

Optimizing Resource Utilization of a Data Center

© 2016 IEEE. Personal use of this material is permitted. Permission from IEEE must be obtained for all other uses, in any current or future media, including reprinting/republishing this material for advertising or promotional purposes, creating new collective works, for resale or redistribution to servers or lists, or reuse of any copyrighted component of this work in other works.

This material is presented to ensure timely dissemination of scholarly and technical work. Copyright and all rights therein are retained by authors or by other copyright holders. All persons copying this information are expected to adhere to the terms and constraints invoked by each author's copyright. In most cases, these works may not be reposted without the explicit permission of the copyright holder.

Citation:

X. Sun, N. Ansari, and R. Wang, "Optimizing Resource Utilization of a Data Center," *IEEE Communications Surveys and Tutorials*, DOI: 10.1109/COMST.2016.2558203, vol. 18, no. 4, Fourth Quarter 2016.

URL:

http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=7458812

Optimizing Resource Utilization of a Data Center

Xiang Sun, *Student Member, IEEE*, Nirwan Ansari, *Fellow, IEEE*, and Ruopeng Wang

Abstract—To provision IT solutions with reduced operating expenses, many businesses are moving their IT infrastructures into public data centers or start to build their own private data centers. Data centers can provide flexible resource provisioning in order to accommodate the workload demand. In this paper, we present a comprehensive survey of most relevant research activities on resource management of data centers that aim to optimize the resource utilization. We first describe the resource overprovisioning problem in current data centers. Then, we summarize two important components in the resource management platform and present the benefit of accurately predicting the workload in resource management. Afterwards, we classify existing resource management in a data center into three categories: Virtual Machine (VM) based, Physical Machine (PM) based and application based resource management mechanisms. We discuss the performance degradation for implementing these three kinds of resource management in a heterogeneous data center. Finally, we present three important issues arisen in the data center resource management and some potential approaches to address the issues. This paper presents a timely survey on resource management in a data center, and provides a comprehensive reference for further research in this field.

Index Terms—Resource management, resource utilization, resource allocation, workload prediction, data center.

I. INTRODUCTION

INTERNET based commerce has been blooming and growing rapidly in recent years, and data centers are enabling IT solutions for facilitating e-commerce because of their great potential in reducing the operating expenses (*OPEX*) and management overheads, i.e., data centers provide a shared elastic computing infrastructure to different business tenants for hosting multiple applications. More specifically, data centers provision services in terms of Infrastructure as a Service (*IaaS*), Platform as a Service (*PaaS*) or Software as a Service (*SaaS*) to different tenants based on their demands. Meanwhile, the Service Level Agreement (*SLA*), a service contract assigned between a tenant and a data center provider (e.g., Amazon), is guaranteed by the data center among tenants. As a reward, tenants would make payment for renting the resources from data center providers via different charging models (e.g., pay-as-you-go model and reservation model). However, resource utilization of a data center is not efficient for managing on-demand applications. Studies [1], [2] have shown that servers (note that the terms “server” and “Physical Machine (*PM*)” are interchangeably used in the paper) in the data center are underutilized most of the time due to overprovisioning for the peak resource demand. Resource overprovisioning results in the energy inefficient problem in the data center that increases the data center provider’s budget

and CO_2 footprint [3]. Although server virtualization [4] by enabling resource multiplexing among Virtual Machines (*VMs*) in one server mitigates the resource overprovisioning problem, resource utilization in a data center is still rather poor. It is reported that the mean utilization of CPU, memory and disk on several thousands of servers, which were randomly selected from different data centers during the period from June 2009 to May 2011, were 17.76%, 77.93% and 75.28%, respectively [5]. Reiss *et al.* [6] collected the resource utilization traces of VMs in one cluster over the 29 day period from the Google Cluster Trace [7] and found out that the average CPU utilization is less than 60% and the average memory utilization is less than 50%. Obviously, the resources in a data center are overprovisioned most of the time, especially the CPU resource which consumes more energy than other resources. Therefore, adopting an agile resource management mechanism to accommodate the dynamics of application resource demand is critical in enhancing resource utilization without violating applications’ SLAs in the data center. The application resource demand refers to the number of VMs required to serve the application and the amount of resources (CPU cycles, memory, and network I/O) assigned for each VM. Instead of optimizing resource utilization, some studies manage the resources in order to minimize the energy consumption of the data center [8], [9], and others aim to reduce the operational cost of running a data center by making use of resource management [10], [11]. The resource management approach to achieving the three objectives is quite similar. In this paper, we focus on optimizing resource utilization, one aspect of resource management.

