
Fei Xu, Fangming Liu, Member, IEEE, Hai Jin, Senior Member, IEEE, Athanasios V. Vasilakos, Senior Member, IEEE

Abstract—Infrastructure-as-a-Service (IaaS) cloud computing offers customers (tenants) a scalable and economical way to provision virtual machines (VMs) on demand while charging them only for the leased computing resources by time. However, due to the VM contention on shared computing resources in datacenters, this new computing paradigm inevitably brings noticeable performance overhead (i.e., unpredictable performance) of VMs to tenants, which has become one of the primary issues of the IaaS cloud. Consequently, increasing efforts have recently been devoted to guaranteeing VM performance for tenants. In this survey, we review the state-of-the-art research on managing the performance overhead of VMs, and summarize them under diverse scenarios of the IaaS cloud, ranging from the single server virtualization, a single mega datacenter, to multiple geo-distributed datacenters. Specifically, we unveil the causes of VM performance overhead by illustrating representative scenarios, discuss the performance modeling methods with a particular focus on their accuracy and cost, and compare the overhead mitigation techniques by identifying their effectiveness and implementation complexity. With the obtained insights into the pros and cons of each existing solution, we further bring forth future research challenges pertinent to the modeling methods and mitigation techniques of VM performance overhead in the IaaS cloud.

Index Terms—Cloud computing, virtualization, predictable performance, VM performance overhead.

1 INTRODUCTION

With the ability to scale computing resources on demand and provide a simple pay-as-you-go business model for customers, cloud computing is emerging as an economical computing paradigm, and has gained much popularity in the industry. Currently, a number of big companies such as Netflix and Foursquare [1], have successfully moved their business services from the dedicated computing infrastructure to Amazon Elastic Computing Cloud (EC2) [2], which is a leading public Infrastructure-as-a-Service (IaaS) cloud platform worldwide. Undoubtedly, more individuals (tenants) and enterprises will leverage the cloud to maintain or scale up their business while cutting down the budget, as reported by International Data Corporation (IDC) that the business revenue brought by cloud computing will reach 1.1 trillion US dollars by 2015 [3].

Unfortunately, the running performance of virtual machines (VMs) on the IaaS cloud platform is unpredictable, which significantly impacts the tenant’s Service-Level Agreement (SLA) offered by the cloud provider [4]. Specifically, due to the resource contention of VMs, the VM performance obtained in the IaaS cloud is severely degraded and highly variable, compared to the performance of VMs running in an isolated environment (e.g., running alone on a physical server in a local cluster) [5]. In this paper, we formally refer to such VM performance degradation and variation as VM performance overhead. Not surprisingly, several latest measurement studies on Amazon EC2 [2] have recently shown that diverse cloud applications listed in Table 1, ranging from latency-sensitive [6], network-sensitive [7], to data-intensive [5], suffer significant performance overhead, which consequently hinders users from moving their performance-critical business to the cloud. In particular, the performance degradation and variation metrics listed in Table 1 are measured as (1) the ratio of the degraded performance obtained in the IaaS cloud, in terms of the increased job execution time and the decreased network throughput, to the application performance obtained in isolated VMs, and (2) the variation of application performance obtained in the IaaS cloud over a period of time. Accordingly, these evidences bring forth a series of urgent demands for managing the performance overhead of VMs and guaranteeing VM performance for tenants in the cloud.

From a high level perspective of the IaaS cloud, existing studies control the performance overhead of VMs under three representative scenarios, ranging from the single server virtualization, a single large datacenter, to multiple geo-distributed datacenters, as summarized in Fig. 1. On one hand, even with built-in resource isolation mechanisms on CPU cores, memory and disk capacities across VMs [8], the contention on shared cache and I/O
bandwidth resources among co-located VMs is difficult to be alleviated by existing Hypervisors [9] in practice [10]. Recent efforts on single server virtualization are primarily devoted to alleviating the VM performance overhead caused by the contention on shared server resources (e.g., CPU cache and I/O bandwidth) by resource isolation. Moreover, to gain deep understandings on the relationship between the performance overhead and resource consumption of VMs, there have been a number of recent studies devoted to the performance modeling. With an accurate modeling method, the VM performance overhead can be alleviated in a cost-effective manner, as compared to the resource isolation techniques.

On the other hand, the performance overhead of VMs within a single mega datacenter and across multiple geodistributed datacenters, however, can hardly be managed by the traditional resource isolation techniques. Specifically, in a single large datacenter, the distributed network and storage resources are shared among a huge amount of VMs leased by multiple tenants, as illustrated in Fig. 2. Such a cloud scenario undoubtedly makes the VM performance issue more intractable than the single server virtualization. Furthermore, among multiple geodistributed datacenters, the issue of VM performance overhead becomes even more complicated due to the highly variable and scarce network resource over the Wide Area Network (WAN). While a number of solutions that mitigate such a VM performance issue have emerged recently, the fast innovations of these overhead mitigation techniques mandate comprehensive surveying of the state-of-the-art literature under the three cloud scenarios discussed above, with the aim of highlighting future directions in managing the performance overhead of VMs in IaaS clouds.

Therefore, we compare the latest solutions of managing the performance overhead of VMs under the three cloud scenarios shown in Fig. 1. The focus of this paper is on obtaining the insights into the benefits and costs of these solutions, as well as raising the future research challenges on the performance modeling methods and overhead mitigation techniques.

There also exists several alternative approaches dealing with the VM performance issue. For example, CloudCmp [11] has been developed for guiding tenants in selecting the IaaS cloud platform with the best application performance from several cloud providers, such as Amazon EC2 [2], Microsoft Azure [12], and Rackspace Cloud Servers [13]. Several studies [14], [15] explore the hardware heterogeneity within the same type of Amazon EC2 instances (i.e., VMs). Potential improvements of VM performance can accordingly be obtained by placing workloads onto the instances with good performance in the EC2 datacenters. The strategies above can provide acceptable VM performance and reduce budgets on behalf of tenants, while they might hurt the benefits of cloud providers. Accordingly, we limit the scope of this survey to the solutions that alleviate VM performance overhead on behalf of both tenants and cloud providers.

The rest of this survey is organized as follows. Sec. 2 explores the causes and summarizes the metrics of VM performance overhead by illustrating the representative scenarios of the IaaS cloud. The existing measurement and modeling methods of VM performance overhead as well as the future research issues are discussed in Sec. 3. Sec. 4 summarizes and compares the state-of-the-art solutions that manage VM performance overhead under diverse scenarios of the IaaS cloud, along with a fruitful discussion of future research challenges. Finally, we conclude this survey in Sec. 5.

2 Causes and Metrics of VM Performance Overhead in IaaS Cloud

There are various kinds of causes that can lead to the performance overhead of VMs in IaaS cloud datacenters, as illustrated in Fig. 1. In this section, we first expose and discuss these causes of VM performance overhead in detail, especially on single server virtualization, within a single datacenter, and across multiple geo-distributed datacenters, respectively. We next summarize the metrics of VM performance overhead used in the literature.
2.1 Performance Impact of Single Server Virtualization

The server virtualization technique [9], which serves as the cornerstone of cloud computing, multiplexes the computing resources of a physical server for multiple co-located VMs. Current Virtual Machine Monitors (VMMs) or Hypervisors [9] (e.g., Citrix XenServer, VMware vSphere and Microsoft Hyper-V) have already provided good performance isolation mechanisms on sharing the resource of CPU cores, memory and disk capacities among co-located VMs. The isolation mechanisms can be implemented by fully-fledged CPU schedulers (e.g., Credit Scheduler in Xen [8]) and static partition of memory and disk capacities. However, server resources like CPU cache space, memory bandwidth, network and disk I/O bandwidth are very hard to be isolated in existing Hypervisors [10]. This inevitably leads to the contention on these aforementioned server resources among co-located VMs. As shown in [16], with each VM allocated to a dedicated CPU core, statically partitioned memory capacity and disk space, the execution time of the lbm application from the SPECCPU 2006 benchmark [17] deployed on two co-located VMs on an Intel Core-2-Duo processor can be prolonged by up to 120%, compared with the performance of the lbm application VM running alone. The above significant performance overhead of VMs is caused by the contention of cache and memory bandwidth on a physical server.

Furthermore, with the tide of green computing and the recent prevalence of deploying multi-core architecture systems in datacenters, VM consolidation is now widely used to improve the resource utilization of a multi-core physical server [18] and reduce the energy consumption of datacenters [19]. As more VMs are co-located on a physical server, VM consolidation, however, increases the severity of performance overhead of these co-located VMs on a physical server. As a result, the contention of shared cache and I/O bandwidth resources among co-located VMs on a physical server becomes one of the causes of VM performance overhead in the IaaS cloud.

2.2 Cause of VM Performance Overhead within a Single Datacenter

It is essential for tenants to lease a cluster of VMs from cloud providers when processing large jobs with a large amount of data (e.g., terabyte), such as MapReduce [20] and scientific workloads. In such cases, one natural question is that: are the performance isolation techniques for co-located VMs on a single physical server enough to eliminate the performance overhead among multiple VMs in a single datacenter? Unfortunately, the routine operations of the IaaS cloud datacenter, such as the (concurrent) live migration, deployment and snapshotting of multiple VMs, the contention of multiple VMs on the shared cloud network and storage resources, complicate the issue of VM performance overhead and significantly degrade the performance of VMs. We elaborate them in the following three cases, respectively.

The (concurrent) live migration of VMs [21] is an indispensable tool that enables the load balancing and power saving of a datacenter. It can nevertheless cause the performance overhead to the migrating VMs and co-located VMs on the migration source and destination servers, which is illustrated in Fig. 2. As shown in [22], the live migration of a VM configured with 256 MB memory hosting the MySQL server, over a 100 Mbps Ethernet link, severely prolongs the end-to-end mean response time of the multi-tier Web applications (e.g., RUBiS [23]) by 211%, compared with the application performance before the live VM migration. Undoubtedly, migrating multiple VMs simultaneously in a cloud datacenter will overload the datacenter network even worse, which further constrains the performance of CPU and network resources on the migration and destination servers, and incurs a long migration time (i.e., from several minutes to hours) [24].

