Towards Efficient Resource Allocation for Heterogeneous Workloads in IaaS Clouds

Lei Wei, Chuan Heng Foh, Bingsheng He, Jianfei Cai

Abstract—Infrastructure-as-a-service (IaaS) cloud technology has attracted much attention from users who have demands on large amounts of computing resources. Current IaaS clouds provision resources in terms of virtual machines (VMs) with homogeneous resource configurations where different types of resources in VMs have similar share of the capacity in a physical machine (PM). However, most user jobs demand different amounts for different resources. For instance, high-performance-computing jobs require more CPU cores while big data processing applications require more memory. The existing homogeneous resource allocation mechanisms cause resource starvation where dominant resources are starved while non-dominant resources are wasted. To overcome this issue, we propose a heterogeneous resource allocation approach, called skewness-avoidance multi-resource allocation (SAMR), to allocate resource according to diversified requirements on different types of resources. Our solution includes a VM allocation algorithm to ensure heterogeneous workloads are allocated appropriately to avoid skewed resource utilization in PMs, and a model-based approach to estimate the appropriate number of active PMs to operate SAMR. We show relatively low complexity for our model-based approach for practical operation and accurate estimation. Extensive simulation results show the effectiveness of SAMR and the performance advantages over its counterparts.

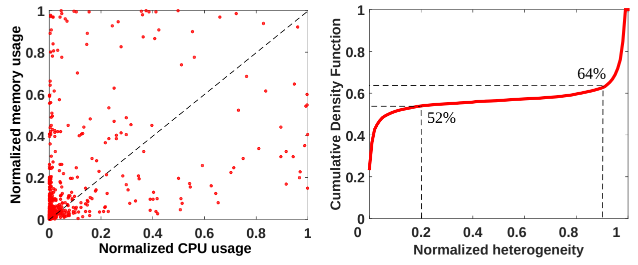
Keywords—Cloud computing, heterogeneous workloads, resource allocation

1 Introduction

Public clouds have attracted much attention from both industry and academia recently. Users are able to benefit from the clouds by highly elastic, scalable and economical resource utilizations. By using public clouds, users no longer need to purchase and maintain sophisticated hardware for the resource usage in their peak load. In recent years, many efforts [1], [2], [3], [4], [5], [6], [7] have been devoted to the problem of resource management in IaaS public clouds such as Amazon EC2 [8] and Rackspace cloud [9]. All these works have shown their strength in some specific aspects in resource scheduling and provisioning. However, existing works are all on the premise that cloud providers allocate virtual machines (VMs) with homogeneous resource configurations. Specifically, homogeneous resource allocation offers resources in terms of VMs where all the resource types have the same share of the physical machine (PM) capacity. Both dominant resource and non-dominant resource are allocated with the same share in such manner even if the demands for different resources from a user are different.

Obviously, using homogeneous resource allocation approach to serve users with different demands on various resources is not efficient in terms of green and economical computing [10]. For instance, if users need Linux servers with 16 CPU cores but only 1GB memory, they still require to purchase *m4.4xlarge* (with 16 vCPU and 64 GB RAM) or *c4.4xlarge* (with 16 vCPU and 30 GB RAM) in Amazon EC2 [8] (July 2, 2015), or *Compute1-30* (with 16 vCPU and 30 GB RAM) or *I/O1-60* (with 16 vCPU and 60 GB RAM) in Rackspace [9] (July 2, 2015) to satisfy users' demands. In this case, large memory will be wasted. As the energy consumption by PMs in data centers and the corresponding cooling system is the largest portion of cloud costs [10], [11], [12], homogeneous resource allocation that provisions large amounts of idle resources wastes tremendous energy. Even in the most energy-efficient data centers, the idle physical resources may still contribute more than one half of the energy consumption in their peak loads. Besides, for cloud users, purchasing the appropriate amounts of resources for their practical demands is able to reduce their monetary costs, especially when the resource demands are mostly heterogeneous.

- *L. Wei, B.S. He, J.F. Cai are with the School of Computer Engineering, Nanyang Technological University, Singapore.*
  E-mail: {weil0008, BSHE, ASJFCai }@ntu.edu.sg.
- *C.H. Foh is with the Centre for Communication Systems Research, University of Surrey, Guildford, Surrey, UK.*
  E-mail: c.foh@surrey.ac.uk.



(a) Resource usage of CPU and RAM (normalized to $(0, 1)$)

(b) CDF of heterogeneity

Fig. 1. Resource usage analysis of Google Cluster Traces.

We observe that most resource demands of the applications in cloud workloads are diversified on multiple resource types (e.g., number of CPU cores, RAM size, disk size, bandwidth, etc.). As shown in Fig. 1, we analyzed the normalized resource (CPU and RAM) usages of a cloud computing trace from Google [13], [14] which consists of a large amount of cloud computing jobs. It is clear that different jobs in Google trace have different demands in various resource types. Fig. 1(a) shows the comparisons of normalized CPU and RAM usages for the first 1000 jobs in Google trace. We can see that most jobs do not utilize the same share of different resource types. Allocating resource according to the dominant resource naturally wastes many non-dominant resources. Fig. 1(b) analyzes the distribution of the heterogeneity (defined as the difference between CPU and RAM usage, or $|CPU_{usage} - RAM_{usage}|$) for all jobs in Google trace. It reveals that more than 40% of the jobs are highly unbalanced between CPU and memory usage, and there are approximately 36% jobs with heterogeneity higher than 90%. Homogeneous resource allocation will not be cost-efficient for such heterogeneous workloads in the clouds because the non-dominant resources will be wasted significantly. Therefore, a flexible and economical resource allocation method for heterogeneous workloads is needed.

Nevertheless, consideration of heterogeneous workloads in resource allocation results in a number of challenges. Firstly, the resource demands in users' jobs are skewed among various resources. If the skewness of resource usages is ignored in resource allocation, some specific resource types with high demand may be exhausted before other resource types with low demand. Secondly, the complexity of resource allocation considering multiple resource types will be significantly increased. The complexity of provisioning algorithms for homogeneous resource allocation [15], [16] is already high and the computational time is long given the large number of PMs in data centers nowadays. The further consideration of multiple resources adds additional dimensions to the computation which will significantly increase the complexity. Thirdly, the execution time of some jobs (e.g., Google trace) can be as short as a couple of minutes which rapidly changes the PM utilization. This rapid change makes provisioning and resource allocation challenging.

To cope with the heterogeneous workloads, this paper proposes a skewness-avoidance multi-resource (SAMR) allocation algorithm to efficiently allocate heterogeneous workloads into PMs. SAMR designs a heterogeneous VM offering strategy that provides flexible VM types for heterogeneous workloads. To measure the skewness of multi-resource utilization in data center and reduce its impact, SAMR defines the multi-resource skewness factor as the metric that measures both the inner-node and the inter-node resource balancing. In resource allocation process, SAMR first predicts the required number of PMs under the predefined VM allocation delay constraint. Then SAMR schedules the VM requests based on skewness factors to reduce both the inner-node resource balance among multiple resources and the inter-node resource balance among PMs in the data center. By such manner, the total number of PMs are reduced significantly while the resource skewness is also controlled to an acceptable level.

Based on our earlier work in [15] which provisions heterogeneous workloads for preset delay constraint, in this paper, we propose a skewness factor based scheme to further optimize the resource allocation for heterogeneous workloads in clouds. Our experimental evaluation with both synthetic workloads and real world traces from Google shows that our approach is able to reduce the resource provisioning for cloud workloads by 45% and 11% on average compared with the single-dimensional method and the multi-resource allocation method without skewness consideration, respectively.

Organization. The rest of the paper is organized as follows. We first review the related work in Section 2. Section 3 describes the system model of our proposed algorithm SAMR and Section 4 provides detailed description of our proposed heterogeneous resource allocation algorithm SAMR. Section 5 introduces our developed resource prediction model based on Markov Chain. We present experimental results and discussions in Section 6, and draw important conclusion in Section 7.

2 Related Work

There have been efforts in resource allocation in cloud data center. In this section, we organize the literature review for resource management in clouds in two main categories: homogeneous resource allocation and heterogeneous resource allocation.

2.1 Homogeneous Resource Allocation

In the field of homogeneous resource allocation, the main research issue is the mapping of VMs into PMs under some specific goals. Bin packing is a typical VM scheduling and placement method that has been explored by many heuristic policies [17], [18], [19], [1] such as first fit, best fit and worst fit, and others. Some recent studies [18], [20] show that the impact on resource usage among various heuristic policies is similar. However, these policies cannot apply directly to heterogeneous resource provisioning because they may cause resource usage imbalance among different resource types.

Some recent works investigated scheduling of jobs with specific deadlines [3], [4], [5], [6]. As cloud workload is highly dynamic, elastic VM provisioning is difficult due to load burstiness. Ali-Eldin *et al.* [21] proposed using an adaptive elasticity control to react the sudden workload changes. Niu *et al.* [22] designed an elastic approach to dynamically resize the virtual clusters for HPC applications. These methods are shown to be effective on specific performance objectives. However, none of these scheduling methods is able to consistently offer the best performance for all workload patterns. Thus, Deng *et al.* [7] recently proposed a portfolio scheduling framework that attempts to select the optimal scheduling approach for different workload patterns with limited time. So far, all the research works assume that cloud providers offers VMs homogeneously and all resources are allocated according to their dominant resources. As discussed in Section 1, such single-dimensional resource allocation method is inefficient on resource usage as well as cost of both users and cloud providers.