A resource management platform, normally, consists of two parts: Global Resource Manager (*GRM*) and Local Resource Manager (*LRM*) [12], [13]. *GRM* provides a global view of the resource provisioning strategy, i.e., *GRM* determines each VM’s location (VM that is hosted by a corresponding server) so that the physical server has sufficient but not superfluous resources to host the VMs. Live VM migration is applied to implement global resource management. *GRM* provisions coarse time scale resource management because of the complexity of running the global resource allocation algorithm. *LRM*, installed in each server, is to implement VM based operations (such as creating, starting, terminating and migrating VM), which are controlled by *GRM*. Meanwhile, *LRM* assigns server-based resources to the hosting VMs according to the information from *GRM* and dynamically adjusts the resources to its VMs based on their real-time resource demands. However, *LRM* is unaware of other servers’ information (such as other servers’ resource utilization) or application-based information (such as the performance of applications which are running on the other servers).

GRM comprises two components as shown in Fig. 1.

X. Sun and N. Ansari are with Advanced Networking Lab., Electrical & Computer Engineering Dept, New Jersey Institute of Technology, Newark, NJ 07102, USA. E-mail:{xs47, nirwan.ansari}@njit.edu.

R. Wang is with Huawei Technologies Co., Shanghai, China.

The workload demand predictor is to estimate the workload demand of the selected object for the next time period based on the historical data traces stored in the database. The object can be VM, server or application; the workload demand of a different object may incur different resources, i.e., if GRM selects VM as the object, then the VM workload demand normally refers to the resource (CPU cycles, memory, and network I/O) utilization of the VM in the dedicated server [15]; if GRM selects server as the object, then the server workload is defined as the sum of the resource utilization of all VMs [17], and the application workload is referred to as the average number of application requests arrived in the time period [18]. Also, owing to the different characteristics of objects' workload data traces, choosing different objects results in adopting different prediction models to estimate the objects' future workload demands. The Global Resource Allocator (*GRA*) in Fig. 1 is to assign resources to every object based on its estimated workload demand and to map the estimated resources of the objects to different servers so that the total resource utilization of servers in the data center is maximized and applications' SLAs are guaranteed. The *GRA* in GRM would adopt a different strategy in estimating the workload of choosing a different object. Often, LRM does not implement the workload demand predictor because the prediction algorithm is computationally intensive and running the prediction algorithm on each server will drain the server resources. However, LRM can trace the workload of the object or sense the object's performance, and utilizes the information to fulfill local resource provisioning by the Local Resource Allocator (*LRA*) in LRM.

Workload demand prediction is critical for resource management for two reasons. First, the workload of an object is fluctuating over time, and the resource allocation strategy in the current time period may not be suitable in the next time period. Second, a resource allocation strategy may incur VM migration, which takes time to complete the process [19]. So predicting the workload demand is necessary to proactively manage resources of the objects. Zhen *et al.* [12] showed that resource management with workload prediction outperforms resource management without workload prediction. The accuracy of the workload demand prediction algorithm is a very important factor in determining the performance of the whole resource management mechanism. A huge bias of workload prediction leads to a big deviation in resource demand estimation of the objects that may result in resource overprovisioning or underprovisioning.

Based on the workload demand of the objects to be predicted, we can category the resource management mechanism into three types: VM, PM, and application workload prediction based resource management mechanisms, respectively. We structure the rest of the paper as follows. Sections II, III, and IV present the VM, PM, and application workload based resource management mechanisms, respectively. Section V discusses the performance degradation of implementing these three types of resource management in a heterogeneous data center. Section VI presents several issues that arise in the data center resource management and some potential approaches to address the issues. Section VII summarizes and concludes the

survey.

II. VM WORKLOAD BASED RESOURCE MANAGEMENT

VM workload based resource management is to predict each VM workload based on its historical workload data traces and to assign the necessary resources to every VM based on the estimated workload. The architecture for realizing the VM based resource management mechanism is shown in Fig. 1. The server in Fig. 1 is implemented by Xen based hypervisor. LRM, which is located in Domain 0, traces the workloads of all VMs in Domain U and reports them to GRM. VM workload based resource management strategy consists of two parts: the VM workload prediction algorithm, which is run in the GRM's VM workload predictor component, and the VM workload based resource allocation method, which is implemented by *GRA* in GRM and *LRA* in LRM.

