 2074–2090.

[11] Ram Srivatsa Kannan, Animesh Jain, Michael A Laurenzano, Lingjia Tang, and Jason Mars. 2018. Proctor: Detecting and investigating interference in shared datacenters. In *2018 IEEE International Symposium on Performance Analysis of Systems and Software (ISPASS)*. IEEE, 76–86.

[12] Melanie Kambadur, Tipp Moseley, Rick Hank, and Martha A Kim. 2012. Measuring interference between live datacenter applications. In *SC'12: Proceedings of the International Conference on High Performance Computing, Networking, Storage and Analysis*. IEEE, 1–12.

[13] Ilia Pietri and Rizos Sakellariou. 2016. Mapping Virtual Machines onto Physical Machines in Cloud Computing: A Survey. *ACM COMPUTING SURVEYS* 49, 3 (2016).

[14] Mcs Filho, C. C. Monteiro, Pedro R. M. Inacio, and Mario M. Freire. 2017. Approaches for optimizing virtual machine placement and migration in cloud environments: A survey. *J. Parallel and Distrib. Comput.* 111, jan. (2017), 222–250.

[15] Minxian Xu and Rajkumar Buyya. 2019. Brownout approach for adaptive management of resources and applications in cloud computing systems: A taxonomy and future directions. *ACM Computing Surveys (CSUR)* 52, 1 (2019), 1–27.

[16] F.Zhang, G.Liu, X.Fu, and R.Yahyapour. 2018. A Survey on Virtual Machine Migration: Challenges, Techniques, and Open Issues. *IEEE Communications Surveys Tutorials* 20, 2 (2018), 1206–1243. DOI: http://dx.doi.org/10.1109/COMST.2018.2794881

[17] Prashanth C S R Bloch M T, Sridaran R. 2014. Analysis and survey of issues in live virtual machine migration interferences. *International Journal of Advanced Networking & Applications* (2014).

[18] Tarannum Bloch, R. Sridaran, and CSR Prashanth. 2018. Understanding Live Migration Techniques Intended for Resource Interference Minimization in Virtualized Cloud Environment. In *Big Data Analytics*, V. B. Aggarwal, Vasudha Bhatnagar, and Durgesh Kumar Mishra (Eds.). Springer Singapore, Singapore, 487–497.

[19] S. Amri, H.Hamdi, and Z.Brahmi. 2017. Inter-VM Interference in Cloud Environments: A Survey. In *2017 IEEE/ACS 14th International Conference on Computer Systems and Applications (AICCSA)*. 154–159. DOI: http://dx.doi.org/10.1109/AICCSA.2017.122

[20] F. Xu, F. Liu, H. Jin, and A. V. Vasilakos. 2014. Managing Performance Overhead of Virtual Machines in Cloud Computing:A Survey,State of the Art,and Future Directions. *Proc. IEEE* 102, 1 (2014), 11–31. DOI: http://dx.doi.org/10.1109/JPROC.2013.2287711

[21] Armando Fox, Rean Griffith, Anthony Joseph, Randy Katz, Andrew Konwinski, Gunho Lee, David Patterson, Ariel Rabkin, Ion Stoica, et al. 2009. Above the clouds: A berkeley view of cloud computing. *Dept. Electrical Eng. and Comput. Sciences, University of California, Berkeley, Rep. UCB/EECS* 28, 13 (2009), 2009.

[22] Dejan Novaković, Nedeljko Vasić, Stanko Novaković, Dejan Kostić, and Ricardo Bianchini. 2013. Deepdive: Transparently identifying and managing performance interference in virtualized environments. In *2013 {USENIX} Annual Technical Conference ({USENIX}{ATC} 13)*. 219–230.

[23] Kartik Joshi, Arun Raj, and Dharanipragada Janakiram. 2017. Sherlock: Lightweight detection of performance interference in containerized cloud services. In *2017 IEEE 19th International Conference on High Performance Computing and Communications; IEEE 15th International Conference on Smart City; IEEE 3rd International Conference on Data Science and Systems (HPCC/SmartCity/DSS)*. IEEE, 522–530.