Another significant problem in homogeneous resource allocation is resource provisioning which targets on determining the required resources for cloud workloads. To achieve green and power-proportional computing [10], cloud providers always seek elastic management on their physical resources [23], [12], [15], [11], [24]. Li *et al.* [23] and Xiao *et al.* [11] both designed similar elastic PM provisioning strategy based on predicted workloads. They adjust the number of PMs by consolidating VMs in over-provisioned cases and powering on extra PMs in under-provisioned cases. Such heuristic adjusting is simple to implement, but the prediction accuracy is low. Model-based PM provisioning approaches [16], [12], [25], [15], on the other hand, are able to achieve more precise prediction. Lin *et al.* [12] and Chen *et al.* [25] both proposed algorithms that minimize the cost of data center to seek power-proportional PM provisioning. Hacker *et al.* [16] proposed a hybrid provisioning for both HPC and cloud workloads to cover their features in resource allocation (HPC jobs are all queued by the scheduling system, but jobs in public clouds use all-or-nothing policy). However, these approaches only consider CPU as the dominant resource in single-dimensional resource allocation. To handle the provisioning problem for heterogeneous workloads, this paper proposes a model-based provisioning method that provisions minimum amount of resources while satisfying the allocation delay constraint.

2.2 Heterogeneous Resource Allocation

There have been a number of attempts made on heterogeneous resource allocation [26], [27], [28], [29], [30], [31] for cloud data centers. Dominant resource fairness (DRF) [28] is a typical method based on max-min fairness scheme. It focuses on sharing the cloud resources fairly among several users with heterogeneous resource requirements on different resources. Each user takes the same share on its dominant resource so that the performance of each user is nearly fair because the performance relies on the dominant resource significantly. Motivated by this work, a number of extensions based on DRF have been proposed [27], [31], [30]. Bhattacharya *et al.* [31] proposed a hierarchical version of DRF that allocates resources fairly among users with hierarchical organizations such as different departments in a school or company. Wang *et al.* [27] extended DRF from one single PM to multiple heterogeneous PMs and guarantee that no user can acquire more resource without decreasing that of others. Joe *et al.* [30] claimed that DRF is inefficient and proposed a multi-resource allocating framework which consists of two fairness functions: DRF and GFJ (Generalized Fairness on Jobs). Conditions of efficiency for these two functions are derived in their work. Ghodsi *et al.* [29] studied a constrained max-min fairness scheme that has two important properties compared with current multi-resource schedulers including DRF: incentivizing the pooling of shared resources and robustness on users' constraints. These DRF-based approaches mainly focus on performance fairness among users in private clouds. They do not address the skewed resource utilization.

Zhang *et al.* [32], [33] recently proposed a heterogeneity-aware capacity provisioning approach which considers both workload heterogeneity and hardware heterogeneity in IaaS public clouds. They divided user requests into different classes (such as VMs) and fit these classes into different PMs using dynamic programming. Garg *et al.* [34] proposed an admission control and scheduling mechanism to reduce costs in clouds and guarantee the performance of user's jobs with heterogeneous resource demands. These works made contributions on serving heterogeneous workloads in clouds. But they did not consider the resource starvation problem which is the key issue in heterogeneous resource provisioning in clouds. Thus, in this paper, we propose a novel approach to allocate resources with a skewness-avoidance mechanism to further reduce the PMs provisioned for heterogeneous workloads with acceptable resource allocation delay.

3 System Overview

In this section, we introduce the application scenario of our research problem and provide a system overview on our proposed solution for heterogeneous resource allocation. Table 1 lists the key notations used throughout this paper.

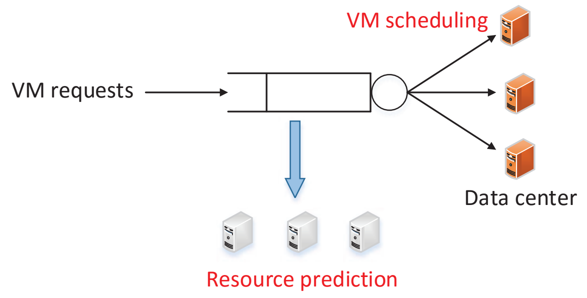


Fig. 2. System architecture of SAMR.

Similar to other works that optimize the resource usages in the clouds [10], [11], [12], we use the number of active PMs as the main metric to measure the degree of energy consumption in clouds. Reducing the number of active PMs in data center to serve the same amount of workloads with similar performance to users is of great attraction for cloud operators.

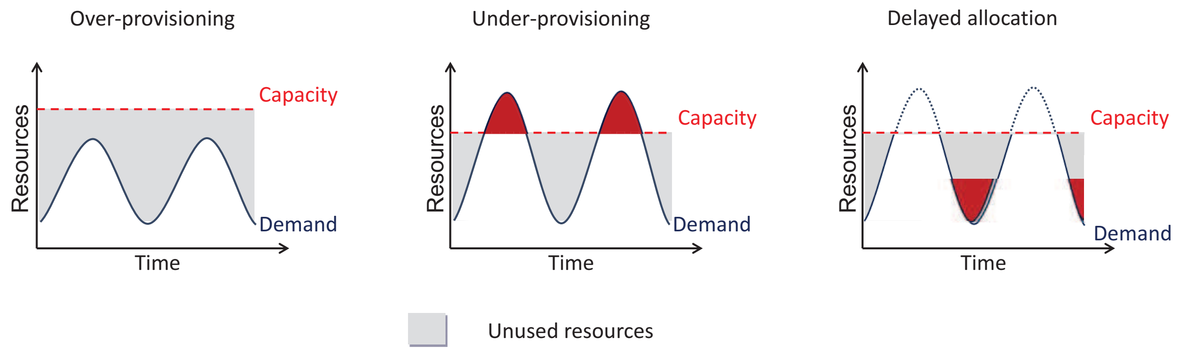


Fig. 3. The cases of over-provisioning, under-provisioning and delay caused by under-provisioning.

TABLE 1
Notations used in algorithms and models

K	Number of resource types
N_{total}	Total number of PMs in the considered data center
$\vec{R}$	r_i is the capacity of type-i resource in a PM, $i = [1, 2, ..., K]$
$\vec{M}$	m_i, $m_i < r_i$ is the maximum resource for type-i resource in a VM, $i = [1, 2, ..., K]$
X	Total number of VM types
$\vec{V}^x$	The resource configuration of type-x VM, v_i^x ($v_i^x \leq m_i$) represents the amount of type-i resource, $x = [1, 2, ..., X]$ and $i = [1, 2, ..., K]$
$\vec{C}$	c_i is the total consumed type-i resource in a PM, $c_i \leq r_i$, $i = [1, 2, ..., K]$
$\vec{U}$	u_i is the utilization of type-i resource in a PM, $u_i \in [0, 1]$, $i = [1, 2, ..., K]$
λ_x	Arrival rate of type-x requests, $x = [1, 2, ..., X]$
μ_x	Service rate of type-x requests, $x = [1, 2, ..., X]$
D	Predefined VM allocation delay threshold
d	Actual average VM allocation delay in a time slot
N	Provisioned number of active PMs (predicted by the model)
$\vec{S}$	s_n is the skewness factor for n^{th} active PM, $n = [1, 2, ..., N]$

We consider the scenario where cloud users rent VMs from IaaS public clouds to run their applications in a pay-as-you-go manner. Cloud providers charge users according to the resource amounts and running time of VMs. Fig. 2 shows the system model of our proposed heterogeneous resource allocation approach SAMR. Generally, we assume that a cloud data center with N_{total} PMs offers K different resource types (e.g., CPU, RAM, Disk, ...). The cloud system offers X different VM types, each of which is with a resource combination $\vec{V}^x = \{v_i^x | i = 1, 2, ..., K\} (x = 1, 2, ..., X)$ where v_i^x denotes the resource capacity of i^{th} resource type in x^{th} VM type. Cloud users submit their VM requests (also denoted as workloads in this paper) to the cloud data center according to their heterogeneous resource demands and choose the VM types that are most appropriate in terms of satisfying the user demands while minimizing the resource wastage. We refer a request for x^{th} type of VM as a type-x request in workloads. All VM requests are maintained by a scheduling queue. For each request from users, resource (or VM) scheduler allocates the resources for requested VM in N current active PMs if the resource slot of the VM is available. Otherwise, the request will be delayed waiting for more PMs to power up and join the service. According to the arrival rates and service rates of requests, SAMR conducts resource prediction based on a Markov Chain model periodically in every time slot with a duration of t to satisfy the user experience in terms of VM allocation delay. By such manner, we focus on solving the problem in a small time period to increase the prediction accuracy. After the online prediction of required resources, the cloud system provisions corresponding number of active PMs N in the coming time slot. In VM scheduling phase during each time slot with the length t, cloud providers allocate resources and host each VM into PMs using SAMR allocation algorithm.