A. VM workload prediction

Normally, the VM workload is defined as the average utilization of different types of resources of the VM in its hosting server during a fixed time period. The types of resources include CPU cycle, memory, and network I/O resources. So, VM workload based prediction executed in GRM is to estimate the resource utilization of each VM in the server for the next time period based on each VM's historical workload data traces.

Jheng *et al.* [20] argued that the historical VM workload data traces from the same day have weak correlation with the current VM workload, but there is a strong correlation between the current VM workload and the workload from the same time in the previous week. The Grey forecasting model is applied to predict the tendency of CPU, memory and hard disk utilization of VMs.

Kashifuddin *et al.* [21] proposed to use chaotic theory to predict VMs' workloads. They argued that the VM's workload data traces may not exhibit a cyclic pattern, i.e., the same or similar workload pattern may not appear in a fixed period, and the VM's workload cannot be accurately predicted for a long term, i.e., the prediction error is exponentially increasing with respect to the prediction time. By proving that the workload follows a Chaotic time series, they proved that the maximum Lyapunov Exponent value of the time series data samples from the Google Cluster Data Trace [7], NASA [22] and the World Cup [22] data trace are all positive [23]. Based on the proof, chaos theory is applied to predict each VM's workload in terms of the VM's CPU utilization in the future by using the historical data trace of each VM. In the experimental evaluation, the proposed VM workload based prediction algorithm estimates the CPU usage of a single VM in 5-minute intervals with Mean Square Error (*MSE*) between 0.005 and 0.015 (depending on different data traces run on the VM) that outperforms other prediction methods, i.e., Fourier Transform with Sliding Window [24] and Wavelets with Markov Chains [25].

Based on the Exponentially Weighted Moving Average (*EWMA*) model, Zhen *et al.* [12] predicted the CPU, memory and network I/O resource demand of each VM for running

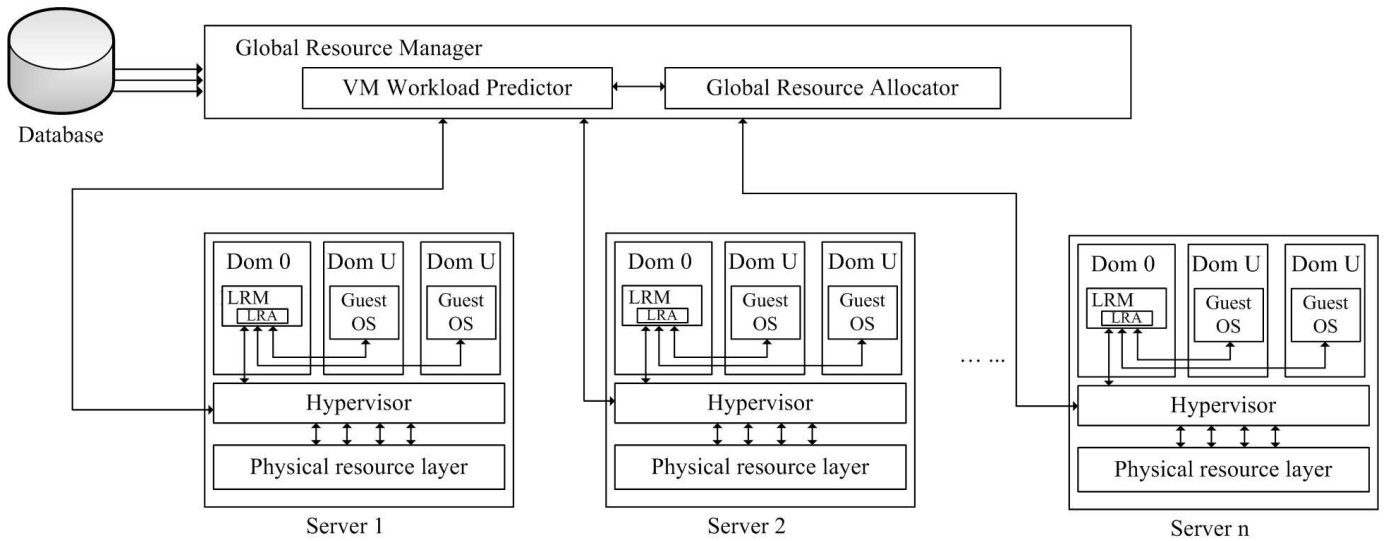


Fig. 1. VM workload based resource management platform.