[24] Virtualization. ([n. d.]). http://en.wikipedia.org/wiki/Virtualization

[25] Wang Xiaoxing Kong Xiangying. 2013. Analysis and research on performance isolation of virtualization [J]. *Electronic Measurement Technology* 8 (2013).

[26] Quan Chen, Shuai Xue, Shang Zhao, Shanpei Chen, Zhuo Song, Yihao Wu, Yu Xu, Tao Ma, Yong Yang, and Minyi Guo. 2020. Alita: comprehensive performance isolation through bias resource management for public clouds. In *2020 SC20: International Conference for High Performance Computing, Networking, Storage and Analysis (SC)*. IEEE Computer Society, 442–454.

[27] Abbas Dehghani Keyvan RahimiZadeh. 2021. Design and evaluation of a joint profit and interference-aware VMs consolidation in IaaS cloud datacenter. *CLUSTER COMPUTING-THE JOURNAL OF NETWORKS SOFTWARE TOOLS AND APPLICATIONS* 24, 4 (2021), 3249–3275.

[28] Willis Lang, Karthik Ramachandra, David J DeWitt, Shize Xu, Qun Guo, Ajay Kalhan, and Peter Carlin. 2016. Not for the Timid: On the Impact of Aggressive Over-booking in the Cloud. *Proceedings of the VLDB Endowment* 9, 13 (2016), 1245–1256.

[29] Xing Pu, Ling Liu, Yiduo Mei, Sankaran Sivathanu, Younggyun Koh, and Calton Pu. 2010. Understanding performance interference of i/o workload in virtualized cloud environments. In *2010 IEEE 3rd International Conference on Cloud Computing*. IEEE, 51–58.

[30] Dan Huang. *Managing IO Resource for Co-running Data Intensive Applications in Virtual Clusters*. Ph.D. Dissertation. College of Engineering and Computer Science.

[31] Homa Aghilinasab, Waqar Ali, Heechul Yun, and Rodolfo Pellizzoni. 2020. Dynamic Memory Bandwidth Allocation for Real-Time GPU-Based SoC Platforms. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* 39, 11 (2020), 3348–3360.

[32] Luis A Garrido and Paul Carpenter. 2017. vmca: Memory capacity aggregation and management in cloud environments. In *2017 IEEE 23rd International Conference on Parallel and Distributed Systems (ICPADS)*. IEEE, 674–683.

[33] Shin-gyu Kim, Hyeonsang Eom, and Heon Y Yeom. 2013. Virtual machine consolidation based on interference modeling. *the journal of Supercomputing* 66, 3 (2013), 1489–1506.

[34] Liuhua Chen, Haiying Shen, and Stephen Platt. 2016. Cache contention aware Virtual Machine placement and migration in cloud datacenters. In *2016 IEEE 24th International Conference on Network Protocols (ICNP)*. 1–10. DOI: http://dx.doi.org/10.1109/ICNP.2016.7784447

[35] Xiangping Bu, Jia Rao, and Cheng-zhong Xu. 2013. Interference and locality-aware task scheduling for MapReduce applications in virtual clusters. In *Proceedings of the 22nd international symposium on High-performance parallel and distributed computing*. 227–238.

[36] Xuesong Peng, Barbara Pernici, and Monica Vitali. 2018. Virtual machine profiling for analyzing resource usage of applications. In *International Conference on Services Computing*. Springer, 103–118.

[37] Fei Xu, Fangming Liu, Linghui Liu, Hai Jin, Bo Li, and Baochun Li. 2013. iAware: Making live migration of virtual machines interference-aware in the cloud. *IEEE Trans. Comput.* 63, 12 (2013), 3012–3025.

[38] Rachael Shaw, Enda Howley, and Enda Barrett. 2019. An Energy Efficient and Interference Aware Virtual Machine Consolidation Algorithm Using Workload Classification. In *International Conference on Service-Oriented Computing*. Springer, 251–266.