In cloud service, one of the most significant impacts on user experience is the service delay caused by schedulers. Here we consider the resource (or VM) allocation delay as the main metric for service-level-agreements (SLA) between users and cloud providers. Specifically, SAMR uses a VM allocation delay threshold D to be the maximum SLA value that cloud providers should comply with. Thus, there is a trade off between cost and SLA (as shown in Fig. 3) for cloud providers. To cope with the large amount of random request arrivals from users, it is important to provision enough active PMs. However, maintaining too many active PMs may cope well even under peak load but wastes energy unnecessary. Maintaining too few PMs may cause significant degradation in user experience due to lacks of active PMs and the need to wait for powering up more PMs. It is challenging to find the adequate number of active PMs. In our work, during the resource prediction phase, SAMR uses a Markov Chain model to find the adequate number of active PMs that satisfies the SLA value. Precisely, the model determines the number of active PMs, N, such that the average VM allocation delay d is smaller than the agreed threshold D.

We use the Markov Chain model to determine the adequate number of active PMs for operation. The model assumes heterogeneous workloads and balanced utilization of all types of resources within a PM. To realize the balanced utilization, we define a multi-resource skewness as the metric to measure the degree of unbalancing among multiple resource types as well as multiple PMs. The SAMR scheduling aims to minimize the skewness in data center in order to avoid the resource starvation. The detail of skewness-avoidance resource allocation algorithm and model-based resource prediction are discussed in Section 4 and Section 5, respectively.

4 SKEWNESS-AVOIDANCE MULTI-RESOURCE ALLOCATION

In this section, we describe our proposed skewness-avoidance multi-resource allocation algorithm. Firstly, we introduce new notions of VM offering for heterogeneous workloads in clouds. Then we define skewness factor as the metric to characterize the skewness of multiple resources in a data center. Finally, based on definition of skewness factor, we propose a SAMR allocation algorithm to reduce resource usage while maintaining the VM allocation delay experienced by users to a level not exceeding the predefined threshold.

4.1 New Notions of VM Offering

Generally, we consider a cloud data center with N_{total} PMs, each of which have K types of computing resources. We denote $\vec{R} =< r_1, r_2, ..., r_K >$ to be the vector describing the capacity of K types of resources and $\vec{C} =< c_1, c_2, ..., c_K >$ to be the vector that describing the amount of resource used in a PM. To support better utilization of resources for cloud applications with heterogeneous resource demands, it is necessary to consider a new VM offering package to cover the flexible resource allocation according to different resource types. We propose SAMR to offer a series of amounts for each resource type and allow arbitrary resource combinations that a user can pick. For instance, a cloud provider offers and charges VMs according to K resource types (e.g., CPU, RAM, disk storage, bandwidth,...) and the maximum amount of type-i resource ($i = 1, 2, ..., K$, we refer i^{th} resource type as type-i resource in this paper) is m_i. For each type of resource, there is a list of possible amounts for users to choose, and we consider a list of power of 2 for the amounts (e.g., $1, 2, 4, 8, ...$) for convenience (SAMR can actually support arbitrary sizes of VMs). Thus, the total number of VM types are $X = \prod_{i=1}^{K} (\log_2(m_i) + 1)$. We use $\vec{V^x} =< v_1, v_2, ..., v_K >^x$, for $x = [1, 2, ..., X]$, to present a resource combination for type-x VM. SAMR allows users to select the suitable number of resource for each type. Thus, users are able to purchase the appropriate VMs that optimally satisfy their demands to avoid over-investments. We use an example to illustrate above VM offering package. A cloud system may identify two resource types: CPU and memory. The amounts of CPU (number of cores), memory (GB) are expressed by $\vec{V^x} =< v_1, v_2 >^x$. If each PM have 16 CPU cores and 32 GB memory and it allows the maximum VM to use all the resources. Users can select 1 core, 2 cores, 4 cores, ..., or 16 cores of CPU combining with 1 GB, 2 GB, 4 GB, ..., or 32 GB of memory for their VMs. Thus, this configuration permits a total of 30 ($X = 30$) different types of VMs, namely $< 1, 1 >^1, < 1, 2 >^2, ..., < 16, 16 >^{29}, < 16, 32 >^{30}$.

While the current virtualization platforms such as Xen and Openstack are ready to support this flexible offering, finding the right number of options to satisfy popular demands and developing attractive pricing plans that can ensure high profitability are not straightforward. We recognize that the precise design of a new VM offering is a complicated one. Our considered VM offering package is used to illustrate the effectiveness of SAMR. However, SAMR is not limited to a particular VM offering package.

4.2 Multi-Resource Skewness

As discussed in Section 1, heterogeneous workloads may cause starvation of resources if the workloads are not properly managed. Although live migration can be used to consolidate the resource utilization in data centers to unlock the wasted resources, live migration operations result in service interruption and additional energy consumption. SAMR avoids resource starvation by balancing the utilization of various resource types during the allocation. Migration could be used to further reduce the skewness in the runtime of cloud data center if necessary.

Skewness [11], [35] is widely used as a metric for quantifying the resource balancing of multiple resources. To better serve the heterogeneous workloads, we develop a new definition of skewness in SAMR, namely *skewness factor*.

Let $\mathbf{G} = \{1, 2, ..., K\}$ be the set that carries all different resource types. We define the mean difference of the utilizations of K resource types as

$$Diff = \frac{\sum_{(i \in \mathbf{G}, j \in \mathbf{G}, i \neq j)} |u_i - u_j|}{K \cdot (K-1)}, \tag{1}$$

where u_i is the utilization of i^{th} resource type in a PM. Then the average utilization of all resource types in a PM is $\overline{U}$, which can be calculated by

$$\overline{U} = \frac{\sum_{i=1}^{K} u_i}{K}. \tag{2}$$

The skewness factor of n^{th} PM in a cloud data center is defined by

$$s_n = \frac{Diff}{\overline{U}} = \frac{\sum_{(i \in \mathbf{G}, j \in \mathbf{G}, i \neq j)} |u_i - u_j|}{(K-1) \cdot \sum_{i=1}^{K} u_i}. \tag{3}$$

The concept of skewness factor is denoted as a factor that quantifies the degree of skewness in resource utilization in a data center with multiple resources. The degree of skewness factor has the following implication and usages.

- The value of skewness factor is non-negative ($s_n \geq 0$), where 0 indicates that all different types of resources are utilized at the same level. The skewness factor closer to 0 reveals lower degree of unbalanced resource usages in a PM. Thus, our scheduling goal is to minimize the average skewness factor. In contrast, a larger skewness factor implies higher skewness, which means that the resource usages are skewed to some specific resource types or some PMs. It also indicates that the PMs have a high probability of resource starvation.

- The skewness factor is the main metric in skewness-avoidance resource allocation for heterogenous workloads. Thus, in the definition of skewness factor, we consider two aspects of the characteristics of the resource usages in PMs to keep the inner-node and inter-node resource balancing. The first aspect is the mean differences between the utilizations of multi-resources within a PM, or inner-node aspect. A higher degree of difference leads to a higher skewness factor, which is translated to higher degree of unbalanced resource usage. The second aspect in skewness factor is the mean of utilization of multi-resources in a PM. When the first aspect, the mean difference, is identical in each PM in data center, SAMR always choose the PM with the lowest mean utilization to host new VM requests such that the inter-node balance between PMs is covered in the definition of skewness factor.
- The resource scheduler makes scheduling decisions according to the skewness factors of all active PMs in data center. For each VM request arrival, the scheduler calculates the skewness factor for each PM as if the VM request was hosted in the PM. Thus, the scheduler is able to find the PM with the most skewness reduction after hosting the VM request. This strategy not only keeps the mean skewness factor of the PM low, but also maintain a low mean skewness factor across PMs. The detailed operation of the skewness-avoidance resource allocation algorithm is provided in the next subsection.