The concurrent deployment and snapshotting of multiple VMs are two commonly occurring patterns of the routine operations in an IaaS cloud datacenter [25]. For example, to execute jobs with a large amount of data, it is the basic requirement of tenants to fast deploy (initialize) a
large number of VMs simultaneously in a public IaaS cloud platform [26]. As for the private cloud owned by a big IT enterprise, it is also the common demand of the employees for concurrently provisioning a large number of VMs fast in the daily working hours, even when there are a burst of VM deployment requests. Once the VMs are initialized and running, there is another basic requirement of tenants to fast take snapshots of their leased cluster of VMs simultaneously. Fast snapshotting is also desired by Amazon Elastic Block Store (EBS) [27] for the checkpoint of VM states or the VM data sharing among workmates. Currently, both the public and private clouds, however, suffer the VM performance overhead caused by the concurrent deployment and snapshotting of multiple VMs. This is because these two operation patterns can generate a significant amount of network traffic, which inevitably interferes with the other VMs in the same datacenter [25].

The inability to isolate shared cloud network and storage resources among multiple VMs is another cause of performance overhead of VMs within a datacenter. In general, to serve the disk and network I/O requests for tenant VMs, the IaaS cloud datacenter deploys the shared back-end storage such as Storage Area Network (SAN), and shares the network resources such as network bandwidth on physical servers and switches, as shown in Fig. 2. This accordingly leads to the I/O contention on the shared storage and network resources among multiple VMs of tenants in a cloud datacenter. It is revealed by the latest measurements on Amazon EC2 that the standard medium instances experience up to 66% variation of the network I/O throughput [28], and the write I/O bandwidth of standard small instances can vary by as much as 50% from the mean [6].

2.3 Cause from Live WAN Migration across Multiple Geo-distributed Datacenters

It is reported by Gartner Inc. that the hybrid use of public and private clouds becomes an imperative trend of the cloud computing [29], and more than half of large enterprises over the world will have such hybrid cloud deployments by the end of 2017 [30]. A commercial solution, Amazon Virtual Private Cluster (VPC) [31], now allows the private datacenter of IT companies connecting to a cluster of Amazon EC2 instances. These evidences above hence bring new challenges to manage VM performance overhead across the multiple geo-distributed datacenters, such as the performance overhead of VMs caused by the live migration of VMs and storage over the WAN connection.

Specifically, live migration of VMs and storage across geo-distributed datacenters over a WAN connection serves as an indispensable technique for managing computing resources of these datacenters. For big IT enterprises with multiple geo-distributed datacenters, for example, live VM migration over the WAN enables the load balancing and power saving across datacenters, by migrating several VMs from heavy-loaded datacenters to light-loaded datacenters and consolidating the VMs from the small datacenter into the large datacenter, respectively. For small IT companies deploying their business on the hybrid cloud, live WAN migration of VMs allows them to move or replicate their business services to the public cloud from the private cloud when the loads of service requests are bursting [32].

Different from the live VM migration in a Local Area Network (LAN) discussed in Sec. 2.2, the live WAN migration of VMs includes the data transfer of not only the VM memory state, but also the VM disk image and the on-going network connections [33]. Accordingly, this would cause a long VM migration time, application downtime, and a large amount of network traffic, which further leads to severe performance overhead to a certain number of VMs in the migration source and destination datacenters over the WAN, as well as the migrating VMs.

Such performance overhead cannot be easily managed by VM performance guarantee solutions on a single server or in a single datacenter. For example, over the WAN with a maximum throughput of 85 Mbps, the live migration of a VM configured with 1.7 GB memory and 10 GB disk hosting a Web service application, performed between the datacenters in Illinois and Texas, can undergo up to 40 minutes of disk transfer and 210 seconds of memory transfer. The response time of the Web service during the migration process is highly variable, can be degraded to 52 milliseconds on average, as compared to 10 milliseconds before the migration [32]. Consequently, the migration of VMs over the WAN becomes another cause of performance overhead of VMs in IaaS clouds.

2.4 Evaluation and Metrics of VM Performance Overhead

After discussing the causes of VM performance overhead, the next natural and fundamental question is that: how can we evaluate the performance overhead of VMs, and what are the metrics? To simply capture the performance overhead of VMs, the first type of metric is the performance degradation of various applications running in the IaaS cloud, compared with the application performance obtained in the VMs running in an isolated environment. Specifically, the performance degradation of VMs $P_d$ can be measured as

$$P_d = \frac{|x_{iaas} - x_{isolation}|}{x_{isolation}}.$$  \hspace{1cm} (1)

where $x_{iaas}$ and $x_{isolation}$ are the performance of VM applications running in the IaaS cloud and in isolation, respectively. The larger magnitude of performance degradation value $P_d$ indicates the more severe performance overhead of VMs.

The second type of metric that capture the performance overhead of VMs is the variation of VM performance obtained in the IaaS cloud over a period of time. Specifically, the VM performance variation $P_v$ can be represented by the Coefficient of Variation [34], which is
formulated as
\[ P_v = \frac{1}{\bar{x}} \sqrt{\frac{1}{n_x - 1} \sum_{i=1}^{n_x} (x_i - \bar{x})^2}, \]  
(2)

where \( x_i \) is the VM performance measured in the IaaS cloud over time, \( n_x \) is the number of performance measurements, \( \bar{x} \) is the average of VM performance over time. The larger magnitude of performance variation value \( P_v \) implies the more severe performance overhead of VMs.

Due to the security concern and noticeable measurement costs, the performance degradation of VM applications can hardly be obtained in several scenarios. In such cases, the third type of metric is required to reflect the performance overhead of VMs by examining the performance of routine operations in IaaS cloud datacenters, as introduced in Sec. 2.2 and 2.3. Specifically, such a type of metric includes the duration of live VM migration (i.e., migration time), downtime of the migrating VM, energy consumption of live VM migration, the amount of network traffic caused by the live VM migration, deployment, and snapshotting. The smaller migration time, migration downtime, amount of network traffic and energy consumption implies the better performance of routine operations and lighter performance overhead of VMs. In summay, existing studies in the literature leverage the above three types of metrics to evaluate the performance overhead of VMs in the IaaS cloud.

3 Models and Measurements on VM Performance Overhead

In order to well manage the performance overhead of VMs in datacenters, the primary requirement is to accurately model and measure VM performance overhead. Existing studies have devoted great efforts to modeling the performance overhead of VMs on two aspects: performance interference of co-located VMs on a physical server and VM performance overhead caused by the live migration of VMs. We review the representative performance modeling methods on the two aspects, and summarize them in Table 2 and Table 3, respectively. In particular, the table column of kind/type of model indicates whether the model is linear or non-linear to its parameters. The column of resource consumption profile in Table 2 represents the parameters of VM performance interference models. The table column of prediction error shows the accuracy of the predicted VM performance overhead in comparison to the measured overhead data. Specifically, the smaller value of the prediction error implies the more accurate prediction modeling approach.

3.1 Models of VM Performance Overhead Caused by Single Server Virtualization

To model the performance interference of co-located VMs on a single server, the representative approach is to develop an accurate function of resource consumption profile of these VMs. For example, Zhu et al. [35] propose a simple dilation factor to represent the performance interference between two co-located VMs on each of the four-dimensional resource consumption of VMs (i.e., CPU, memory, network and disk I/O). By pre-running each workload which is co-located with a background benchmark workload, this method manually varies the resource consumption of workloads (e.g., varying CPU consumption from 10% to 100% with a step of 10%), and acquires the dilation factors offline for the workloads. The larger dilation factor means the more severe performance interference of co-located VMs. Similar with the previous approach, Koh et al. [36] predict the performance interference between two VMs by two linear statistical approaches (i.e., weighted mean method and linear regression analysis). In addition, this method enriches the resource consumption profile of VMs with other system-level workload metrics, such as the statistics of cache hits and misses, VM switches. The above linear statistical model of performance interference \( I \) is presented as

\[ I = \alpha_0 + \sum_{i=1}^{p} \alpha_i X_i, \]  
(3)

where \( X_i \) is the \( i \)-th dimension of VM resource consumption, \( p \) is the dimensions of VM resource consumption, and \( \alpha_0, \alpha_i \) are the model coefficients.

With a particular focus on the co-located VMs hosting data-intensive applications, two non-linear statistical models have been developed in [37] and [38]. Specifically, by extending the linear model shown in Eq. (3), TRACON [37] models the performance interference \( I \) between two co-located VMs as a quadratic function of CPU and I/O bandwidth consumption \( X_i \) of VMs, which is given by

\[ I = \alpha_0 + \left( \sum_{i=1}^{4} \alpha_i X_i \right)^2, \]  
(4)

A more accurate model has been presented in [38], where the performance interference of multiple co-located VMs is modeled as an exponential function of VM CPU and I/O resource consumption, which is shown as below,

\[ I = \alpha_0 + \alpha_1 \exp \left( \sum_{i} \gamma_i X_i^{cpu} \right) + \alpha_2 \exp \left( \sum_{i} \omega_i X_i^{io} \right). \]  
(5)

where \( X_i^{cpu} \) and \( X_i^{io} \) are the CPU and I/O resource consumption of VMs, respectively. \( \gamma_i \) and \( \omega_i \) are the model coefficients. Nevertheless, these above linear and non-linear statistical models in Eq. (3), (4) and (5) require the offline-training of experimental statistics to obtain the model coefficients, e.g., \( \alpha_i, \gamma_i \). To particularly predict the performance interference on the cache and memory bandwidth among multiple co-located VMs, Cuanta [16] first leverages a certain synthetic cache benchmark and an interference matrix to create the cache clones with the same cache behaviors of these VMs. This method then looks up a degradation table to model the effect of interference from co-located VMs. In particular, the
interference matrix and degradation table are generated offline by the pre-runs of the cache benchmark and VMs.