[39] Yuxia Cheng, Wenzhi Chen, Zonghui Wang, and Yang Xiang. 2017. Precise contention-aware performance prediction on virtualized multicore system. *Journal of Systems Architecture* 72 (2017), 42–50.

[40] Jiacheng Zhao, Huimin Cui, Jingling Xue, and Xiaobing Feng. 2015. Predicting cross-core performance interference on multicore processors with regression analysis. *IEEE Transactions on Parallel and Distributed Systems* 27, 5 (2015), 1443–1456.

[41] Quan Chen, Hailong Yang, Minyi Guo, Ram Srivatsa Kannan, Jason Mars, and Lingjia Tang. 2017. Prophet: Precise qos prediction on non-preemptive accelerators to improve utilization in warehouse-scale computers. In *Proceedings of the Twenty-Second International Conference on Architectural Support for Programming Languages and Operating Systems*. 17–32.

[42] Ron C Chiang and H Howie Huang. 2011. TRACON: Interference-aware scheduling for data-intensive applications in virtualized environments. In *Proceedings of 2011 International Conference for High Performance Computing, Networking, Storage and Analysis*. 1–12.

[43] Seyyed Ahmad Javadi, Sagar Mehra, Bharath Kumar Reddy Vangoor, and Anshul Gandhi. 2016. UIE: User-centric interference estimation for cloud applications. In *2016 IEEE International Conference on Cloud Engineering (IC2E)*. IEEE, 119–122.

[44] Hailong Yang, Alex Breslow, Jason Mars, and Lingjia Tang. 2013. Bubble-flux: Precise online qos management for increased utilization in warehouse scale computers. *ACM SIGARCH Computer Architecture News* 41, 3 (2013), 607–618.

[45] Xiao Zhang, Eric Tune, Robert Hagmann, Rohit Jnagal, Vrigo Gokhale, and John Wilkes. 2013. CPI2: CPU performance isolation for shared compute clusters. In *Proceedings of the 8th ACM European Conference on Computer Systems*. 379–391.

[46] Seyyed Ahmad Javadi and Anshul Gandhi. 2017. Dial: Reducing tail latencies for cloud applications via dynamic interference-aware load balancing. In *2017 IEEE International Conference on Autonomic Computing (ICAC)*. IEEE, 135–144.

[47] Yogesh D Barve, Shashank Shekhar, Ajay Chhokra, Shweta Khare, Anirban Bhattacharjee, Zhuangwei Kang, Hongyang Sun, and Aniruddha Gokhale. 2019. Fecbench: A holistic interference-aware approach for application performance modeling. In *2019 IEEE International Conference on Cloud Engineering (IC2E)*. IEEE, 211–221.

[48] Xi Chen, Lukas Rupprecht, Rasha Osman, Peter Pietzuch, Felipe Franciosi, and William Knottenbelt. 2015. Cloudscope: Diagnosing and managing performance interference in multi-tenant clouds. In *2015 IEEE 23rd international symposium on modeling, analysis, and simulation of computer and telecommunication systems*. IEEE, 164–173.

Just Accepted

[49] Fan-Hsun Tseng, Xiaofei Wang, Li-Der Chou, Han-Chieh Chao, and Victor CM Leung. 2017. Dynamic resource prediction and allocation for cloud data center using the multiobjective genetic algorithm. *IEEE Systems Journal* 12, 2 (2017), 1688–1699.

[50] Tajwar Mehmood, Seemab Latif, and Sheheryaar Malik. 2018. Prediction of cloud computing resource utilization. In *2018 15th International Conference on Smart Cities: Improving Quality of Life Using ICT & IoT (HONET-ICT)*. IEEE, 38–42.

[51] Xiaoli Sun, Qingbo Wu, Yusong Tan, and Fuhui Wu. 2014. Mvei: an interference prediction model for cpu-intensive application in cloud environment. In *2014 13th international symposium on distributed computing and applications to business, engineering and science*. IEEE, 83–87.