Algorithm 1 Allocation algorithm of SAMR

```
1: Provision $N$ PMs with predition model in Section 5
2: Let $N^{'}$ be the current number of PMs at the beginning of the time slot
3: if $N > N^{'}$ then
4:     Powering on $N - N^{'}$ PMs
5: else if $N < N^{'}$ then
6:     Shut down $N^{'} - N$ PMs
7: if a type-$x$ VM request arrives at cloud system with demand $\vec{V}^x$ then
8:     $opt = 0$
9:     $s_{opt} = 0$
10:    for $n = 1$ to $N$ do
11:        if $\vec{C} + \vec{V}^x \leq \vec{R}$ then
12:            Compute $s_n$ with Eq. 3
13:            Compute new $s_n^{'}$ if host the type-$x$ request
14:            if $s_n - s_n^{'} > s_{opt}$ then
15:                $opt = n$
16:                $s_{opt} = s_n - s_n^{'}$
17:    if $opt == 0$ then
18:        Power on a PM to allocate the request
19:        Delay the VM allocation for time $t_{power}$
20:        $N = N + 1$
21:    else
22:        Allocate this VM request to $opt^{th}$ PM: $\vec{C} = \vec{C} + \vec{V}^x$
23: if a type-$x$ VM finishes in the $n^{th}$ PM then
24:     Recycle the resource: $\vec{C} = \vec{C} - \vec{V}^x$
```

4.3 Skewness-Avoidance Resource allocation

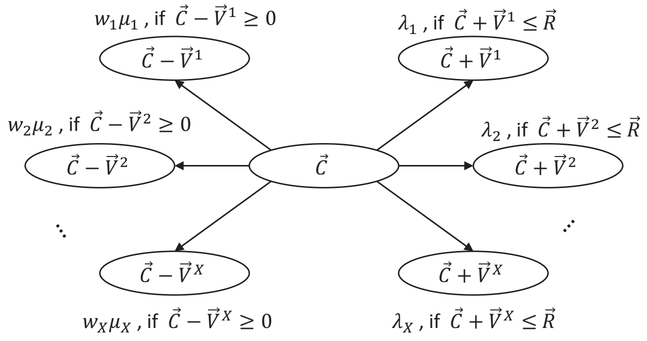
Based on the specification of the multi-resource skewness, we propose SAMR as the resource allocation algorithm to allocate heterogeneous workloads. Algorithm 1 outlines the operation of SAMR for each time slot of duration t.

At the beginning of a time slot, the system uses past statistics to predict the number of active PMs needed to serve the workloads. Our model-based prediction will be discussed in detail in Section 5. Then, the system will proceed to add or remove active PMs based on the prediction.

As each VM request arrives, the system conducts the following steps: 1) The scheduler fetches one request from the request queue. According to the VM type requested, the scheduler starts searching the active PM list for a suitable vacancy for the VM. 2) In the search of each PM, the scheduler first checks whether there are enough resources for the VM in the current active PM. If a PM has enough resources to host the requested VM, the scheduler calculates the new multi-resource skewness factor and records the PM with maximum decease in skewness factor. For the PM without enough resources, the scheduler simply skips the calculation. 3) After the checking for all active PMs, the scheduler picks the PM with the most decrease in skewness factor to host the VM. The most decrease in skewness factor indicates the most improvement in balancing utilization of various resources. In the case that there is no available active PM to host the requested VM, an additional PM must be powered up to serve the VM. This request will experience additional delay (t_{power}) due to the waiting time for powering up a PM. 4) After each VM finishes its execution, the system recycles the resources allocated to the VM. These resources will become available immediately for new requests.

5 Resource Prediction Model

In this section, we introduce the resource prediction model of SAMR. The objective of the model is to provision the active number of PMs, N, at the beginning of each time slot. To form an analytical relationship between operational configurations and performance outcomes, we develop a Markov Chain model describing the evolution of resource usage for SAMR in the cloud



$w_1\mu_1$, if $\vec{C}-\vec{V}^1 \geq 0$
$\vec{C}-\vec{V}^1$
λ_1 , if $\vec{C}+\vec{V}^1 \leq \vec{R}$
$\vec{C}+\vec{V}^1$
$w_2\mu_2$, if $\vec{C}-\vec{V}^2 \geq 0$
$\vec{C}-\vec{V}^2$
$\vec{C}$
λ_2 , if $\vec{C}+\vec{V}^2 \leq \vec{R}$
$\vec{C}+\vec{V}^2$
$\vec{C}-\vec{V}^X$
$\vec{C}+\vec{V}^X$
$w_X\mu_X$, if $\vec{C}-\vec{V}^X \geq 0$
λ_X , if $\vec{C}+\vec{V}^X \leq \vec{R}$

Fig. 4. State transitions in the model.

data center. With the model, we can determine the optimal number of PMs for cost-effective provisioning while meeting VM allocation delay requirement.

One of the advantages of cloud computing is the cost effectiveness for users and service providers. Cloud users wish to have their jobs completed in the cloud in lowest possible cost. Therefore, reducing their cost by eliminating idle resources due to homogeneous resource provisioning is an effective approach. However, due to the complexity in multiple dimensional resource type management, large scale deployment of PMs, and the highly dynamic nature of workloads, it is a non-trivial task to predict the suitable number of active PMs that can meet the user requirement. Modeling all N_{total} PMs and all K types of resource in a data center leads to a model complexity level of $O((\prod_{i=1}^{K} r_i)^{3N_{total}})$ and $O((\prod_{i=1}^{K} r_i)^{2N_{total}})$ for computation and space complexity, respectively. For example, with 1000 PMs, 2 types of resources, each with 10 options, the system evolves over 10^{4000} different states. It is computationally intensive to solve a model involving such a huge number of states. Since the resources allocated to a VM must come from a single PM, we see an opportunity to utilize this feature for model simplification. Instead of considering all PMs simultaneously, we can develop a model to analyze each PM separately which significantly reduces the complexity.

We observe that the utilizations of different types of resources among different PMs in data center are similar in a long run under SAMR allocation algorithm because the essence of SAMR is keeping the utilizations balanced among different PMs. Since all active PMs share similar statistical behavior of the resource utilization, we focus on modeling a particular PM in the system. Such approximation method can largely reduce the complexity while providing an acceptable prediction precision. The model permits the determination of allocation delay given a particular number of active PMs, N. With the model, we propose a binary search to find the suitable number of active PMs such that the delay condition of $d \leq D$ can be met.

In our model, we first predict the workloads at the beginning of each time slot. There are many load prediction methods available in the literature [11], [36], we simply use the Exponential Weighted Moving Average (EWMA) in our paper. EWMA is a common method used to predict an outcome based on past values. At a given time τ, the predicted value of a variable can be calculated by

$$E(\tau) = \alpha \cdot O_b(\tau) + (1-\alpha) \cdot E(\tau - 1), \tag{4}$$

where $E(\tau)$ is the prediction value, $O_b(\tau)$ is the observed value at time τ, $E(\tau-1)$ is the previous predicted value, and α is the weight.

Next, we introduce the details for modeling each PM in SAMR provisioning method. Similar to previous works [16], [12], [15], we assume that the arrival rate of each type of VM request follows Poisson distribution and the execution time follows Exponential distribution. For type-x VM, the arrival rate and service rate are expressed by λ_x and μ_x, respectively. Since we consider each PM separately, the arrival rate for one single PM is divided by N.

Let $\vec{C}$ (a K-dimensional vector) be the system state in Markov Chain model where c_i represents the total number of used type-i resource in a PM. We denote $T\{\vec{S}|\vec{C}\}$ to be the rate of transition from state $\{\vec{C}\}$ to state $\{\vec{S}\}$. The outward rate transition from a particular system state, $\vec{C}$, in our model is given in Fig. 4 where the evolution of the system is mainly governed by VM request arrivals and departures. We provide the details of the state transitions in the following.

Let $I(\vec{C})$ be an indicator function defining the validity of a system state, where

$$I(\vec{C}) = \begin{cases} 1, & 0 \leq c_i \leq r_i, i = 1, 2, ..., K \\ 0, & \text{otherwise.} \end{cases} \tag{5}$$

An allocation operation occurs when there is an arrival of VM request to the cloud data center. When a VM request for type-x VM demands for $\vec{V}^x$ ($\vec{V}^x \leq \vec{R}$) resources, the system evolves from a particular state $\vec{C}$ to a new state $\vec{C}+\vec{V}^x$ provided that $\vec{C}+\vec{V}^x$ is a valid state. The rate of such a transition is

$$T\{\vec{C}+\vec{V}^x|\vec{C}\} = \lambda_x \cdot I(\vec{C}+\vec{V}^x). \tag{6}$$

The release of resources occurs when a VM finishes its execution. The rate of a release operation is decided by the number of VMs of each types because different type of VMs have different execution time. The number of a particular type in service is proportionate to its utilization of the system. Let w_x be the number of type-x VMs in a PM, w_x can be computed by

$$w_x = \frac{\sum_{i=1}^{K} \left[\frac{\frac{\lambda_x v_i^x}{\mu_x}}{\sum_{z=1}^{X} \frac{\lambda_z v_i^z}{\mu_z}} \cdot c_i \right]}{K}, \tag{7}$$

where the number of type-x VMs is determined by the mean value of the number of type-x VM calculated by K different resource types. Upon a depart of a type-x

request, the system state transits from state $\{\vec{C}\}$ to state $\{\vec{C} - \vec{V}^x\}$ with a transition rate given by the following

$$R\{\vec{C} - \vec{V}^x | \vec{C}\} = w_x \cdot \mu_x \cdot I(\vec{C} - \vec{V}^x). \tag{8}$$