Internet applications. Meanwhile, in order to predict the VM resource demand conservatively, they proposed the Fast Up and Slow Down (*FUSD*) prediction algorithm by adapting the smooth coefficient of the EWMA model when the observed resource demand is going up or down, i.e., *FUSD* algorithm would predict the resource demand a little higher than the expected whenever the observed resource demand is going up or down (the predicted resource demand is increasing faster than the observed resource demand when it is going up and the predicted resource demand is decreasing slower than the observed resource demand when it is going down) to reserve more resource for the demand. The experimental results show that by choosing proper system parameters, 77% of the predicted CPU utilization is higher than the observed one with the mean error of 9.4%.

Arijit *et al.* [26] showed that each application running in the cloud can be decomposed into one or more components in terms of jobs and each of which is served by a collaborative set of VMs. By analyzing the discretized time series data, they found that the workload data traces of VMs from the same collaborative set tend to vary in a collated fashion. So, they analyzed the historical time series data of workload traces of all VMs and grouped the VMs as a cluster in which all VMs show the similar recurring workload patterns. Instead of predicting the VM's workload individually, the proposed prediction is based on per cluster, thus eliminating the noise and randomness in individual VM workload measurements. Meanwhile, they pointed out that not all the clusters' workload demands are predictable since the predictability of the VMs in the same cluster varies. The cluster-based autocorrelation function and time-lagged cross-correlation function are introduced to determine the predictability of the cluster, i.e., whether the workload of a cluster can be predicted by its historical data trace or by at least one other cluster's workload data traces. Hidden Markov model is applied as an estimation model for a group of clusters (i.e., every cluster in the group is self-predictable or predictive of other clusters in the group) to

foresee the CPU utilization of all VMs in the group. The experiments are based on 21 days of CPU utilization time series collected from one enterprise customer. The results show that 91% of the VMs' workload in the cluster can be correctly predicted within 10% error and the VMs with higher workloads in the cluster achieve higher prediction accuracy (approaching approximately 100%) than VMs with lower workloads.

Chen and Shen [27] also drew a similar conclusion that VMs from the same tenant (i.e., VMs collaboratively serve the same application) exhibit the similar resource utilization time series. Meanwhile, the resource demands of VMs exhibit a daily periodical pattern. They analyzed the Google cluster trace [7] and PlanetLab trace [28] to prove the conclusion. Based on these two characteristics of VMs' resource utilization time series, they estimated the future resource utilization of VMs by detecting the resource demand patterns (i.e., finding the smooth envelop of the resource utilization time series) of a set of VMs (i.e., VMs serve the same application) in the past 24 hours.

Bobroff *et al.* [29] argued that not all the VMs' workloads are predictable, i.e., the prediction of some VMs' workloads may result in relatively larger prediction errors (i.e., the width (standard deviation) of the prediction error distribution is larger than the width of the workload distribution). The larger prediction error brings about the misleading of VM resource management later on. They pointed out that if a VM's workload time series is lack of periodicity or suffers a quickly decaying autocorrelation function (i.e., the VM's workload undergoes random fluctuations over time), then proactive resource management applied into these type of VMs is meaningless.

The VM workload prediction module can be easily implemented because it does not need to acquire any application-based configuration. The monitoring engines (e.g., XenMon [31] and virtual firewall-router [4]) in each server's hypervisor periodically monitor its VMs' resource utilization and report