[52] Giuliano Casale, Stephan Kraft, and Diwakar Krishnamurthy. 2011. A model of storage I/O performance interference in virtualized systems. In *2011 31st International Conference on Distributed Computing Systems Workshops*. IEEE, 34–39.

[53] Achilleas Tzenetopoulos. 2020. Interference-aware Container Orchestration in Kubernetes Clusters. (2020).

[54] Christina Delimitrou and Christos Kozyrakis. 2013. ibench: Quantifying interference for datacenter applications. In *2013 IEEE international symposium on workload characterization (IISWC)*. IEEE, 23–33.

[55] Jason Mars, Lingjia Tang, and Mary Lou Soffa. 2011. Directly characterizing cross core interference through contention synthesis. (2011), 167–176.

[56] Sriram Govindan, Jie Liu, Aman Kansal, and Anand Sivasubramaniam. 2011. Cuanta: Quantifying Effects of Shared On-chip Resource Interference for Consolidated Virtual Machines. *ACM* (2011).

[57] Hamidreza Moradi, Wei Wang, Amanda Fernandez, and Dakai Zhu. 2020. uPredict: A User-Level Profiler-Based Predictive Framework in Multi-Tenant Clouds. In *2020 IEEE International Conference on Cloud Engineering (IC2E)*. IEEE, 73–82.

[58] Javid Taheri, Albert Y Zomaya, and Andreas Kassler. 2017. vmBBProfiler: a black-box profiling approach to quantify sensitivity of virtual machines to shared cloud resources. *Computing* 99, 12 (2017), 1149–1177.

[59] Dimosthenis Masouros, Sotirios Xydis, and Dimitrios Soudris. 2020. Rusty: Runtime interference-aware predictive monitoring for modern multi-tenant systems. *IEEE Transactions on Parallel and Distributed Systems* 32, 1 (2020), 184–198.

[60] Thomas Willhalm, Roman Dementiev, and Patrick Fay. 2012. Intel performance counter monitor-a better way to measure cpu utilization. *Dosegljivo: https://software. intel. com/en-us/articles/intelperformance-countermonitor-a-better-way-to-measure-cpu-utilization.[Dostopano: September 2014]* (2012).

[61] David Buchaca, Joan Marcual, Josep LLuis Berral, and David Carrera. 2020. Sequence-to-sequence models for workload interference prediction on batch processing datacenters. *Future Generation Computer Systems* (2020).

[62] Vinícius Meyer, Dionatrã F Kirchoff, Matheus L da Silva, and AF De Rose César. 2020. An Interference-Aware Application Classifier Based on Machine Learning to Improve Scheduling in Clouds. In *2020 28th Euromicro International Conference on Parallel, Distributed and Network-Based Processing (PDP)*. IEEE, 80–87.

[63] Uillian L Ludwig, Miguel G Xavier, Dionatrã F Kirchoff, Ian B Cezar, and César AF De Rose. 2019. Optimizing multi-tier application performance with interference and affinity-aware placement algorithms. *Concurrency and Computation: Practice and Experience* 31, 18 (2019), e5098.

[64] Wei Wei Lin Jinwei Wozniak Marcin Damasevicius Robertas Li Jingwei, Qi Yong. 2019. dCCPI-predictor: A state-aware approach for effectively predicting cross-core performance interference. *FUTURE GENERATION COMPUTER SYSTEMS-THE INTERNATIONAL JOURNAL OF ESCIENCE* 105 (2019), 184–195.

[65] VR Anu and Sherly Elizabeth. 2019. IALM: Interference Aware Live Migration Strategy for Virtual Machines in Cloud Data Centres. In *Data Management, Analytics and Innovation*. Springer, 499–511.

[66] Sa Wang, Wenbo Zhang, Tao Wang, Chunyang Ye, and Tao Huang. 2015. Vmon: Monitoring and quantifying virtual machine interference via hardware performance counter. In *2015 IEEE 39th Annual Computer Software and Applications Conference*, Vol. 2. IEEE, 399–408.