With the above transition, the total number of valid states that the system can reach is expressed by

$$S = \prod_{i=1}^{K} (r_i + 1). \tag{9}$$

Then, an S-by-S infinitesimal generator matrix for the Markov Chain model (Q) can be constructed. The steady-state probability of each state, $p(\vec{C})$, can be solved numerically using the following balance equation,

$$p(\vec{C}) \cdot \left[\sum_{x=1}^{X} (w_x \cdot \mu_x \cdot I(\vec{C} - \vec{V}^x) + \lambda_x \cdot I(\vec{C} + \vec{V}^x)) \right] = \sum_{x=1}^{X} [p(\vec{C} - \vec{V}^x) \cdot \lambda_x \cdot I(\vec{C} - \vec{V}^x) I(\vec{C}) + p(\vec{C} + \vec{V}^x) \cdot w_x \cdot \mu_x \cdot I(\vec{C} + \vec{V}^x) I(\vec{C})]. \tag{10}$$

Obtaining the steady-state probabilities of the system allows us to study the performance at the system level. The resource utilization vector of a PM can be determined by

$$\vec{U} = \sum_{c_1=0}^{r_1} \sum_{c_2=0}^{r_2} ... \sum_{c_K=0}^{r_K} p(\vec{C}) \cdot (\vec{C}/\vec{R}). \tag{11}$$

We now analyze the probability that a VM request is delayed due to under-provision of active PMs. Let P_{d_x} be the delay probability of type-x requests, it can be computed by

$$P_{d_x} = \sum_{c_1=0}^{r_1} \sum_{c_2=0}^{r_2} ... \sum_{c_K=0}^{r_K} p(\vec{C}) \cdot (1 - I(\vec{C} + \vec{V}^x)) \tag{12}$$

The overall probability of a request being delayed in the considered time slot, P_d, can be determined by

$$P_d = \frac{\sum_{x=1}^{X} P_{d_x} \lambda_x}{\sum_{x=1}^{X} \lambda_x}. \tag{13}$$

After obtaining the above, the average VM allocation delay can be determined by

$$d = P_d \cdot J \cdot t_{power}, \tag{14}$$

where J is the total number of VM requests and t_{power} is the time for powering up an inactive PM.

Model Complexity. The prediction model in SAMR uses a multi-dimensional Markov chain that considers the K types of resources simultaneously. The time complexity to obtain a solution for the model is $O((\prod_{i=1}^{K} r_i)^3)$ where the r_i is the capacity of i^{th} resource type. The space complexity of the model is $O((\prod_{i=1}^{K} r_i)^2)$ which is the size of the infinitesimal generator matrix. Based on the analysis, adding more resources to each PM contributes insignificant to the complexity, however it may trigger introduction of new VM options to the system which increases r_i as well as the computational time and space. Likewise, considering additional resource type will certainly add VM options which increases the computational time and space. Nevertheless, current cloud providers usually consider two ($K = 2$) or three ($K = 3$) resource types on offering VMs, and thus it remains practical for SAMR to produce the prediction of resource allocation scheme in real time.

PM Scalability. The number of PMs, N_{total}, influences the prediction model and VM allocation algorithm. In the prediction model, a binary search is needed to check for the suitable number of PMs. The complexity is $O(\log(N_{total}))$. For the VM allocation algorithm execution, as it performs linear check on each active PM, the complexity is $O(N_{total})$. The overall complexity of our solution is thus linear to the number of PMs.

6 Evaluation

In this section, we evaluate the effectiveness of our proposed heterogeneous resource allocation approach with simulation experiments. First, we introduce the experimental setups including the simulator, methods for comparison and the heterogeneous workload data. Second, we validate SAMR with simulation results and then compare the results with other methods.

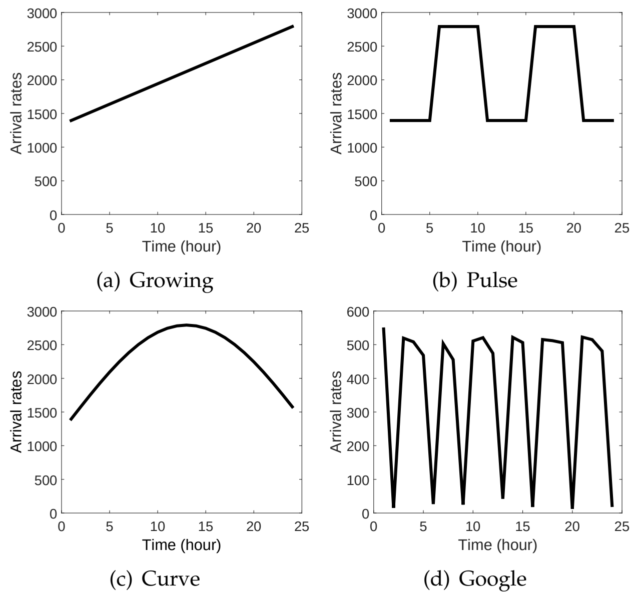


Fig. 5. Three synthetic workload patterns and one real world cloud trace from Google.

6.1 Experimental setup

Simulator. We simulate a IaaS public cloud system where VMs are offered in a on-demand manner. The

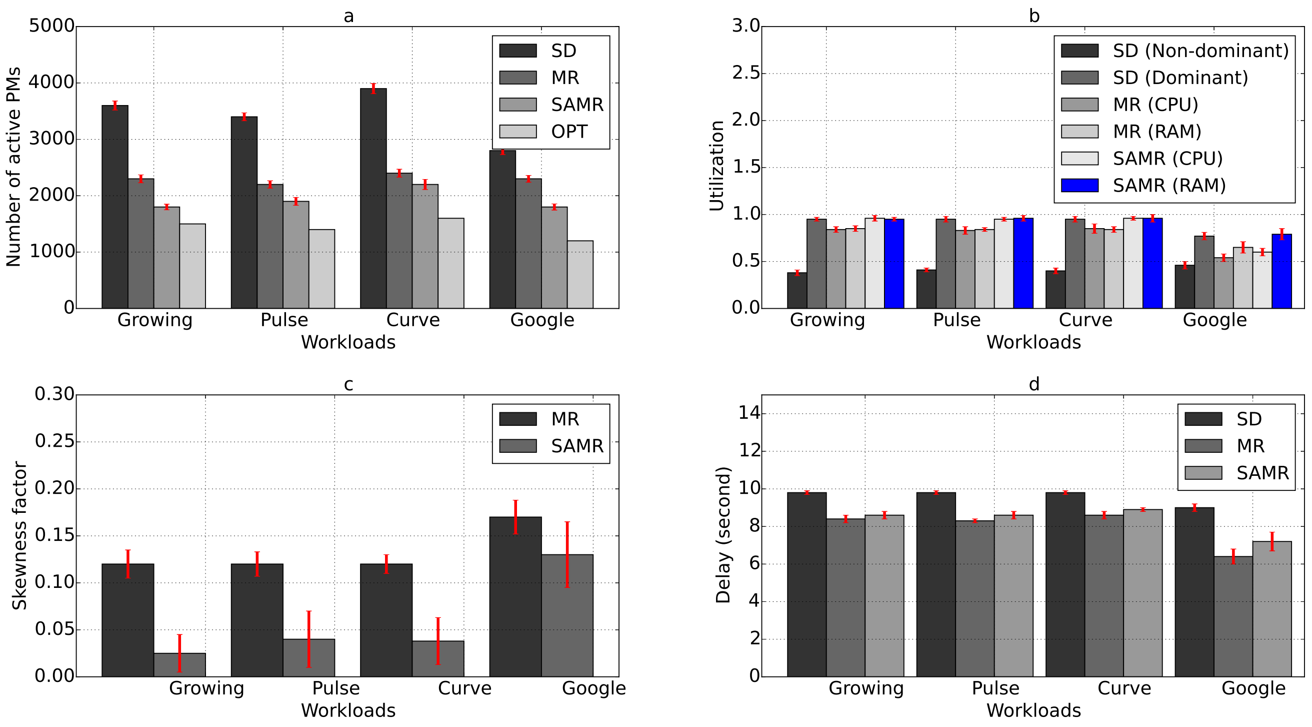


Fig. 6. Overall results of four metrics under four workloads. The bars in the figure show average values and the red lines indicate 95% confidence intervals.

simulator maintains the resource usage of PMs in the cloud and support leasing and releasing the resources for VMs requested by users. We consider offering of two resource types: CPU cores and memory. In our experiments, we set the time for powering on a PM to 30 seconds and the default average delay constraint is set to 10 seconds. The default maximum VM capacity is set to 32% of the normalized capacity of a PM. Besides, the default time slot for resource allocation is 60 minutes. To study their impact on system performance, sensitivities of these parameters are investigated in the experiments. We study the following performance metrics in each time slot: number of PMs per time slot, mean utilization of all active PMs, multi-resource skewness factor and average VM allocation delay. The number of PMs is the main metric which can impact the other three metrics.