it to GRM. However, VM's workload exhibits non-stationarity over time, i.e., VM's workload changes unpredictably with time, especially, the VMs serving some short-running tasks (e.g., DAG-of-tasks in MapReduce system). In other words, the ownership of a VM varies over time (i.e., a VM may be rent by different tenants over a certain time period), and thus the tasks running in the VM may also vary over time. Different tasks incur different computational complexities, hence leading to the resource utilization of a VM randomly fluctuated over time, and so the VM resource utilization may be unpredictable. Some long-running tasks exhibit predictable features (e.g., periodicity), but VMs' resource utilization may also be poorly estimated in some scenarios. This is because each VM's resource utilization not only depends on the application users' activities (i.e., the number of requests for running the tasks in the VM), but also relies on other factors, such as the workload scheduling strategy. For instance, one VM may cooperate with other VMs in serving the same application, and so the dynamic workload scheduling strategy can affect the individual VM's workload. Therefore, it is a big challenge to accurately predict the VM workload. In order to notify and capture the dynamic changes of VMs' resource demands in time, VM workload prediction models need to update the model parameters frequently. Meanwhile, there is a tradeoff between the prediction period (i.e., the duration allowed to predict the VM workload) and the accuracy of prediction results, and so the prediction period is setup relatively small (e.g., several minutes or hours) to guarantee the accurate prediction results. Furthermore, the VM resource utilization reflects the VM workload, and thus the VM workload can be depicted as resource utilization of the VM in the server. However, if the servers in the data center are heterogeneous (the servers with different resource capacities), the VM resource utilization prediction model, applicable for the current server, may no longer work for the server to which the VM has migrated. Thereby, the accuracy of the prediction algorithm degrades considerably. The premise of server homogeneity imposed by the above methods impedes their practical deployments.

B. VM workload based resource allocation

Recent resource management platforms, like VMware DRS [32], Microsoft PRO [33], HP PRM [34] and IBM PLM [35], are all VM workload based resource management. They are trying to achieve better performance isolation (the resource consumption of one VM should not impact the promised guarantees of other virtual machines on the same hardware) and provide a platform to dynamically allocate resources to VMs based on the VMs' resource utilization or the static shares so that various resource allocation strategies can be implemented. Normally, VM workload based resource allocation is implemented by two means: LRA in GRM and GRA in LRM. LRA provides fine-grained resource allocation by utilizing real time resource utilization information and the information provided by GRM. GRA, on the other hand, provides coarse-grained resource allocation based on the estimated resource utilization of VMs.

1) *VM workload based Global Resource Manager*: The optimal resource allocation, as mentioned before, is to maximize resource utilization without violating applications' SLAs. However, VM workload based GRA is agnostic to the application level information, implying that GRA is unaware of the application's performance (such as the average application response time and the average application throughput). In order to account for the application level aspect, an assumption is made that applications' SLA are violated whenever the server is overloaded, i.e., the summation of resource utilization of VMs housed within the same server reaches a predefined threshold [12], [19], [36]. So, in order to guarantee SLA, VM workload based resource allocation should guarantee the total resource utilization of the server to be less than the threshold; otherwise, VM migration is triggered to move the workload to the lightly loaded servers or a new server. On the other hand, if the estimated resource utilization of the server (the summation of resource utilization of VMs in the server) is too low, server consolidation [12], [37], [38] is enabled to enhance the resource utilization.

Studies [19], [36] have proposed the similar ideas of VM migration strategy which tries to detect underprovisioned servers (i.e., the server's resource utilization is higher than a threshold) and determine a suitable VM in the underprovisioned server to be migrated to an overprovisioned server, which has enough space to host the VM without becoming an underprovisioned server. Wood *et al.* [19] chose the VM with smaller volume value (i.e., lower resource demand) to be migrated from the underprovisioned server to the overprovisioned one, while Farahnakian *et al.* [36] selected the VM with the minimum migration time (the migration time is determined by the VM's memory size and available network bandwidth [49]), which is more suitable for the live VM migration among data centers (i.e., wide area network VM live migration).

Eliminating the number of underprovisioned servers cannot solve the problem that the resources in the data center are underprovisioned, i.e., the number of awaked servers in the data center cannot satisfy the resource demands of applications, and so new servers should be woken up. Chen and Shen [27] proposed a complementary VM allocation (CompVM) mechanism, which tries to optimize the location of each VM in each time slot so that the number of awaked servers is minimized. In other words, CompVM tries to find a VM allocation solution such that each VM is assigned to the server, which can satisfy its resource demands, and the multiple dimensional resource utilization of all the awaked servers is minimized. Bobroff *et al.* [29] also proposed the similar VM allocation mechanism. They formulated the VM allocation problem as a one-dimensional bin-packing problem (only CPU is considered as the resource of interest) and applied the first-fit heuristic algorithm to derive the optimal VM placement in each time slot. Recalculating the optimal location of every VM in each time slot can maximize the resource utilization of the data center, but the complexity of running the algorithm in a large data center is a big challenge. Moreover, recalculating the optimal location of every VM in each time slot may incur the unnecessary migrations (i.e., the cost of migration is larger than the benefit of migration) and oscillations (i.e., some VMs

migrate back and forth in every time slot).