[67] Luis Carlos Jersak and Tiago Ferreto. 2016. Performance-aware server consolidation with adjustable interference levels. In *Proceedings of the 31st Annual ACM symposium on applied computing*. 420–425.

[68] Yu Gan, Yanqi Zhang, Kelvin Hu, Dailun Cheng, Yuan He, Meghna Pancholi, and Christina Delimitrou. 2019. Seer: Leveraging big data to navigate the complexity of performance debugging in cloud microservices. In *Proceedings of the Twenty-Fourth International Conference on Architectural Support for Programming Languages and Operating Systems*. 19–33.

[69] Scott Votke, Seyyed Ahmad Javadi, and Anshul Gandhi. 2017. Modeling and Analysis of Performance under Interference in the Cloud. In *2017 IEEE 25th International Symposium on Modeling, Analysis, and Simulation of Computer and Telecommunication Systems (MASCOTS)*. IEEE, 232–243.

[70] Zoltán Ádám Mann. 2015. Allocation of virtual machines in cloud data centers—a survey of problem models and optimization algorithms. *Acm Computing Surveys (CSUR)* 48, 1 (2015), 1–34.

[71] Wei Zhang, Sundaresan Rajasekaran, Shaohua Duan, Timothy Wood, and Mingfa Zhu. 2015. Minimizing Interference and Maximizing Progress for Hadoop Virtual Machines. *ACM SIGMETRICS Performance Evaluation Review* 42, 4 (2015), 62–71.

[72] Altino M Sampaio, Jorge G Barbosa, and Radu Prodan. 2015. PIASA: a Power and Interference Aware Resource Management Strategy for Heterogeneous Workloads in Cloud Data Centers. *Simulation Modelling Practice and Theory* 57 (2015), 142–160.

[73] Maicon Melo Alves, Luan Teylo, Yuri Frota, and Lúcia M.A. Drummond. 2018. An Interference-Aware Virtual Machine Placement Strategy for High Performance Computing Applications in Clouds. In *Symposium on High Performance Computing Systems*.

[74] Hedi Hamdi, Sabrine Amri, and Zaki Brahmi. 2019. Managing performance interference effects for Intelligent and efficient Virtual Machines Placement based on GWO Approach in Cloud. *IJCDS Journal* (2019).

[75] Jenn Wei Lin and Chien Hung Chen. 2012. Interference-aware virtual machine placement in cloud computing systems. In *International Conference on Computer & Information Science*.

[76] M.Reza HoseinyFarahabady, Javid Taheri, Albert Y. Zomaya, and Zahir Tari. 2021. Data-Intensive Workload Consolidation in Serverless (Lambda/FaaS) Platforms. In *2021 IEEE 20th International Symposium on Network Computing and Applications (NCA)*. 1–8. DOI: http://dx.doi.org/10.1109/NCA53618.2021.9685244

[77] C. K. Swain and A. Sahu. 2021. Interference Aware Workload Scheduling for Latency Sensitive Tasks in Cloud Environment. *Computing* 3 (2021), 1–26.

[78] Francisco Romero and Christina Delimitrou. 2018. Mage: online and interference-aware scheduling for multi-scale heterogeneous systems. In *the 27th International Conference*.