Comparisons. To evaluate the effectiveness of SAMR in serving highly heterogeneous cloud workloads, we simulate and compare the results of SAMR with the following methods: 1) single-dimensional (SD). SD is the basic homogeneous resource allocation approach that is used commonly in current IaaS clouds. Resource allocation in SD is according to the dominant resource, other resources have the same share of dominant resource regardless of users' demands. For scheduling policy, we simply choose first fit because different scheduling policies in SD have similar performance impact on resource usage. In first fit, the provisioned PMs are collected to form a list of active PMs and the order of PMs in the list is not critical. For each request, the scheduler searches the list for available resources for the allocation. If the allocation is successful, the requested type of VM will be created. Otherwise, if there is no PM in the list that can offer adequate resources, this request will be delayed. 2) multi-resource (MR). Different from SD, MR is a heterogeneous resource allocation method which do not consider multi-resource skewness factor in resource allocation. MR offers flexible resource combinations among different types of resource to cover different user demands on different resource types. MR also uses first fit policy to host VMs in cloud data center. 3) optimal (OPT). An optimal resource allocation (OPT) is compared as the ideal provisioning method with oracle information of workloads. OPT assumes that all PMs run with utilizations of 100%. The provisioning results of OPT are calculated simply by dividing the total resource demands in each time slot by the capacity of the PMs. Thus, OPT is considered as the most extreme case that minimum number of PMs are provisioned for the workloads.

Workloads. Two kinds of workloads are utilized, synthetic workloads and real world cloud trace, in our experiments as shown in Fig. 5. In order to study the sensitivity of performance under different workload features, three synthetic workload patterns are used: growing, pulse and curve. By default, the lowest average request arrival rates of all three synthetic workload patterns are 1400 and the highest points are 2800. We keep the total resource demands of each type of VM requests similar so that the number of VM requests with higher resource demands is smaller. The service time of the VMs in synthetic workloads are set to exponential distribution

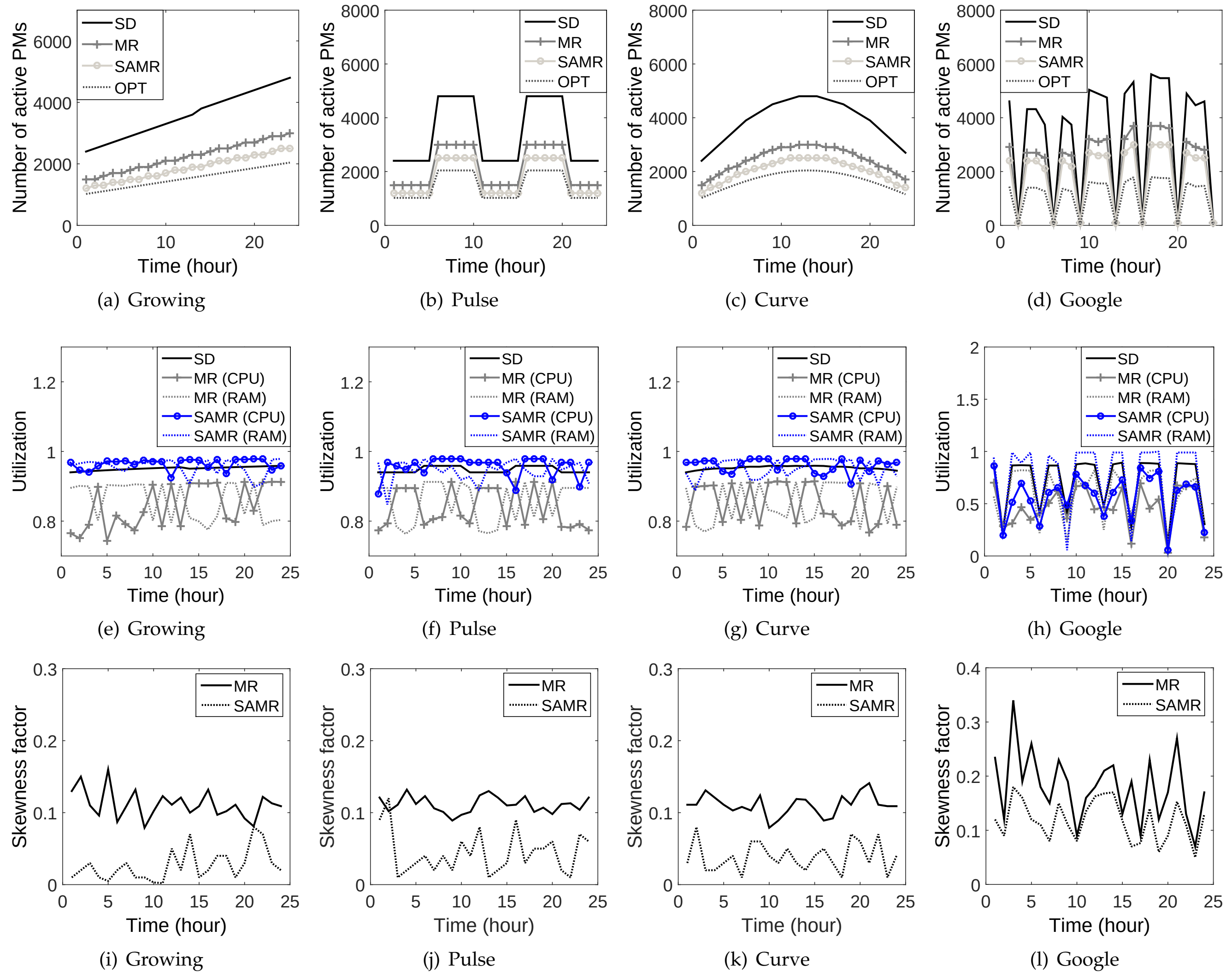


Fig. 7. Detailed results of three metrics under four workload patterns.

with average value of 1 hour.

To validate the effectiveness of our methods, we also use a large scale cloud trace from Google which is generated by the logs from the large scale cloud computing cluster containing 11000 servers in Google company. The trace records the system logs during 29 days from May 2011 and we pick the logs in the first day of the third week for experiments. We extract 73905 job submissions, each of which contains the job starting time, running time, CPU usage and memory usage. The exact configurations of the servers in Google cluster are not given in the trace and the resource usages use normalized values from 0 to 1 (1 is the capacity of a PM). Thus we also use the normalized resource usages in experiments for both synthetic workloads and Google trace. In experiments, we allocate a VM for each job according to its demands on multiple types of resources.

6.2 Experimental results

Overall results. We first present the overall results of the four methods for the four workloads. Fig. 6 shows the overall results for different metrics with all workloads and resource management methods. The bars in the figure show the average values for different results and the vertical red lines indicate the 95% confidence intervals.

We make the following observations based on the results. Firstly, heterogeneous resource management methods (MR and SAMR) significantly reduce resources in terms of number of active PMs for the same workloads. As shown in Fig. 6(a), the resource conservation achieved by MR compared with SD is around 34% for all four workloads. SAMR further reduces the required number of PMs by another 11%, or around 45% compared with SD. It shows that SAMR is able to effectively reduce the resource usage by avoiding resource starvation in cloud data center. Besides, the number of active PMs for SAMR is quite close to the optimal solution with only 13% difference. Note that the presented number of active PMs for SAMR is the actual required number for the given workloads. Based on our experiment records, the predicted numbers of PMs from our model have no more than 5% (4.3% on average) error rates compared

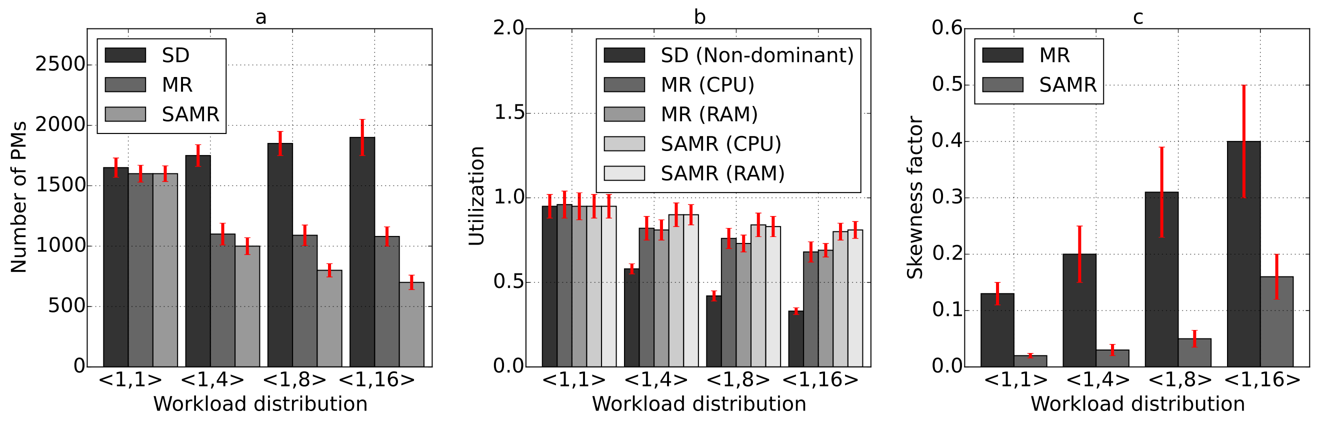


Fig. 8. Sensitivity studies for different degrees of heterogeneity (workload distributions). The bars in the figure show average values and the red lines indicate 95% confidence intervals.