Whereas the previous approaches require pre-run of workloads to acquire necessary parameters of the statistical prediction model, several studies make efforts to break such a limitation. For example, pSciMapper [19] leverages the Pearson Correlation Coefficient [34] of multi-dimensional VM resource consumption (e.g., CPU, memory, network and disk I/O), to estimate the performance interference $I_{XY}$ between two VMs. Specifically, the interference model is given by

$$I_{XY} = \frac{p \sum X_i Y_i - \sum X_i \sum Y_i}{\sqrt{p \sum X_i^2 - (\sum X_i)^2} \cdot \sqrt{p \sum Y_i^2 - (\sum Y_i)^2}},$$

where $X_i$ and $Y_i$ are the resource consumption of the two co-located VMs, and $p$ is the dimensions of VM resource consumption. In particular, the larger correlation result $I_{XY}$ indicates the more severe VM performance interference. Q-Clouds [39] develops a discrete-time (linear or non-linear) model of history performance feedbacks and current CPU resource caps of VMs $u_i$, to online predict the current performance of co-located VMs $P_i$. At each time step $i$, the discrete-time model is given by

$$P_i = \Phi(P_{i-1}, \ldots, P_{i-n}, u_i, \ldots, u_{i-m}),$$

where $m$ and $n$ represent to what extent the history values of performance data and CPU resource caps impact the current performance, respectively. In particular, the model construction of $\Phi(\cdot)$ requires a learning phase using the historical observed performance data $P_i$ and CPU resource caps $u_i$ of multiple VMs.

### 3.2 Models of VM Performance Overhead Caused by Live Migration of VMs

As discussed in Sec. 2.4, the migration time is an important metric that reflects VM performance overhead caused by the live migration of VMs. A number of methods have recently been proposed to model the migration time. For example, Wu et al. [40] particularly study the relationship between the domain-0 CPU allocation $U_{cpu}$ and the live migration time $T_{mig}$. Specifically, given a VM to be migrated, $T_{mig}$ can be modeled as a non-linear function of $U_{cpu}$, which is given by

$$T_{mig} = \frac{\beta}{U_{cpu}},$$

where $\beta$ and $\kappa$ are the model coefficients, and $\beta, \kappa \in (0, \infty)$. In particular, these coefficients require the pre-run of workload applications and are trained offline using the experimental results of multiple pre-runs. Given a VM migration environment, CloudScale [41] estimates the migration time $T_{mig}$ using a linear function of the memory size $V_{mem}$ of the migrating VM. Similar to Eq. (8), such a linear function is also derived based on the regression analysis of experimental results of multiple migration pre-runs. The impact of another two parameters (i.e., network link speed $R$ and memory page dirty rate $D$) to the VM migration time $T_{mig}$ has been examined by simulation models in [42], where $T_{mig}$ is proved to be non-linear to $R$ and $D$.

Based on the four parameters (i.e., $U_{cpu}$, $V_{mem}$, $R$ and $D$) above, an integrated model of the VM migration time $T_{mig}$ has been developed in [43], which is given by

$$T_{mig} = \sum_{i=1}^{n} t_i, \quad t_i = \begin{cases} \frac{t_i-1 \cdot D}{r}, & i \geq 2 \\ \frac{V_{mem}}{r}, & i = 1 \end{cases},$$

where $n$ is the total rounds of memory pre-copy phases$^1$, $t_i$ is the phase time for each round of memory pre-copy, and $r$ is the network transmission rate for the live VM migration. As $r$ highly correlates with $U_{cpu}$ [40] and is restricted by $R$, it can be further modeled as $r = f(U_{cpu}, R)$, where $f(\cdot)$ is a non-linear function. In addition to the performance metric of migration time, Liu et al. [43] further develop a series of statistical models to predict several other important VM migration metrics including the network traffic $N_{mig}$, energy consumption $E_{mig}$, and application downtime $T_{down}$, respectively. Accordingly, the migration performance $P_{mig}$ of migrating a VM can be comprehensively evaluated as

$$P_{mig} = a \cdot T_{mig} + b \cdot T_{down} + c \cdot N_{mig} + d \cdot E_{mig},$$

where $a$, $b$, $c$ and $d$ are the model coefficients. In particular, the application downtime $T_{down}$ is the phase time for the last round of memory copy $t_r$, according to Eq. (9). The network traffic $N_{mig}$ is the total transferred memory pages during the live VM migration, which is calculated as $V_{mem} + \sum_{i=1}^{n} t_i \cdot D$. The energy consumption $E_{mig}$ is proportional to the network traffic $N_{mig}$, which can be modeled as $\rho \cdot N_{mig} + \theta$, where $\rho$ and $\theta$ are the model

$^1$ We mainly discuss the pre-copy migration [21] in this survey.
coefficients. In particular, these model coefficients are obtained based on the regression analysis of experimental statistics which are measured offline.

The former approaches model the performance overhead of VMs caused by the live migration from the perspective of migration performance itself. While from the perspective of the migrating VM and other co-located VMs on the migration source and destination servers, VM performance overhead can simply be measured as performance degradation for each specific application in an offline manner during the live migration [22], [41]. In addition, such a VM performance overhead can be modeled mathematically. Specifically, considering the live VM migration as a job that consumes the computing resources of physical server, Lim et al. [44] predict the performance overhead (i.e., the prolonged execution time \( T_{ext} \)) of co-located VMs on the migration source and destination servers, using a two-resource contention model which is given by

\[
T_{ext} = \tau(q_1 q_2 + (1 - q_1)(1 - q_2)),
\]

where \( \tau \) is the application execution time running alone on a physical server. \( q_1, q_2 \) are the probabilities of accessing a kind of VM resource by the live VM migration and the co-located VM, respectively, and \( q_1, q_2 \in [0, 1] \). The parameters in Eq. (11) are acquired offline using the VM performance statistics. Especially for the migrating VM hosting a Web server, Breitgand et al. [45] adopt the queuing theory to model the Service Level Objective (SLO) violation of such a migrating Web application, which is quantified as a function of the network bandwidth allocated to the live VM migration \( B_m \). Specifically, given the SLO of the Web service \( t_{SLO} \) and the maximum network bandwidth of the Web server \( B \), the probability of SLO violations \( P(\tau > t_{SLO}) \) is modeled by

\[
P(\tau > t_{SLO}) = \exp((t_{SLO} - \mu(B - B_m)))
\]

where \( \mu(B - B_m) \) denotes the requests’ response rate of the Web server, given the migration bandwidth \( B_m \). \( \lambda \) is the requests arrive rate to the Web server.

### 3.3 Summaries, Insights and Open Research Issues

From the above comparison of performance modeling methods summarized in Table 2 and Table 3, we raise the following conclusions, insights and open research issues:

**First**, statistical methods, such as regression analysis and correlation analysis [34], have been proved to be effective in the model construction and estimation of VM performance overhead caused by the single server virtualization and live VM migration. Although the statistical methods are widely used in the existing modeling approaches with a high prediction accuracy, most of them require the pre-run of VM workloads to acquire necessary parameters of the prediction models of VM performance overhead. Such a requirement is not always practical in datacenters, as it is under the assumption that few kinds of workloads are running on the homogeneous physical servers. In real-world cloud datacenters, the heterogeneity of hardwares [14] and workloads [46] becomes common. The pre-run of workloads or offline measurements on each type of hardware will undoubtedly bring noticeable processing overhead to the model construction. Accordingly, how to develop a lightweight yet effective prediction model of VM performance overhead without the pre-run of workloads becomes an open research problem.

**Second**, from the above discussion on the complexity of model construction, we further obtain a key insight into the relationship between the accuracy and cost (e.g., pre-run of VM workloads) of the model of VM performance overhead. For example, although the correlation analysis of VM resource consumption in Eq. (6) does not require the pre-run of VM workloads, this method only provides an estimation of performance interference between two VMs, rather than an accurate value of the VM performance interference. As a result, the accuracy of the prediction model should be determined by its practical usage. On one hand, a lightweight estimation model that catches the qualitative trend of VM performance overhead is sufficient to guide in determining an appropriate assignment of VMs, for mitigating the performance overhead of VMs (e.g., [19]). On the other hand, a sophisticated prediction model that accurately quantifies the performance overhead of VMs is required to re-allocate the computing resources to the competing VMs within a single datacenter (e.g., [39]).

**Third**, although a number of performance models of single server virtualization have been proposed (refer to...
Table 2), there lacks a holistic prediction model to quantify the performance overhead of multiple (i.e., more than three) co-located VMs on multi-dimensional resources of a single physical server (i.e., CPU cache, memory bandwidth, network and disk I/O bandwidth) without the pre-run of workloads. Hence, how to design such a holistic model of performance overhead among multiple VMs becomes a promising research problem. One potential modeling solution is to extend the discrete-time model of CPU resource in [39], i.e., Eq. (7), to support multi-dimensional server resources. However, this method requires learning the relationship between the current performance data and the historical resource consumption of VMs for the model construction. The overhead (e.g., CPU cycles) caused by such a learning phase cannot be neglected. Hence, it would be also interesting to mitigate such an overhead incurred by the learning phase for constructing this model.

Fourth, to quantify the performance overhead of VMs caused by live VM migration, existing modeling methods focus on the performance of either the migrating VM or co-located VMs on migration source and destination servers (refer to Table 3). Accordingly, it would be desirable to design a prediction model that jointly quantifies the performance overhead of the migrating VM and its co-located VMs, in order to comprehensively evaluate the performance overhead caused by the live VM migration process [47]. Furthermore, as the multi-tier application that requires multiple interdependent VMs are widely deployed in datacenters [22], the migration of a VM (e.g., the SQL server) can adversely impact the performance of other associated VMs (e.g., the Web server). Hence, the evaluation of performance overhead of live VM migration cannot be restricted to the single migrating VM (i.e., [45]). The cascading performance effects to other associated VMs should be incorporated into the model of VM performance overhead, particularly for evaluating multi-tier VM applications.

Last but not least, although the performance of live VM migration has been comprehensively studied and modeled (e.g., [43]) in the LAN environment, the performance model of live VM migration over the WAN has not been investigated yet. Such a performance model would be beneficial for the tenant to utilize the computing resources in the hybrid cloud (i.e., public and private clouds across multiple geo-distributed datacenters). For example, the migration performance model in the WAN is effective in improving the performance of migrating VM and the migration itself, by selecting an appropriate migrating VM and a migration destination datacenter. However, the performance model of live VM migration in the LAN cannot be directly applied in the WAN environment due to at least two reasons: (1) the network link bandwidth can vary significantly over time in the WAN, and (2) the VM disk image is also required to be transferred over the WAN. Consequently, how to develop a performance model of live WAN migration of VMs becomes another future research problem.