Just Accepted

[79] Evangelos Angelou, Konstantinos Kaffes, Athanasia Asiki, Georgios Goumas, and Nectarios Koziris. 2016. Improving virtual host efficiency through resource and interference aware scheduling. *arXiv preprint arXiv:1601.07400* (2016).
[80] Navarro Leandro Monte Enric Vlassov Vladimir Rameshan, Navaneeth. 2014. Stay-Away, protecting sensitive applications from performance interference. In *15th ACM/IFIP/USENIX International Middleware Conference*. 301–312.
[81] H. Lee, J. Lee, I. Yeom, and H. Woo. 2020. Panda: Reinforcement Learning-Based Priority Assignment for Multi-Processor Real-Time Scheduling. *IEEE Access* 8 (2020), 185570–185583. DOI : http://dx.doi.org/10.1109/ACCESS.2020.3029040
[82] Y.Bao, Y.Peng, and C.Wu. 2019. Deep Learning-based Job Placement in Distributed Machine Learning Clusters. In *IEEE INFOCOM 2019 - IEEE Conference on Computer Communications*. 505–513. DOI : http://dx.doi.org/10.1109/INFOCOM.2019.8737460
[83] Jeongseob Ahn, Changdae Kim, Jaeung Han, Young-Ri Choi, and Jaehyuk Huh. 2012. Dynamic virtual machine scheduling in clouds for architectural shared resources. In *Presented as part of the*.
[84] Hadi Salimi and Mohsen Sharifi. 2013. Batch scheduling of consolidated virtual machines based on their workload interference model. *Future Generation Computer Systems* 29, 8 (2013), 2057–2066.
[85] R.Nishtala, V.Petrucci, P.Carpenter, and M.Sjalander. 2020. Twig: Multi-Agent Task Management for Colocated Latency-Critical Cloud Services. In *2020 IEEE International Symposium on High Performance Computer Architecture (HPCA)*. 167–179. DOI : http://dx.doi.org/10.1109/HPCA47549.2020.00023
[86] Qian Zhu and Teresa Tung. 2014. Performance interference model for managing consolidated workloads in QOS-aware clouds. (May 2014). US Patent 8,732,291.
[87] Yiling Qin, Lun Zhang, Fei Xu, and Daidong Luo. 2019. Interference and Topology-Aware VM Live Migrations in Software-Defined Networks. In *2019 IEEE 21st International Conference on High Performance Computing and Communications; IEEE 17th International Conference on Smart City; IEEE 5th International Conference on Data Science and Systems (HPCC/SmartCity/DSS)*. IEEE, 1068–1075.
[88] Mohsen Tarighi, Seyed Ahmad Motamedi, and Ehsan Arianyan. 2010. Performance improvement of virtualized cluster computing system using TOPSIS algorithm. In *The 40th International Conference on Computers & Indutrial Engineering*. IEEE, 1–6.
[89] Renuga Kanagavelu, Bu Sung Lee, Nguyen The Dat Le, Luke Ng Mingjie, and Khin Mi Mi Aung. 2014. Virtual machine placement with two-path traffic routing for reduced congestion in data center networks. *Computer Communications* 53, nov.1 (2014), 1–12.
[90] Subhadra Bose Shaw and Anil Kumar Singh. 2015. Use of proactive and reactive hotspot detection technique to reduce the number of virtual machine migration and energy consumption in cloud data center. *Computers & Electrical Engineering* (2015).
[91] Linjiun Tsai and Wanjiun Liao. 2012. Cost-aware workload consolidation in green cloud datacenter. In *IEEE International Conference on Cloud Networking*.
[92] Chen Wei, Qiao Xiaoqiang, Wei Jun, and Huang Tao. 2012. A Profit-Aware Virtual Machine Deployment Optimization Framework for Cloud Platform Providers. In *IEEE Fifth International Conference on Cloud Computing*.
[93] Aman Kansal, Feng Zhao, Jie Liu, Nupur Kothari, and Arka A Bhattacharya. 2010. Virtual machine power metering and provisioning. In *Proceedings of the 1st ACM Symposium on Cloud Computing*. 39–50.