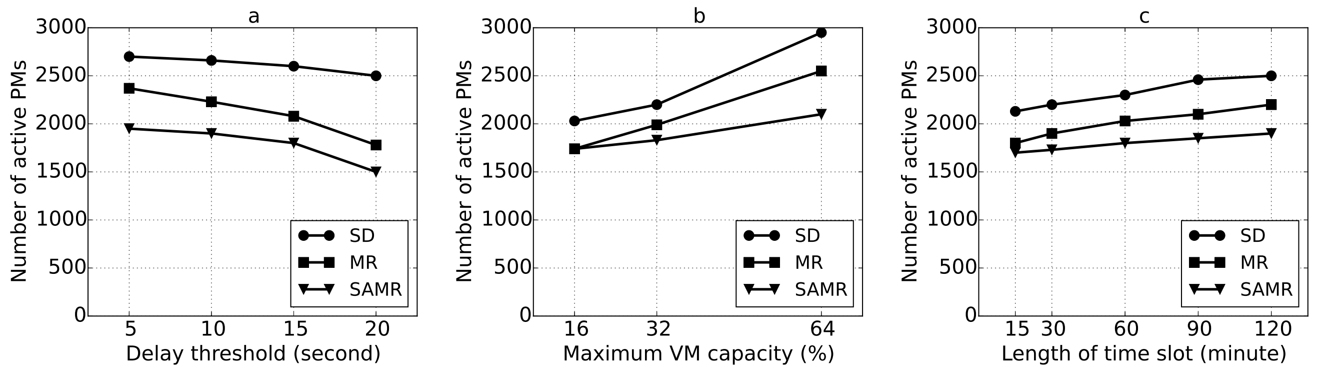


Fig. 9. Sensitivity studies for delay threshold, maximum VM capacity and length of time slot using Google trace.

with the actual required numbers presented in the figure. Secondly, although the utilization of dominant resource using SD method is high as shown in Fig. 6(b), the non-dominant resources are under-utilized. However, the resource utilizations in MR and SAMR policies are balanced. This is the reason that SD must provision more PMs. Thirdly, the effectiveness of resource allocation in SAMR is validated by the skewness factor shown in Fig. 6(c), where the average resource skewness factors in SAMR method are less than that in MR. Finally, all three policies achieve the predefined VM allocation delay threshold as shown in Fig. 6(d). SD holds slight higher average delays than SAMR and MR, which is due to the fact that SD always reacts slowly to the workload dynamicity and cause more under-provisioned cases to make the delay longer.

Impacts by the amount of workloads. Fig. 7 shows the detailed results of all methods for different metrics under four workloads. We highlight and analyze the following phenomenons in the results. Firstly, heterogeneous resource allocation methods significantly reduce the required number of PMs in each time slot for 4 workloads as in Fig. 7(a) to Fig. 7(d). Secondly, from Fig. 7(e) to Fig. 7(h) we can see that SAMR is able to maintain high PM utilization in data center but the PM utilization of MR method fluctuates, falling down under 80% frequently. This is due to the starvation or unbalanced usage among multiple resource types in MR as shown in Fig. 7(i) to Fig. 7(l). Thirdly, we observe that the utilization of CPU and RAM resources using SAMR are close in the three synthetic workloads but the difference in Google trace is large as shown in Fig. 7(e) to Fig. 7(h). This is caused by the fact that the total demands of RAM is more than that of CPU in traces from Google Cluster. It can also be verified by the higher resource skewness factors in Fig. 7(i) to Fig. 7(l), where the skewness factors in Google trace are much higher than the other three workloads.

We now perform sensitivity studies on major parameters. We investigate the impact of the system parameters including the degree of heterogeneity, delay threshold, the number of VM types and time slot length on the performance of multiple resource usage. For each experiment, we study the impact of varying one parameter while setting other parameters to their default values.

Impacts by workload heterogeneity. We first investigate the performance under different workload distributions with different degrees of heterogeneity. We run four experiments using Growing pattern in this study. In each experiment, the workload consists of only two types of VMs (the amounts of two types of VM are the same) with the same heterogeneity degree. Specifically, we use $<1,1> + <1,1>$, $<1,4> + <4,1>$, $<1,8> + <8,1>$, and $<1,16> + <16,1>$ in the first, second, third and fourth experiments, respectively. For all the experiments, we keep the total amounts of dominant resource identical in order to compare the impacts of heterogeneity on resource usage. Fig. 8 shows the results using SD, MR and SAMR with different heterogeneity. It can be seen that the required number of PMs increases as the heterogeneity increases in SD method but the number of PMs required in MR and SAMR falls with the increase of heterogeneity of the workloads. The reason is that large amounts of resources are wasted in SD, while MR and SAMR are capable to provide balanced utilization of resources. This phenomenon again shows the advantage of heterogeneous resource management for serving diversified workloads in IaaS clouds. The advantage becomes more obvious in SAMR which is specifically designed with skewness avoidance.

Impacts by delay threshold. Fig. 9(a) shows the results for varying the delay threshold D for Google trace. We use a set of delay threshold (minutes): $15, 30, 60, 90, 120$. We can see from the figure that the number of active PMs in each time slot reduces as we allow higher delay threshold. This is because a larger D value permits more requests in the waiting queue for powering up additional PMs, and thus the cloud system is able to serve more VMs with current active PMs. In practice, cloud providers is able to set an appropriate D to achieve a good balance between quality of service and power consumption.

Impacts by maximum VM capacitiy. In Fig. 9(b), we design an experiment on Google trace where the cloud providers offer different maximum VM capacity. For example, a cloud system with the normalized maximum resource m_i offers $(\log_2 m_i \cdot 100 + 1)$ options on resource type-i. We test three maximum resource values $16\%, 32\%, 64\%$, respectively. From the figure we can see that with bigger VMs offered by providers, more PMs are needed to serve the same amount of workloads. The reason is that bigger VMs have higher chance to be delayed when the utilization of resources in the data center is high.

Impacts by time slot length. Fig. 9(c) shows the results for varying slot length from 15 minutes to 120 minutes using Google trace. Our heterogeneous resource management allows cloud providers to specify time slot according to their requirements. As shown in the figure, the number of active PMs can be further optimized with smaller time slots. These results suggest that we can obtain better optimization effect if our proposed prediction model and PM provisioning can be executed more frequently. However, the model computation overhead prohibits a time slot being too small.

7 Conclusion

Real world jobs often have different demands on different computing resources. Ignoring the differences in the current homogeneous resource allocation causes resource starvation on one type and wastage on other types. To reduce the monetary costs for users in IaaS clouds and wastage in computing resources for cloud system, this paper first emphasized the need to have a flexible VM offering for VM requests with different resource demands on different resource types. We then proposed a heterogeneous resource allocation approach named skewness-avoidance multi-resource (SAMR) allocation. Our solution includes a VM allocation algorithm to ensure heterogenous workloads are allocated appropriately to avoid skewed resource utilization in PMs, and a model-based approach to estimate the appropriate number of active PMs to operate SAMR. Particularly for our developed Markov Chain, we showed its relatively low complexity for practical operation and accurate estimation.

We conducted simulation experiments to test our proposed solution. We compared our solution with the single-dimensional method and the multi-resource method without skewness consideration. From the comparisons, we found that ignoring heterogeneity in the workloads led to huge wastage in resources. Specifically, by conducting simulation studies with three synthetic workloads and one cloud trace from Google, it revealed that our proposed allocation approach that is aware of heterogenous VMs is able to significantly reduce the active PMs in data center, by 45% and 11% on average compared with single-dimensional and multi-resource schemes, respectively. We also showed that our solution maintained the allocation delay within the preset target.

References

[1] S. Genaud and J. Gossa, "Cost-wait trade-offs in client-side resource provisioning with elastic clouds," in *Proc. of 2011 IEEE International Conference on Cloud Computing (CLOUD'10)*. IEEE, 2011, pp. 1–8.

[2] E. Michon, J. Gossa, S. Genaud *et al.*, "Free elasticity and free cpu power for scientific workloads on iaas clouds." in *ICPADS*. Citeseer, 2012, pp. 85–92.

[3] P. Marshall, H. Tufo, and K. Keahey, "Provisioning policies for elastic computing environments," in *Proc. of 2012 IEEE 26th International Parallel and Distributed Processing Symposium Workshops & PhD Forum (IPDPSW)*. IEEE, 2012, pp. 1085–1094.

[4] L. Wang, J. Zhan, W. Shi, and Y. Liang, "In cloud, can scientific communities benefit from the economies of scale?" *IEEE Transactions on Parallel and Distributed Systems*, vol. 23, no. 2, pp. 296–303, 2012.

[5] R. V. den Bossche, K. Vanmechelen, and J. Broeckhove, "Cost-optimal scheduling in hybrid iaas clouds for deadline constrained workloads," in *IEEE CLOUD'10*, 2010.

[6] M. Malawski, G. Juve, E. Deelman, and J. Nabrzyski, "Cost- and deadline-constrained provisioning for scientific workflow ensembles in iaas clouds," in *Proc. of the International Conference on High Performance Computing, Networking, Storage and Analysis (SC'12)*. IEEE Computer Society Press, 2012, p. 22.