4 MANAGING PERFORMANCE OVERHEAD OF VMs IN IaaS CLOUD

A large number of approaches have recently been proposed to manage VM performance overhead and guarantee the performance of VMs in the IaaS cloud. We broadly classify and summarize these solutions according to the high-level hierarchy of IaaS cloud datacenters shown in Fig. 1. We compare these solutions on single server virtualization, within a single mega datacenter, and across multiple geo-distributed datacenters, in the following three subsections, respectively. These overhead mitigation techniques are summarized in Table 4 – 8. In particular, the table column of complexity represents the implementation complexity of techniques introduced to the original systems (e.g., Hypervisors, shared storage and networks). It also shows the time complexity of VM assignment algorithms in a number of overhead mitigation techniques (e.g., PSciMapper [19], iAware [47], Pesto [48] and Oktopus [49]). The time complexity notations m and n denote the number of physical servers and virtual machines in a datacenter, respectively.

4.1 Alleviating VM Performance Overhead Caused by Single Server Virtualization

Currently, Hypervisors [9] such as Citrix XenServer and Microsoft Hyper-V, provides a baseline of performance isolation on CPU, memory and disk resources by simply allocating each VM (including domain-0) with dedicated CPU core(s) and an amount of non-overlapped memory and disk capacities. However, the cache and I/O bandwidth resources shared among multiple co-located VMs shown in Fig. 3, cannot be easily isolated in existing Hypervisors, as we have discussed in Sec. 2.1. To deal with such a performance issue, a number of approaches have been proposed to mitigate the performance overhead of VMs caused by the single server virtualization. We summarize these solutions in Table 4.

### 4.1.1 Resource Isolation among Co-located VMs

To alleviate performance overhead of VMs on a single physical server, one conventional approach is to isolate the computing resources among co-located VMs. To mitigate the performance interference on the CPU cache...
and memory bandwidth, Q-Clouds [39] dynamically provisions the underutilized or idle CPU resource to the impacted (victim) VMs using closed loop resource management. Accordingly, the workload SLAs can be guaranteed and the datacenter utilization is improved. Page coloring mechanism\(^2\) assigns the virtual memory pages and physical cache pages with different colors, where the virtual memory page can only be allocated to the physical cache page with the same color. Such a technique can accordingly be used to partition the cache space and control the contention on the cache resource among co-running applications [50]. However, whether page coloring can alleviate the performance interference on the cache and memory bandwidth resource among co-located VMs requires to be further investigated.

To control the disk I/O contention among co-located VMs on a physical server, one possible solution is to leverage the disk I/O scheduling technique on the VM and VMM level. For example, Boucher et al. [51] empirically show that the aggregate amount of I/O throughput of co-located VMs can significantly be improved, by choosing the appropriate disk I/O schedulers of VM and VMM from the existing Linux I/O schedulers. In particular, the disk I/O fairness among co-located VMs can only be achieved at the cost of I/O throughput and latency. To improve the response time for I/O requests with a high priority, a priority-based scheduler has been proposed in [52]. This scheduler intercepts I/O requests at the VM level, and re-orders these requests in the disk I/O queue of a physical server according to their priority information at the VMM level.

The performance interference on network bandwidth among co-located VMs is managed through two kinds of solutions in existing studies. On one hand, it is reported that the network bandwidth of VMs can be isolated from the other co-located VMs by hardware solutions, such as installing multiple network adapters or multi-queue network adapters for the VMs on a physical server [53]. On the other hand, the network bandwidth of VMs can be guaranteed by software solutions (i.e., bandwidth capping techniques), via statically setting a bandwidth cap in the VM configuration file before booting the VM [8], or limiting the network bandwidth using the “tc” utility in the Linux kernel [6]. To allow scalable and dynamic programmable rate limiting with low CPU overhead, NicPic [54] classifies and stores the network packets in transmit queues in memory, and schedules the packets from memory to NIC using direct memory access according to the rate limits of VMs.

### 4.1.2 Optimization of VM Assignment

In addition to the resource isolation techniques, it is also effective in alleviating VM performance overhead by finding an optimal mapping of VMs to physical servers. Specifically, by greedily minimizing the VM performance overhead calculated by the model introduced in Eq. (3), (4), or (5), such an optimized VM assignment\(^3\) can accordingly minimize the performance interference among

<table>
<thead>
<tr>
<th>Approaches</th>
<th>Resources</th>
<th>Optimization</th>
<th>Complexity</th>
<th>Techniques</th>
<th>Effectiveness</th>
</tr>
</thead>
<tbody>
<tr>
<td>Q-Clouds [39]</td>
<td>CPU cache</td>
<td>Performance</td>
<td>Medium</td>
<td>CPU core compensation</td>
<td>CPU utilization improvement: 35%</td>
</tr>
<tr>
<td>Zhang et al. [50]</td>
<td>Disk I/O</td>
<td>Performance, fairness</td>
<td>Low</td>
<td>VMM- and VM-level I/O schedulers</td>
<td>I/O throughput improvement: 72%</td>
</tr>
<tr>
<td>Boucher et al. [51]</td>
<td></td>
<td>Performance</td>
<td>Low</td>
<td>Priority-based I/O scheduler</td>
<td>Response time improvement: 200%</td>
</tr>
<tr>
<td>Blagojevic et al. [52]</td>
<td>Network I/O</td>
<td>Performance</td>
<td>Low</td>
<td>Multiple or multi-queue network adapters</td>
<td>Guarantee I/O throughput</td>
</tr>
<tr>
<td>Hardware [53]</td>
<td></td>
<td>Performance, scalability</td>
<td>Low</td>
<td>Bandwidth capping</td>
<td>Mitigate I/O interference</td>
</tr>
<tr>
<td>Xen manual [8]</td>
<td></td>
<td>Performance</td>
<td>Medium</td>
<td>Interference calculation, VM assignment</td>
<td>Application completion time reduction: 50%</td>
</tr>
<tr>
<td>Barker et al. [6]</td>
<td></td>
<td>Performance</td>
<td>Medium</td>
<td>Cache pressure, VM assignment</td>
<td>Balance tradeoff between performance and energy</td>
</tr>
<tr>
<td>NicPic [54]</td>
<td></td>
<td>Performance, energy</td>
<td>Medium</td>
<td>Metrics correlation, VM assignment</td>
<td>Power reduction: 56%</td>
</tr>
<tr>
<td>Zhu et al. [35]</td>
<td></td>
<td>CPU, memory, disk, network</td>
<td>Performance, energy</td>
<td>Cache pressure, VM assignment</td>
<td>Performance improvement: 17%</td>
</tr>
<tr>
<td>TRACON [37]</td>
<td></td>
<td>CPU cache</td>
<td>Performance</td>
<td>Medium, (O(m \cdot n))</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>pSciMapper [19]</td>
<td></td>
<td>CPU, memory, disk, network</td>
<td>Performance</td>
<td>Medium, (O(n^2))</td>
<td></td>
</tr>
<tr>
<td>Ahn et al. [55]</td>
<td></td>
<td>CPU cache</td>
<td>Performance</td>
<td>Medium, (O(m \cdot n))</td>
<td>VM assignment</td>
</tr>
<tr>
<td>DeepDive [10]</td>
<td></td>
<td>CPU, memory, disk, network</td>
<td>Performance</td>
<td>Low, (O(n))</td>
<td>Identify and mitigate VM interference with low overhead</td>
</tr>
<tr>
<td>Bobtail [56]</td>
<td>Network I/O</td>
<td>Performance</td>
<td>Low, (O(m \cdot n))</td>
<td>VM examining and assignment</td>
<td>Network I/O latency reduction: 50 – 65%</td>
</tr>
</tbody>
</table>

\(^2\) Page coloring mechanism is the traditional cache and memory management technique in the Operating System.

\(^3\) VM assignment decides that which VM is assigned to which physical server. It requires the live migration of VMs.
co-located VMs [35], [37]. As more physical servers are provisioned, the performance interference of VMs will be reduced while leading to higher energy consumption. Accordingly, to jointly balance the tradeoff between energy consumption and VM performance, Cuanta [16] and pSciMapper [19] determine the VM assignment and the number of physical servers, by greedily maximizing the ratio of VM performance to VM power. A similar performance-aware VM assignment mechanism has been proposed in [55], where the cloud scheduler greedily alleviates the VM contention on Last-Level Cache (LLC) in the entire cloud system. Specifically, for the physical server with maximum LLC misses and the physical server with minimum LLC misses, this approach iteratively swaps the VM with maximum LLC misses on the former server and the VM with minimum LLC misses on the latter server, via live migration of VMs.

To efficiently identify and manage the performance interference of VMs, DeepDive [10] first qualitatively pinpoints culprit resource (i.e., the resource which is the source of interference) using a clustering technique of low-level VM metrics. Then, this approach migrates the VM consuming the culprit resource most aggressively to other servers, so that the VM interference is sufficiently mitigated or totally eliminated. With a particular focus on the network I/O resource, Xu et al. [56] observe that the co-scheduling of CPU-bound and latency-sensitive applications on a physical server can cause long I/O latency to VMs. Based on such an observation, they further design a simple VM examining method, which assigns latency-sensitive applications to good-performing VMs, avoiding the co-location with bad-neighboring VMs.

4.2 Mitigating VM Performance Overhead within a Single Datacenter

As analyzed in Sec. 2.2, the routine operations of an IaaS cloud datacenter including the (concurrent) live migration, deployment and snapshotting of multiple VMs, and the contention of multiple VMs on the shared datacenter network and storage resources, can severely degrade the performance of VMs. We review and summarize the existing approaches that mitigate the performance overhead of VMs within a single datacenter in Table 5, Table 6 and Table 7, according to the aforementioned three aspects, respectively.

4.2.1 Live Migration, Deployment and Snapshotting of Multiple VMs

Live Migration of VMs. The migration process causes different aspects of performance overhead to VMs in terms of migration time, application downtime, network traffic, and energy consumption [43]. To alleviate such performance overhead, one intuitive method is to reduce the amount of transferred data during the live migration of VMs. For example, by taking advantage of plenty multi-core CPU resource, MECOM [57] utilizes the traditional data compression technique to significantly reduce the amount of transferred memory pages on the migration source server. Yet, this method requires decompressing those transferred memory pages to restore VM memory state on the migration destination server.