[94] Ismael Solis Moreno, Renyu Yang, Jie Xu, and Tianyu Wo. 2013. Improved energy-efficiency in cloud datacenters with interference-aware virtual machine placement. In *2013 IEEE Eleventh International Symposium on Autonomous Decentralized Systems (ISADS)*. IEEE, 1–8.
[95] Xibo Jin, Fa Zhang, Lin Wang, Songlin Hu, Biyu Zhou, and Zhiyong Liu. 2015. Joint optimization of operational cost and performance interference in cloud data centers. *IEEE Transactions on Cloud Computing* 5, 4 (2015), 697–711.
[96] R. Nasim, J. Taheri, and A. J. Kassler. 2016. Optimizing Virtual Machine Consolidation in Virtualized Datacenters Using Resource Sensitivity. In *IEEE CloudCom 2016*.
[97] K. Wang, Mmh Khan, N. Nguyen, and S. Gokhale. 2017. Design and implementation of an analytical framework for interference aware job scheduling on Apache Spark platform. *Cluster Computing* 22 (2017), 2223–2237. DOI : http://dx.doi.org/10.1007/s10586-017-1466-3
[98] Yasaman Amannejad, Diwakar Krishnamurthy, and Behrouz Far. 2015. Detecting performance interference in cloud-based web services. In *2015 IFIP/IEEE International Symposium on Integrated Network Management (IM)*. IEEE, 423–431.
[99] Yusen Li, Chuxu Shan, Ruobing Chen, Xueyan Tang, Wentong Cai, Shanjiang Tang, Xiaoguang Liu, Gang Wang, Xiaoli Gong, and Ying Zhang. 2019. GAugur: Quantifying performance interference of colocated games for improving resource utilization in cloud gaming. In *Proceedings of the 28th international symposium on high-performance parallel and distributed computing*. 231–242.
[100] Faruk Caglar, Shashank Shekhar, and Aniruddha Gokhale. 2011. Towards a Performance Interference-aware Virtual Machine Placement Strategy for Supporting Soft Real-time Applications in the Cloud. *Universidad Carlos III De Madrid* (2011).
[101] Marisol García-Valls, Tommaso Cucinotta, and Chenyang Lu. 2014. Challenges in real-time virtualization and predictable cloud computing. *Journal of Systems Architecture* 60, 9 (2014), 726–740.
[102] Zhiheng Zhong, Minxian Xu, Maria Alejandra Rodriguez, Chengzhong Xu, and Rajkumar Buyya. 2022. Machine learning-based orchestration of containers: A taxonomy and future directions. *ACM Computing Surveys (CSUR)* (2022).
[103] Wen-Yan Chen, Ke-Jiang Ye, Cheng-Zhi Lu, Dong-Dai Zhou, and Cheng-Zhong Xu. 2020. Interference analysis of co-located container workloads: A perspective from hardware performance counters. *Journal of Computer Science and Technology* 35 (2020), 412–417.
[104] Z. Ou, H. Zhuang, J. K. Nurminen, A. Yl-Jski, and P. Hui. 2012. Exploiting Hardware Heterogeneity within the Same Instance Type of Amazon EC2. In *The 4th USENIX Workshop on Hot Topics in Cloud Computing (HotCloud '12)*.
[105] Benjamin Farley, Ari Juels, Venkatanathan Varadarajan, Thomas Ristenpart, and Michael M. Swift. 2012. More for Your Money: Exploiting Performance Heterogeneity in Public Clouds. In *Acm Symposium on Cloud Computing*.
[106] Fei Xu, Fangming Liu, and Hai Jin. 2016. Heterogeneity and Interference-Aware Virtual Machine Provisioning for Predictable Performance in the Cloud. *IEEE Trans. Comput.* 65, 8 (2016), 2470–2483.
[107] Kui Su, Lei Xu, Cong Chen, Wenzhi Chen, and Zonghui Wang. 2015. Affinity and conflict-aware placement of virtual machines in heterogeneous data centers. In *2015 IEEE Twelfth International Symposium on Autonomous Decentralized Systems*. IEEE, 289–294.

Just Accepted

[108] Kejiang Ye, Haiying Shen, Yang Wang, and Chengzhong Xu. 2020. Multi-tier Workload Consolidations in the Cloud: Profiling, Modeling and Optimization. *IEEE Transactions on Cloud Computing* PP, 99 (2020), 1–1.

Just Accepted