[7] K. Deng, J. Song, K. Ren, and A. Iosup, "Exploring portfolio scheduling for long-term execution of scientific workloads in iaas clouds," in *Proceedings of International Conference for High Performance Computing, Networking, Storage and Analysis (SC'13)*. ACM, 2013, p. 55.

[8] Amazon Pricing, https://aws.amazon.com/ec2/pricing/.

[9] Rackspace Cloud Pricing, http://www.rackspace.com/cloud/servers.

[10] L. A. Barroso and U. Hölzle, "The case for energy-proportional computing," *IEEE computer*, vol. 40, no. 12, pp. 33–37, 2007.

[11] Z. Xiao, W. Song, and Q. Chen, "Dynamic resource allocation using virtual machines for cloud computing environment," *IEEE Transactions on Parallel and Distributed Systems*, 2013.

[12] M. Lin, A. Wierman, L. L. H. Andrew, and E. Thereska, "Dynamic right-sizing for power-proportional data centers," in *INFOCOM'11*, 2011.

[13] Google Inc, http://code.google.com/p/googleclusterdata/.

[14] C. Reiss, A. Tumanov, G. R. Ganger, R. H. Katz, and M. A. Kozuch, "Heterogeneity and dynamicity of clouds at scale: Google trace analysis," in *Proceedings of the Third ACM Symposium on Cloud Computing*. ACM, 2012.

[15] L. Wei, B. He, and C. H. Foh, "Towards Multi-Resource physical machine provisioning for IaaS clouds," in *IEEE ICC 2014 - Selected Areas in Communications Symposium (ICC'14 SAC)*, 2014.

[16] T. J. Hacker and K. Mahadik, "Flexible resource allocation for reliable virtual cluster computing systems," in *Proc. of SC'11*, 2011.

[17] D. Villegas, A. Antoniou, S. M. Sadjadi, and A. Iosup, "An analysis of provisioning and allocation policies for infrastructure-as-a-service clouds," in *Proc. of 2012 12th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (CCGrid'12)*. IEEE, 2012, pp. 612–619.

[18] K. Mills, J. Filliben, and C. Dabrowski, "Comparing vm-placement algorithms for on-demand clouds," in *Proc. of CLOUDCOM'11*, 2011.

[19] E. G. Coffman Jr, M. R. Garey, and D. S. Johnson, "Approximation algorithms for bin packing: A survey," in *Approximation algorithms for NP-hard problems*. PWS Publishing Co., 1996, pp. 46–93.

[20] D. Xie, N. Ding, Y. C. Hu, and R. Kompella, "The only constant is change: incorporating time-varying network reservations in data centers," *ACM SIGCOMM Computer Communication Review*.

[21] A. Ali-Eldin, M. Kihl, J. Tordsson, and E. Elmroth, "Efficient provisioning of bursty scientific workloads on the cloud using adaptive elasticity control," in *Proc. of the 3rd workshop on Scientific Cloud Computing Date*. ACM, 2012, pp. 31–40.

[22] S. Niu, J. Zhai, X. Ma, X. Tang, and W. Chen, "Cost-effective cloud hpc resource provisioning by building semi-elastic virtual clusters," in *Proc. of International Conference for High Performance Computing, Networking, Storage and Analysis (SC'13)*. ACM, 2013, p. 56.

[23] J. Li, K. Shuang, S. Su, Q. Huang, P. Xu, X. Cheng, and J. Wang, "Reducing operational costs through consolidation with resource prediction in the cloud," in *Proc. of CCGRID'12*, 2012.

[24] M. Mao and M. Humphrey, "Auto-scaling to minimize cost and meet application deadlines in cloud workflows," in *Proc. of SC'11*, 2011.

[25] G. Chen, W. He, J. Liu, S. Nath, L. Rigas, L. Xiao, and F. Zhao, "Energy-aware server provisioning and load dispatching for connection-intensive internet services." in *Proc. of NSDI'08*, 2008.

[26] C. Delimitrou and C. Kozyrakis, "Qos-aware scheduling in heterogeneous datacenters with paragon," *ACM Transactions on Computer Systems (TOCS)*, vol. 31, no. 4, p. 12, 2013.

[27] W. Wang, B. Li, and B. Liang, "Dominant resource fairness in cloud computing systems with heterogeneous servers," in *INFOCOM'14*, 2014.

[28] A. Ghodsi, M. Zaharia, B. Hindman, A. Konwinski, S. Shenker, and I. Stoica, "Dominant resource fairness: fair allocation of multiple resource types," in *USENIX NSDI*, 2011.

[29] A. Ghodsi, M. Zaharia, S. Shenker, and I. Stoica, "Choosy: max-min fair sharing for datacenter jobs with constraints," in *Proc. of the 8th ACM European Conference on Computer Systems*. ACM, 2013, pp. 365–378.

[30] C. Joe-Wong, S. Sen, T. Lan, and M. Chiang, "Multi-resource allocation: Fairness-efficiency tradeoffs in a unifying framework," in *Proc. of INFOCOM'12*. IEEE, 2012, pp. 1206–1214.

[31] A. A. Bhattacharya, D. Culler, E. Friedman, A. Ghodsi, S. Shenker, and I. Stoica, "Hierarchical scheduling for diverse datacenter workloads," in *Proceedings of the 4th annual Symposium on Cloud Computing*. ACM, 2013, p. 4.

[32] Q. Zhang, M. F. Zhani, R. Boutaba, and J. L. Hellerstein, "Harmony: dynamic heterogeneity-aware resource provisioning in the cloud," in *Proc. of 2013 IEEE 33rd International Conference on Distributed Computing Systems (ICDCS'13)*. IEEE, 2013, pp. 510–519.

[33] Q. Zhang, M. Zhani, R. Boutaba, and J. Hellerstein, "Dynamic heterogeneity-aware resource provisioning in the cloud," *IEEE Transactions on Cloud Computing*.

[34] S. K. Garg, A. N. Toosi, S. K. Gopalaiyengar, and R. Buyya, "Sla-based virtual machine management for heterogeneous workloads in a cloud datacenter," *Journal of Network and Computer Applications*, 2014.

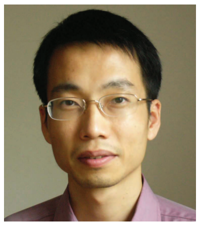
[35] P. Dhawalia, S. Kailasam, and D. Janakiram, "Chisel: A resource savvy approach for handling skew in mapreduce applications," in *Proc. of CLOUD'13*. IEEE, 2013.

[36] S. Di, D. Kondo, and W. Cirne, "Host load prediction in a google compute cloud with a bayesian model," in *Proc. of the International Conference on High Performance Computing, Networking, Storage and Analysis (SC'12)*. IEEE Computer Society Press, 2012.

Lei Wei received his bachelor's degree in 2008 from Dalian University of technology, China and the master's degree in 2011 from Institute of Computing Technology, Chinese Academy of Sciences, China. He is currently working toward Ph.D. degree at School of Computer Engineering of Nanyang Technological University, Singapore. His research interests include cloud computing and Media Computing.

Chuan Heng Foh received his Ph.D. degree from the University of Melbourne, Australia in 2002. After his PhD, he spent 6 months as a Lecturer at Monash University in Australia. In December 2002, he joined Nanyang Technological University, Singapore as an Assistant Professor until 2012. He is now a Senior Lecturer at the University of Surrey. His research interests include protocol design and performance analysis of various computer networks including wireless local area and mesh networks, mobile ad hoc and sensor networks, 5G networks, and data center networks. He has authored or coauthored over 100 refereed papers in international journals and conferences. He is a senior member of IEEE.

Bingsheng He received his Ph.D. degree in computer science in Hong Kong University of Science and Technology (2003-2008). He is an assistant professor in Division of Networks and Distributed Systems, School of Computer Engineering of Nanyang Technological University, Singapore. His research interests are high performance computing, distributed and parallel systems, and database systems. His papers are published in prestigious international journals and proceedings such as ACM TODS, IEEE TKDE, ACM SIGMOD, VLDB/PVLDB, ACM/IEEE SuperComputing, PACT, ACM SoCC, and CIDR. He has been awarded with the IBM Ph.D. fellowship (2007-2008) and with NVIDIA Academic Partnership (2010-2011).



Jianfei Cai received his Ph.D. degree from the University of Missouri-Columbia. He is currently an Associate Professor and has served as the Head of Visual & Interactive Computing Division and the Head of Computer Communication Division at the School of Computer Engineering, Nanyang Technological University, Singapore. His major research interests include visual computing and multimedia networking. He has served as the leading Technical Program Chair for IEEE International Conference on Multimedia & Expo (ICME) 2012 and the leading General Chair for Pacific-rim Conference on Multimedia (PCM) 2012. Since 2013, he has been serving as an Associate Editor for IEEE Trans on Image Processing (T-IP). He has also served as an Associate Editor for IEEE Trans on Circuits and Systems for Video Technology (T-CSVT) from 2006 to 2013 and a Guest Editor for IEEE Trans on Multimedia (TMM), ELSEVIER Journal of Visual Communication and Image Representation (JVCI), etc. He is a senior member of IEEE.