To further reduce the network traffic during live migration of VMs, several practical approaches have been proposed in existing studies (e.g., [58], [60], [61]). Specifically, instead of transferring the entire dirty memory pages in each round of pre-copy [21] phase, (1) the delta compression technique [60] compresses and transfers the data changes between the current and previous version of memory pages, particularly for the applications with a fast dirty rate of memory pages over a slow network. (2) The memory “pruning” approach [58], [59] identifies and transfers the necessary VM memory pages which are mandatory for the VM to run correctly on the destination server after the VM migration, such as the application data and OS kernel data. (3) CR/RT-Motion [61] first transfers a small amount of logs for the VM execution events traced on the migration source server. Then, the VM memory state can be restored by replying the execution trace logs on the migration destination server. (4) Jo et al. [62] only transfer the memory pages that are not available on the shared storage, along with a list of storage blocks with the memory page locations, in order to restore the VM memory state on the destination server.

While the approaches above focus on reducing the amount of transferred data to improve the performance of live VM migration itself, PMigrate [63] aims to parallelize the data and processes of the migration operation using abundant CPU and network resources, in order to accelerate the live migration of VMs. Different from these approaches above, Liu et al. [43] alleviate the performance overhead of the migrating VMs by selecting the VM with the smallest migration cost for migration. In particular, the migration cost is quantified by the migration performance model described in Eq. (10).

Furthermore, the migration process also deteriorates the performance of VMs (including the migrating VM) on the migration source and destination servers [47]. As shown in Fig. 4, the migration traffic interferes with the application traffic among the co-located VMs on both migration source and destination servers. In addition, the migration process consumes a certain amount of CPU cycles in domain-0 [44]. (1) To mitigate the performance overhead on the migrating VM, CloudScale [41] chooses to migrate the VM with the smallest SLO penalty...
which is measured offline for each VM application. (2) To further alleviate the performance overhead of VMs on migration source and destination servers, iAware [47] chooses to migrate the VM with the least performance interference among the co-located and migrating VMs. In particular, the interference is predicted online by a multi-resource demand-supply model. With a particular focus on the performance of migrating VM and the migration itself, Breitgand et al. [45] develop a mathematical model to learn the tradeoff between the migration time and VM performance, as described in Eq. (12). By jointly minimizing the migration time and improving the performance of migrating VM, this approach allocates an appropriate network bandwidth to the live VM migration.

Given the same VM candidates for live migration, it has been shown that the performance of concurrent live migration of these VMs is much worse than that of sequential live migration of these VMs [44]. To mitigate the migration performance overhead (i.e., reducing the network traffic and migration time), the approach proposed in [24] still focuses on reducing the amount of transferred data. Specifically, this method de-duplicates the memory pages among co-located VMs and transfers the identical memory pages only once, during the concurrent live migration of the VMs hosted on a physical server.

### Deployment and Snapshotting of VMs

To reduce the deployment time of multiple VMs, traditional approaches adopt the Peer-to-Peer (P2P) technique, which allows the VMs booting from the same image to share the content of VM image among themselves [74]. However, it cannot mitigate the potential VM performance overhead caused by a significant amount of network traffic due to the concurrent VM deployment. To mitigate such a traffic overhead, VDN [64] proposes a VM image distribution network based on the cross-image duplication characteristic, i.e., the similarity among real-world VM disk images (e.g., VMware images and Amazon machine images) can be as high as 60% [75]. Specifically, this method divides VM images into disjoint chunks, and allows the VMs to share these chunks in a topology-aware network during provisioning these VMs.

As observed that only a small part of the VM image is required to be loaded for the VM startup [65], the booting part of VM images can be cached in compute nodes, in order to reduce the VM booting time and the traffic load to datacenter networks. In addition, a lazy VM deployment mechanism has been proposed in [25], where the VM image blocks are fetched as required by

<table>
<thead>
<tr>
<th>Approaches</th>
<th>Objective</th>
<th>Optimization</th>
<th>Complexity</th>
<th>Techniques</th>
<th>Effectiveness</th>
</tr>
</thead>
<tbody>
<tr>
<td>MECOM [57]</td>
<td>Live migration of a VM</td>
<td>Migration time, application downtime, network traffic</td>
<td>Medium</td>
<td>Compression</td>
<td>Migration time reduction: 32%</td>
</tr>
<tr>
<td>Koto et al. [58]</td>
<td></td>
<td></td>
<td>Low</td>
<td>Filter un-useful memory pages</td>
<td>Migration time reduction: 34 – 68.3%</td>
</tr>
<tr>
<td>Chiang et al. [59]</td>
<td></td>
<td></td>
<td>Medium</td>
<td>Compression, page delta transfer</td>
<td>Migration time reduction: 40%</td>
</tr>
<tr>
<td>Svard et al. [60]</td>
<td></td>
<td></td>
<td>Medium</td>
<td>Transfer and replay of execution logs</td>
<td>Migration time reduction: 31.5%</td>
</tr>
<tr>
<td>CR/RT -Motion [61]</td>
<td></td>
<td></td>
<td>Low</td>
<td>Transfer storage locations of memory</td>
<td>Migration time reduction: over 30%</td>
</tr>
<tr>
<td>Jo et al. [62]</td>
<td></td>
<td></td>
<td>Low</td>
<td>Performance of migration and the migrating VM</td>
<td>Migration speedup factor: 2.49 – 9.88</td>
</tr>
<tr>
<td>PMigrate [63]</td>
<td></td>
<td>Migration performance, energy</td>
<td>Low</td>
<td>Parallelizing migration</td>
<td>Migration cost reduction: 72.9%</td>
</tr>
<tr>
<td>Liu et al. [43]</td>
<td></td>
<td>Performance of migration and the migrating VM</td>
<td>Medium, $O(n)$</td>
<td>Migrate the VM with minimum cost</td>
<td>SLO violation reduction: 83%</td>
</tr>
<tr>
<td>Breitgand et al. [45]</td>
<td></td>
<td>Performance of migration and the migrating VM</td>
<td>Low</td>
<td>Migration bandwidth allocation</td>
<td>Performance improved: 16 – 65%</td>
</tr>
<tr>
<td>CloudScale [41]</td>
<td></td>
<td>Performance of co-located VMs</td>
<td>Low, $O(mn)$</td>
<td>Migrate the VM with minimum cost on VM performance</td>
<td>Network traffic reduction: 75%</td>
</tr>
<tr>
<td>iAware [47]</td>
<td></td>
<td>Performance of VMs on source and destination</td>
<td>Low, $O(mn)$</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Deshpande et al. [24]</td>
<td>Concurrent deployment and snapshotting of multiple VMs</td>
<td>Concurrent migration time, network traffic</td>
<td>Medium</td>
<td>De-duplication</td>
<td></td>
</tr>
<tr>
<td>Razavi et al. [65]</td>
<td></td>
<td>Performance of VM deployment</td>
<td>Low</td>
<td>VM image caching</td>
<td>Speedup factor: 30 – 80</td>
</tr>
<tr>
<td>Nicolae et al. [25]</td>
<td></td>
<td>Performance of VM deployment</td>
<td>Medium</td>
<td>Lazy VM deployment, store incremental update</td>
<td>Speedup factor: 2 – 25, bandwidth reduction: 90%</td>
</tr>
<tr>
<td>VMTorrent [26]</td>
<td></td>
<td>Scalability, VM booting delay</td>
<td>Medium</td>
<td>VM image profile, block pre-fetching</td>
<td>Speedup factor: 30</td>
</tr>
</tbody>
</table>

TABLE 5: Comparison of Approaches to Manage VM Performance Overhead Caused by Live Migration, Deployment and Snapshotting of Multiple VMs within a Single Datacenter
the execution of applications, so as to avoid transferring and loading the unnecessary image data to VMs. VM-Torrent [26] further refines this lazy deployment method by pre-fetching the image data blocks, according to the offline profile information of data block accessing for each specific pair of a workload and a VM image. To reduce the network traffic caused by the concurrent VM snapshotting, Nicolae et al. [25] propose to save the image difference between the current image snapshot and the former-stored VM image to the persistent storage, rather than the snapshot of a whole image file.

### 4.2.2 Contention of Multiple VMs on Shared Storage Resources

Different from the disk I/O sharing among co-located VMs on a single physical server discussed in Sec. 4.1, shared cloud storage service deploys an entire storage system, such as Network Attach Storage (NAS) and Storage Area Network (SAN), to support I/O requests from multiple VMs hosted on multiple physical servers, as illustrated in Fig. 5. To isolate the storage I/O resource among multiple VMs, the straightforward approach is to well manage and schedule the requests in I/O queues on each physical server and shared storage system. For example, PARDA [66] first controls the length of I/O queue on each physical server based on the aggregated weights of the hosted VMs. Then, this method uses a local fair I/O scheduler to achieve proportional-share fairness of storage I/O resource among co-located VMs within each physical server.

In addition to the proportional share of storage I/O resource, SRP [67] further provides several other storage resource controls, such as the minimum guarantee and limits of storage I/O bandwidth for an individual VM or a group of VMs. In particular, the I/O reservations, limits, and proportional shares of I/O queue capacity for each VM and physical server can be determined by periodically predicting the workload demands. With a particular focus on the replicated key-value shared storage, Pisces [68] deploys a series of novel mechanisms to achieve the performance isolation and max-min fairness among tenants (rather than VMs). Specifically, these mechanisms includes placing and replicating the data blocks across multiple storage devices based on the weights of tenants, as well as scheduling VM I/O requests to storage devices using the Deficit (Weighted) Round Robin algorithm. To alleviate the interference on the shared cache of storage arrays among VMs, Tarasov et al. [69] fairly allocate the storage cache space according to the classification of VM I/O operations and data. The data prefetching in the storage cache is further applied to improve the I/O throughput of shared storage.

While the former approaches provide the isolation on sharing storage I/O resource among multiple VMs or tenants, they cannot guarantee the SLO of VM applications, such as the latency and throughput of storage I/O service, due to the storage I/O contention. To achieve this goal, Soundararajan et al. [70] first translate the SLO into the storage I/O allocation by a reinforcement learning algorithm. Then, this method dynamically partitions the I/O bandwidth based on the translated storage I/O

---

**TABLE 6: Comparison of Approaches to Mitigate Contention of VMs on Shared Storage Resource of a Single Datacenter**

<table>
<thead>
<tr>
<th>Approaches</th>
<th>Objective</th>
<th>Optimization</th>
<th>Complexity</th>
<th>Techniques</th>
<th>Effectiveness</th>
</tr>
</thead>
<tbody>
<tr>
<td>PARDA [66]</td>
<td>Storage I/O isolation and fairness</td>
<td>Weighted sharing</td>
<td>Medium</td>
<td>I/O queue control, fair I/O scheduler</td>
<td>Guarantee I/O performance, fairness</td>
</tr>
<tr>
<td>SRP [67]</td>
<td></td>
<td>Minimum guarantee, limits, weighted sharing</td>
<td>Medium</td>
<td>I/O queue control, demand prediction</td>
<td>max-min ratio: 0.99, I/O throughput improvement: 20%</td>
</tr>
<tr>
<td>Pisces [68]</td>
<td></td>
<td>Weighted sharing, high utilization</td>
<td>Medium</td>
<td>I/O queue control, data placement, replica selection</td>
<td>I/O throughput improvement: 11 – 95%</td>
</tr>
<tr>
<td>Tarasov et al. [69]</td>
<td>Storage cache allocation</td>
<td>I/O classification, data pre-fetching</td>
<td>Medium</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Soundararajan et al. [70]</td>
<td>SLO guarantee of storage I/O service</td>
<td>I/O bandwidth allocation</td>
<td>Low</td>
<td>I/O bandwidth partitioning</td>
<td>Minimize SLO violation of I/O service</td>
</tr>
<tr>
<td>Horizon [71], QBox [72]</td>
<td>Storage I/O requests scheduling</td>
<td></td>
<td>Low</td>
<td>I/O requests re-ordering</td>
<td>92 – 99% deadlines of I/O requests are met</td>
</tr>
<tr>
<td>Pesto [48]</td>
<td>Storage I/O hotspot mitigation</td>
<td>I/O Load balancing, migration cost, benefit</td>
<td>Medium, $O(m \cdot n)$</td>
<td>VM assignment, I/O performance modeling</td>
<td>Peak I/O latency reduction: 19%</td>
</tr>
<tr>
<td>Romano [73]</td>
<td>Migration benefit, storage I/O interference</td>
<td></td>
<td>High, $O(n^2)$</td>
<td></td>
<td>Average I/O latency reduction: 58%</td>
</tr>
</tbody>
</table>

**Fig. 5: Typical shared storage architecture in the datacenter.**
allocation using I/O schedulers on physical servers and shared storage. Two SLO-oriented storage I/O scheduling approaches have been proposed in [71], [72], where the high-level SLO of VM application is translated into a series of deadlines for application I/O requests based on the historical I/O statistics. These I/O requests can be completed before their specific translated deadlines, by re-ordering them in the storage I/O queue using the Early Deadline First (EDF) scheduling algorithm.

To mitigate the disk I/O contention in the cloud storage, another practical approach is to leverage the live migration of virtual disks with heavy I/O requests. For example, Pesto [48] first constructs an empirical linear model of latency and throughput of I/O requests in the shared storage. Then, such a model is leveraged to perform I/O load balancing with a cost-benefit analysis for all migration choices of virtual disks, so as to alleviate the I/O performance variation, i.e., Eq. (2). As compared with the former method, a more accurate I/O performance model has been developed in [73], which uses statistical modeling approaches (i.e., the analysis of variance and linear regression) and takes the interference of co-located virtual disks into account. Based on such a model, Romano [73] designs an algorithm to find the global pseudo-optimal mapping of VMs to physical servers using the simulated annealing approach, to reduce the variation of storage I/O latency in datacenters.

4.2.3 Contention of Multiple VMs on Shared Network Resources

Currently, several commercial cloud providers (e.g., Amazon EC2 [2]) cannot provide the performance guarantee of network resource for tenants (or VMs). As discussed in Sec. 2.2, the network bandwidth and latency between two VMs can vary significantly over time. Fortunately, the issue of network performance isolation among tenant VMs in an IaaS cloud datacenter has been extensively studied by recent research, which can be broadly classified into two categories: weighted competition sharing and VM assignment based sharing.

**Weighted Competition Sharing Approach.** The weighted competition approach is to share the network resource among competing entities (i.e., processes, VMs or tenants) according to their weights, as illustrated in Fig. 6. In particular, the weights of entities can be assigned by cloud providers [81] or according to minimum bandwidth guarantees [76] and tenant payments [77]. A typical example of this approach has been proposed in Seawall [28], where the network bandwidth on the contention links is fairly allocated among multiple data senders using a VMM-based rate limiter (e.g., NicPic [54]) according to their assigned weights. However, this approach fails to offer minimum bandwidth guarantees to VMs or tenants.

To support minimum guarantees of network bandwidth and share the network resource among VMs in a fine-grained timescale (i.e., milliseconds), EyeQ [76] proposes an end-to-end rate control mechanism along the data transmission path, by implementing rate limiters on the data sender and congestion detectors on the data receiver, respectively. This approach also achieves high utilization of the cloud network resource. To particularly provide bandwidth guarantees for inter-tenant communication, Hadrian [77] allocates the network bandwidth to inter-tenant flows using the rate limiters on data senders, receivers and weighted fair queuing in switches. As this approach requires traffic shaping (e.g., vShape [86]) in network switches, the implementation complexity is increased accordingly. ElasticSwitch [78] allocates the pairwise VM-to-VM bandwidth based on the network bandwidth guarantees of VMs, and distributes the unused bandwidth to VMs according to their weights. To reduce the implementation overhead and improve the scalability, this approach is fully implemented in Hypervisors and without centralized coordination.

By proposing and comparing three bandwidth allocation policies: Proportional Sharing at the Link level (PS-L), at the Network level (PS-N), and on the Proximate link (PS-P), FairCloud [79] experimentally explores the tradeoff space among three requirements for weighted bandwidth sharing of datacenter networks, including minimum guarantee, network proportionality, and high utilization. Specifically, the minimum guarantee and network proportionality cannot be achieved simultaneously (i.e., hard tradeoff), and there exists a tradeoff between the network proportionality and high utilization. To obtain a theoretical insight into the tradeoff between minimum guarantee and proportional bandwidth share, Guo et al. [80], [81] model the bandwidth allocation as a Nash bargaining game [87], and design a distributed algorithm named Falloc to fairly allocate bandwidth to VMs in a cooperative manner.

In addition to achieving the weighted sharing and minimum guarantees of VM network bandwidth, two approaches [82], [83] further provide another Quality of Service (QoS) control for VM bandwidth allocation, i.e., limits of VM bandwidth. Specifically, Gatekeeper [82] guarantees the network bandwidth of VMs by implementing a weighted packet scheduler on data senders, and determining the bandwidth allocation according to the minimum and maximum rates on data receivers. In particular, this method focuses on guaranteeing the aggregate end-to-end network bandwidth of VMs for

![Fig. 6: Network bandwidth sharing among VMs according to their respective weights.](image-url)
TABLE 7: Comparison of Approaches to Mitigate Contention of VMs on Shared Network Resource of a Single Datacenter

<table>
<thead>
<tr>
<th>Approaches</th>
<th>Hierarchy</th>
<th>Optimization</th>
<th>Complexity</th>
<th>Techniques</th>
<th>Overhead</th>
<th>Implementation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Seawall [28]</td>
<td>VM/process</td>
<td>Weighted sharing</td>
<td>Low</td>
<td>Rate limiting at sender</td>
<td>3.6% CPU cycles</td>
<td>Hypervisor</td>
</tr>
<tr>
<td>EyeQ [76]</td>
<td>VM</td>
<td>Minimum guarantee, weighted sharing</td>
<td>Low</td>
<td>Rate limiting at sender, receiver</td>
<td>3 – 20% CPU cycles</td>
<td>Hypervisor</td>
</tr>
<tr>
<td>Hadrian [77]</td>
<td>Tenant</td>
<td>Minimum guarantee, weighted sharing</td>
<td>Medium</td>
<td>Rate limiting at sender, receiver, switch</td>
<td>4% link traffic</td>
<td>Switch, Hypervisor</td>
</tr>
<tr>
<td>ElasticSwitch</td>
<td>VM</td>
<td>Minimum guarantee, weighted sharing, scalability</td>
<td>Low</td>
<td>Guarantee partition, rate allocation</td>
<td>10% CPU cycles and a few Kbps</td>
<td>Hypervisor</td>
</tr>
<tr>
<td>FairCloud [79]</td>
<td>VM/tenant</td>
<td>Minimum guarantee, weighted sharing, high utilization</td>
<td>High</td>
<td>Weighted fair packet queuing at switch</td>
<td>N/A</td>
<td>Switch, Hypervisor</td>
</tr>
<tr>
<td>Guo et al. [80], [81]</td>
<td>VM</td>
<td>Minimum guarantee, weighted sharing, fairness</td>
<td>Medium</td>
<td>Nash bargaining game analysis</td>
<td>N/A</td>
<td>Switch</td>
</tr>
<tr>
<td>Gatekeeper [82]</td>
<td>Tenant</td>
<td>Minimum guarantee, limits, weighted sharing</td>
<td>Medium</td>
<td>Rate limiting at sender</td>
<td>10% CPU cycles</td>
<td>Hypervisor</td>
</tr>
<tr>
<td>hClock [83]</td>
<td>VM/tenant</td>
<td>Minimum guarantee, limits, weighted sharing</td>
<td>Low</td>
<td>I/O traffic queue scheduling and billing</td>
<td>Within 5% CPU cycles</td>
<td>Hypervisor</td>
</tr>
<tr>
<td>Octopus [49]</td>
<td>VM</td>
<td>Homogeneous and static bandwidth</td>
<td>Low</td>
<td>Static allocation, VM assignment</td>
<td>CPU, network: low</td>
<td>Hypervisor</td>
</tr>
<tr>
<td>TIVC [84]</td>
<td>VM</td>
<td>Time-varying bandwidth</td>
<td>High</td>
<td>Profiling demand, VM assignment</td>
<td>CPU, network: moderate</td>
<td>Switch, router</td>
</tr>
<tr>
<td>Zhu et al. [85]</td>
<td>VM</td>
<td>Heterogeneous bandwidth</td>
<td>O(n·log n)</td>
<td>VM assignment</td>
<td>N/A</td>
<td>N/A</td>
</tr>
</tbody>
</table>

tenants in datacenters. hClock [83] allocates VM bandwidth in a hierarchical manner through maintaining and scheduling I/O requests in the leaf queue of VMs. In more detail, the requests are scheduled by periodically checking the minimum guarantees, limits and weights of VMs, and the scheduling of I/O requests is started at the root node of the leaf queue in the hierarchy.

VM Assignment Based Sharing Approach. In respond to the VM leasing request from tenants, the VM assignment based approach allocates the network resource to tenant VMs and meets their bandwidth requirements, by assigning these VMs to an appropriate network location (physical server). A representative example of this approach has been presented in Octopus [49], where the VM leasing request is simply abstracted to a tuple \((N, B)\) (i.e., the number of VMs \(N\) with a homogenous network bandwidth \(B\)). According to the network abstractions of VM leasing requests, the network resource is statically allocated to tenant VMs, by greedily assigning these VMs to the smallest sub-tree (i.e., a server, a rack, a pod) that can host them.

To break the assumption of static and homogeneous network bandwidth \(B\) of the VM leasing request, two approaches [84], [85] extend the basic network abstraction model \((N, B)\) to more realistic models, where tenants are allowed to specify time-varying (i.e., temporal) and heterogeneous (i.e., spatial) allocations of network bandwidth for their leased VMs, respectively. The objective of VM allocations in these two approaches is to improve the utilization of network resource, so as to allocate more tenant VMs in a datacenter. Specifically, by profiling the network bandwidth demands offline for each VM application, TIVC [84] fits the leased VMs into four types of network abstractions with the time-varying bandwidth. According to the fitted abstraction models of these VMs, this method then assigns them to the datacenter by a first-fit VM assignment algorithm. Zhu et al. [85] propose an abstraction model of heterogeneous VM network bandwidth allocations, and mathematically prove the problem of VM allocations with heterogeneous bandwidth demands to be NP-complete. In addition, an online heuristic VM assignment algorithm is developed to allocate the leased VMs with heterogeneous bandwidth demands from multiple tenants in a datacenter.

4.3 Controlling VM Performance Overhead across Multiple Geo-distributed Datacenters

The live migration of VMs or storage over the WAN is a common tool to manage computing resources of multiple geo-distributed datacenters (e.g., load balancing among datacenters and disaster recovery of IT systems). However, it brings noticeable performance overhead of VMs in IaaS clouds, as we have discussed in Sec. 2.3. In this subsection, we review and compare the solutions that alleviate the performance overhead of VMs caused by the live WAN migration in Table 8.

![Fig. 7: Live WAN migration of a single VM across multiple geo-distributed datacenters.](image-url)
### 4.3.1 Live WAN Migration of VMs

As illustrated in Fig. 7, the live WAN migration of VMs requires transferring the VM disk image and memory state as well as the on-going network connections via the WAN, which is more complicated than the live VM migration in a LAN. Bradford et al. [33] make the first attempt to migrate a single VM hosting a running Web server over the WAN. Specifically, the disk image and VM memory state are iteratively copied to the destination to complete the whole live VM migration. To particularly deal with the migrating VM hosting write-intensive workloads, a write throttling mechanism is proposed. During transferring the disk or memory state, write throttling reduces the number of dirty pages by restricting the number of write operations below a predefined threshold. Furthermore, the existing network connections of the migrating VM can be transparently redirected to the VM migration destination server by the IP tunneling technique, which dynamically routes the network packets to the new IP address of the migrating VM. To accelerate the migration of data-intensive applications over the WAN, Akiyama et al. [88] propose page cache teleportation which reduces the amount of transferred memory for VM page cache. In particular, the page cache of the migrating VM can be restored on the destination server based on its memory location which is identified before the migration.

To reduce the migration time and network traffic caused by the live WAN migration of multiple VMs, one straightforward yet effective approach is de-duplication. Specifically, this approach explores the duplicate contents among the disk images and VM memory states of the migrating VMs, and transfer only one copy of them [32], [89]. To particularly reduce the application downtime of the migrated VMs, VMFlockMS [89] adopts the data prioritization transfer mechanism, by transferring the data that is responsible for booting the VM and running application services with the highest priority. With the same objective, CloudNet [32] proposes a page delta approach which only sends the difference between the current memory page and the page previously transferred, thereby reducing the rounds of memory pre-copy and the number of dirty pages. Generally, the migration cost of transferring the VM disk image is much higher than that of transferring the VM memory state. To avoid such a high migration cost of disk image transfer, one speculative approach [32] is to replicate the VM images in the background before the live migration of VMs, and synchronize the disk state during the live migration.

### 4.3.2 Live WAN Migration of VM Storage

The live migration of VM storage (i.e., one or more VM disk images) is the critical part of live migration of VMs over the WAN, which has recently attracted much attention of researchers. In order to guarantee a minimal application downtime, Mashtizadeh et al. [90] develop a technique named I/O mirroring in VMWare ESX. During the transfer of VM disk image, I/O mirroring blocks the write operations on the migration source server until these operations are finished on the destination server. Hirofuchi et al. [91] propose to fetch the required data blocks on demand from the migration source server, and transfer the remaining disk data blocks to the destination server using the available bandwidth of the WAN. However, these two approaches above leads to high disk I/O latency and low I/O throughput under I/O-intensive workloads. To particularly deal with the I/O-intensive workloads, Nicolae et al. [92] further optimize the migration strategy by two mechanisms: (1) transferring only the modified contents of VM disk image except the VM Operating System to the migration destination server; (2) a hybrid data fetching technique, in which the migration source server pre-copies the cold data chunks when transfer control is on the source server, and the migration destination server pulls the remaining data chunks from the source server when transfer control is on the destination server.

A workload-aware scheduling approach for storage

---

<table>
<thead>
<tr>
<th>Approaches</th>
<th>Objective</th>
<th>Optimization</th>
<th>Complexity</th>
<th>Techniques</th>
<th>Effectiveness</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bradford et al. [33]</td>
<td>Live WAN migration of a VM</td>
<td>QoS of the migrating VM</td>
<td>Low</td>
<td>Write throttling, IP tunneling</td>
<td>Migrate a running Web server in the WAN</td>
</tr>
<tr>
<td>Akiyama et al. [88]</td>
<td>Live WAN migration of a data-intensive VM</td>
<td>Migration time</td>
<td>Low</td>
<td>Page cache restoring</td>
<td>Migration speedup: 3.5</td>
</tr>
<tr>
<td>VMFlockMS [89]</td>
<td>Live WAN migration of multiple VMs</td>
<td>Migration time, downtime</td>
<td>Medium</td>
<td>De-duplication, data priority transfer</td>
<td>Migration time reduction: 65%</td>
</tr>
<tr>
<td>CloudNet [32]</td>
<td>Storage migration in a LAN or WAN</td>
<td>Migration time, downtime, traffic</td>
<td>High</td>
<td>De-duplication, page delta transfer</td>
<td>Migration time reduction: 65%</td>
</tr>
<tr>
<td>Mashtizadeh et al. [90]</td>
<td>Storage migration over the WAN</td>
<td>Migration time</td>
<td>Medium</td>
<td>I/O mirroring</td>
<td>Application downtime: nearly zero</td>
</tr>
<tr>
<td>Nicolae et al. [92]</td>
<td>Migration time, network traffic</td>
<td>Migration time</td>
<td>Medium</td>
<td>Modified data transfer, data pre-fetching</td>
<td>Migration speedup: 10</td>
</tr>
<tr>
<td>Zheng et al. [93]</td>
<td>Network traffic, I/O performance</td>
<td>Migration time</td>
<td>High</td>
<td>data blocks scheduling on locality, popularity</td>
<td>Network traffic reduction: 51 – 86%</td>
</tr>
</tbody>
</table>
migration has been proposed in [93], where the disk data blocks are scheduled in an optimized order for migration, so that the migration network traffic is reduced and the storage I/O performance is improved during the live WAN migration of storage. In particular, the schedule order of data blocks is determined by tracking the temporal and spatial locality characteristics of the read and write operations, as well as the popularity characteristic of data blocks of the migrating workload.

4.4 Summaries, Insights and Open Research Issues

From the above detailed review and comparison of the existing techniques that mitigate the performance overhead of VMs in Sec. 4.1, 4.2, and 4.3, we bring forth the following summaries, insights and open research issues:

First, we come to the conclusion that the live migration of VMs serves as the cornerstone of managing the computing resources and alleviating the performance overhead of VMs in IaaS clouds. Specifically, in addition to enabling the routine operations including load balancing and power saving [47], live VM migration has also been proved to be effective in controlling the VM performance overhead in diverse scenarios, ranging from the single server virtualization (e.g., [10]) to a single datacenter (e.g., shared network resource [49] and shared storage resource [73]). In particular, a number of overhead mitigation techniques (e.g., [10], [48]) requires adjusting the mapping of VMs to physical servers. Such a VM adjusting process is enabled by the live VM migration during the execution of workloads. As a result, managing the performance overhead of VMs caused by live VM migration (Sec. 4.2.1) plays a critical role in guaranteeing the performance of VMs in the IaaS cloud.

Second, the conclusion above gives an impetus to mitigate the performance overhead of VMs caused by live VM migration in a cost-effective manner. To improve the performance of live VM migration itself in a LAN or a WAN, existing techniques mainly rely on two aspects (refer to Table 5 and Table 8): reducing the amount of transferred data during live VM migration and lowering the memory or disk dirty rate. Specifically, (1) a number of techniques have been proposed to reduce the transferred data of migration, for example, memory page compression (e.g., [57]), logging and replying VM execution events [61], memory pages pruning [58], spatial de-duplication of memory contents and disk images of multiple VMs (e.g., [24], [89]), and temporal de-duplication of the current and former memory pages of the migrating VM [32], [60]. (2) To control the dirty rate of memory or disks, existing techniques focus on throttling or blocking I/O requests to the migrating VM during the migration (e.g., [33], [90]). Although these techniques can significantly reduce the migration time, they either bring noticeable CPU and memory performance overhead to co-located VMs, or significantly impact the QoS of applications running on the migrating VM. Therefore, how to design a less “noisy” live migration approach while improving the performance of live VM migration itself will continue to be a challenge.

Third, although a number of migration policies have been extensively studied in the literature (e.g., [41], [43]), they focus on performance overhead of either the migrating VM or co-located VMs on migration source and destination servers during live migration of VMs (refer to Table 5). Few migration policies have devoted adequate attention to the VM performance overhead caused by VM co-location on a single server after live VM migration [47]. Specifically, as the migrating VM is moved onto a new physical server after the migration, the migrating VM will be likely to contend with the VMs co-located on the destination server for scarce physical resources. Accordingly, the VM performance overhead caused by VM co-location on the migration destination server cannot be overlooked (Sec. 2.1), when evaluating a specific migration choice (i.e., a tuple of the migrating VM and migration destination server). Consequently, it is essential for the VM migration policy to jointly consider the VM performance overhead caused by both the live VM migration and VM co-location (i.e., single server virtualization), during and after the migration. It would be a promising research problem to design such a performance-centric VM migration policy.

Fourth, it would be an interesting research topic to explore the tradeoff between VM performance overhead and the hardware provision in IaaS cloud datacenters, including physical servers, network switches and links, storages. Specifically, with hosting the same number of tenant VMs, the more physical hardware the datacenter provisions, the less performance overhead these VMs will experience. To the extreme, it is possible, but not practical to provision each VM with a physical server, each network connection between VMs with a physical network link, and each VM virtual disk with a physical storage device (a disk or a Redundant Array of Independent Disks (RAID)). In such an environment, the performance overhead of VMs can be dramatically decreased or nearly eliminated, whereas the resource utilization can be extremely low, which apparently hurts the benefits of cloud providers. Hence, it would be beneficial for cloud providers to explore such a tradeoff, in order to obtain insights into the tradeoff between guaranteeing the VM performance for tenants and maximizing the profits for cloud providers.

Fifth, most storage I/O sharing solutions in the literature focus on achieving one of three sharing requirements for VMs or tenants: the minimum guarantee, the proportional share, or high utilization of the storage resource (refer to Table 6). Although they have well isolated the storage resource and achieved good storage I/O performance among multiple VMs or tenants in the datacenter, whether one of the above three I/O sharing requirements contradicts with another remains unknown. It would be interesting to explore the tradeoff space among the three requirements for sharing the datacenter storage resource. Such a tradeoff analysis
can be further used for exploring the design space of storage I/O sharing solutions. In addition, the Solid State Driver (SSD) is already displacing the traditional Hard Disk Driver (HDD) in existing datacenters (e.g., Facebook, Dropbox) [94], and Amazon EC2 has recently released a new type of high I/O VM instance which is equipped with the SSD-based local storage [2]. These evidences above mandate a novel I/O sharing among VMs particularly on the SSD storage system, which has been seldom studied in the literature. It would be a future research problem to investigate VM I/O sharing solutions on the SSD-based storage or the hybrid storage system of HDD and SSD in datacenters.

Sixth, a number of existing network sharing solutions (e.g., [76], [78], [79], [82], [84], refer to Table 7) are based on the hose model, where all VMs are connected to a non-blocking logical switch through dedicated network connections. Such an assumption can simplify the VM networking model. The network capacity of central switch, however, cannot be infinite in practice. It has been also shown that, even in a datacenter network with very high bisection bandwidth, the network congestion can still happen anywhere, i.e., network links or switches [95]. As a result, the hose model does not quite conform to the networks in real-world datacenters. How to develop a realistic networking model, and then design a network sharing solution based on such a model becomes one potential research challenge. Furthermore, although existing network sharing solutions have studied the VM allocation of heterogeneous network bandwidth (e.g., [84], [85]), few of them have focused on sharing the network resource among VMs in a heterogeneous hardware environment, which apparently complicates the bandwidth sharing problem. Accordingly, how to design an effective VM network sharing solution under the environment of heterogeneous physical servers, network switches and links becomes a promising research topic.

Seventh, as revealed by an analysis of workload traces in Google’s production datacenter, the computing resource demands of cloud workloads are highly varying over short time intervals. In addition, various kinds of workloads are concurrently running in the datacenter. These evidences above undoubtedly make the prediction of workload resource demands much difficult [46]. Unfortunately, a number of VM performance management solutions (e.g., [67], [84]) rely on an accurate resource demand prediction technique. In face of such an environment with heterogeneous and dynamic workloads, how to design an effective workload resource prediction technique with a reasonable prediction error becomes a compelling research problem. In addition, the issue of VM performance overhead in a heterogeneous hardware environment becomes more severe and more complicated than that in a homogeneous environment. For example, as for selecting the migration destination server for a live VM migration or evaluating the assignment of VMs to physical servers, the heterogeneity of physical servers should be taken into consideration. As the hardware heterogeneity is likely to be encountered in IaaS cloud datacenters, even within the same type of Amazon EC2 instances [14], how to incorporate the hardware heterogeneity into existing solutions of managing VM performance overhead would be an interesting problem.

Last but not least, the hybrid use of public and private clouds will become a popular computing paradigm for tenants in the near future [96]. We believe that the research on managing VM performance overhead across geo-distributed datacenters will be enriched by the emerging applications or usages in the hybrid cloud, not restricted to the live WAN migration of VMs (Sec. 4.3). For example, in the scenario of IT strategy “Follow the sun” [32], the synchronization of a large amount of VM data (including the VM states, snapshots or even virtual disk images) across geo-distributed datacenters is required iteratively by the close collaboration among large business teams all over the world. Such VM synchronization can cause a significant amount of network traffic to the VMs in clouds. Therefore, how to fast alleviate such a heavy and intractable network load to the VMs for tenants? Can existing solutions on live WAN migration still handle such a network traffic problem? These questions above are practical future research problems on guaranteeing the performance of VMs across multiple geo-distributed datacenters.

5 Conclusion
As the trend of shifting the business service from the private datacenter to the public IaaS cloud prevails, one of the most important concerns of tenants lies on the performance overhead (i.e., unpredictable performance) of their leased VMs in the IaaS cloud. In this survey, we first elaborate the causes of VM performance overhead, from the single server virtualization, a single mega datacenter, to multiple geo-distributed datacenters. With a broad review and a detailed comparison of the state-of-the-art approaches that manage the VM performance overhead in the aforementioned three aspects, we further extract fruitful insights into the benefits and costs of existing solutions. Although the latest techniques are able to well control the performance overhead of VMs in IaaS clouds, a series of future research issues on the modeling methods (Sec. 3.3) and mitigation techniques (Sec. 4.4) of VM performance overhead remain to be solved, in order to provide predictable performance of VMs and guarantee the performance SLA of applications for tenants.

Acknowledgement
The corresponding author is Fangming Liu. The research was supported in part by a grant from National Natural Science Foundation of China under grant No.61370232, by a grant from National Basic Research Program (973 program) under Grant of 2014CB347800.
References


Fei Xu is a Ph.D candidate in the School of Computer Science and Technology, Huazhong University of Science and Technology, Wuhan, China. His current research interests focus on cloud computing and virtualization technology.
Fangming Liu is an associate professor in the School of Computer Science and Technology, Huazhong University of Science and Technology, Wuhan, China; and he is awarded as the CHUTIAN Scholar of Hubei Province, China. He is the Youth Scientist of National 973 Basic Research Program Project on Software-defined Networking (SDN)-based Cloud Datacenter Networks, which is one of the largest SDN projects in China. Since 2012, he has also been invited as a StarTrack Visiting Young Faculty in Microsoft Research Asia (MSRA), Beijing. He received his B. Engr. degree in 2005 from Department of Computer Science and Technology, Tsinghua University, Beijing; and his Ph.D. degree in Computer Science and Engineering from the Hong Kong University of Science and Technology in 2011. From 2009 to 2010, he was a visiting scholar at the Department of Electrical and Computer Engineering, University of Toronto, Canada. He was the recipient of two Best Paper Awards from IEEE GLOBECOM 2011 and IEEE CloudCom 2012, respectively. His research interests include cloud computing and datacenter networking, mobile cloud, green computing and communications, software-defined networking and virtualization technology, large-scale Internet content distribution and video streaming systems. He is a member of IEEE and ACM, as well as a member of the China Computer Federation (CCF) Internet Technical Committee. He has been a Guest Editor for IEEE Network Magazine, an Associate Editor for Frontiers of Computer Science, and served as TPC for IEEE INFOCOM 2013-2014 and GLOBECOM 2012-2013.

Hai Jin is a Cheung Kung Scholars Chair Professor of computer science and engineering at the Huazhong University of Science and Technology (HUST), China. He is now dean of the School of Computer Science and Technology at HUST. He received his Ph.D. degree in computer engineering from HUST in 1994. In 1996, he was awarded a German Academic Exchange Service fellowship to visit the Technical University of Chemnitz in Germany. He worked at the University of Hong Kong between 1998 and 2000, and as a visiting scholar at the University of Southern California between 1999 and 2000. He was awarded the Excellent Youth Award from the National Science Foundation of China in 2001. He is the chief scientist of ChinaGrid, the largest grid computing project in China, and chief scientist of the National 973 Basic Research Program Project of Virtualization Technology of Computing Systems. He is a senior member of the IEEE and a member of the ACM. His research interests include computer architecture, virtualization technology, cluster computing and grid computing, peer-to-peer computing, network storage, and network security.

Athanasios V. Vasilakos is currently a professor in the Department of Computer Science and Engineering, Kuwait University, Kuwait. He has authored or co-authored over 250 technical papers in major international journals and conferences. He is author/coauthor of five books and 20 book chapters in the areas of communications. He has served as General Chair and Technical Program Committee Chair for many international conferences. He has served or is serving as an Editor or/and Guest Editor for many technical journals, IEEE TNSM, IEEE TC, IEEE TSMC-PART B, IEEE TITB, ACM TAAS, and IEEE JSAC Special Issues in May 2009, and January and March 2011. He is founding Editor-in-Chief of the International Journal of Adaptive and Autonomous Communications Systems (IJAACS) and the International Journal of Arts and Technology (IJART). He is the Chairman of the Council of Computing of the European Alliances for Innovation.