Zhen *et al.* [12] defined two kinds of server groups according to the predicted server's resource utilization (CPU utilization, memory utilization, and network I/O utilization): if all types of predicted resource utilization of the server are below a cold threshold (a lower resource utilization threshold), then the server is enlisted in the cold spot group; if any type of predicted resource utilization of the server is above a hot threshold (a higher resource utilization threshold), then the server is enlisted in the hot spot group. The predictor shown in the resource manager architecture (Fig. 2) is to estimate the resource utilization of each VM. The Hotspot Solver tries to find a migration strategy for every server in the hot spot group so that the utilization of any of the server's resource is below a hot threshold, i.e., none of the server is overloaded. The Coldspot Solver is triggered when the average resource utilization of all active servers is below a green computing threshold, which indicates that the resource is fully overprovisioned. The Coldspot Solver tries to move all the VMs from the sever, which are in the cold spot group, to the servers whose resource utilization is below a warm threshold (a median resource utilization threshold indicates that the sever is running with a certain resource utilization level below that of the hot spot group) so that the original cold servers can go into the standby mode. All the migration decisions are enlisted into the migration list and executed in the first come first serve fashion. Therefore, in order to guarantee the SLA, all the servers in the hot spot group should be removed by VM migration, meanwhile, in order to improve the resource utilization, the number of servers in the cold spot group should be minimized. Beloglazov and Buyya [30] provided the similar idea to determine the server is under-utilized (i.e., the server's utilization is lower than the minimum threshold) or over-utilized (i.e., the server's utilization is higher than the maximum threshold), and they tested the total energy consumption of the data center as well as the SLA violation rate by selecting different values of minimum and maximum threshold.

2) *VM workload based Local Resource Manager*: Owing to the long time interval required for performing global resource management and dynamic changes of VM's resource demand over time, more flexible resource adjustment in the local server can improve resource utilization and application QoS significantly. In other words, VM workload based GRM determines the suitable servers to host VMs in the data center and LRM tries to satisfy the real-time resource demands of VMs in a dedicated server. One simple approach for assigning resources to local VMs is static allocation, i.e., local resource management assigns the weight to different VMs over time slots and allocate the amount of resources to VMs which are proportional to their weights, meanwhile, each VM's weight is equal to the resource utilization estimated by GRM [12]. Yet, static allocation becomes inefficient if VMs' loads vary over time. Therefore, a more efficient resource allocation strategy in the LRM needs to be designed. The optimal local resource allocation strategy is to achieve high server's resource utilization, reduce applications' response time and fairly assign resources to VMs in the server. Different types of resource

adopt different resource scheduling strategies and we will discuss them separately in the following.

a) *CPU resource scheduling*: A CPU resource allocator (located in Domain 0 in Xen based server), i.e., a CPU scheduler, is to assign the CPU resource to different VMs in the same server. There are three kinds of CPU schedulers provided by Xen: Simple Earliest Deadline First (*SEDF*) [40], Borrowed Virtual Time (*BVT*) [41] and Credit Scheduler [42]. In the *SEDF* scheduler, each job is given two parameters, i.e., period P_i and slice S_i (where i is the index of jobs), and the *SEDF* scheduler tries to guarantee job i by assigning the job at least S_i amount of time in a period of P_i . So, the *SEDF* scheduler defines the priority of a job as the deadline of the job, i.e., the time at which the job's period ends. Based on that, the *SEDF* scheduler provides a dynamic priority real-time scheduling policy among jobs from different VMs. More specifically, *SEDF* maintains a preemptive queue and schedules jobs according to their dynamic deadlines. Tseng and Huang [43] pointed out that *SEDF* performs badly when a server is in the overload condition, i.e., all the jobs running in the server may miss their deadlines (domino-effect of missed deadlines). They proposed to execute the Deadline Monotonic (*DM*) scheduling, i.e., a fixed-priority preemptive scheduling algorithm, in a CPU scheduler when the server is overloaded and the CPU scheduler chooses the *SEDF* scheduling when the server is not overloaded. This is because the *DM* scheduling can guarantee higher priority jobs in meeting their deadlines at the expense of lower priority jobs in missing their deadlines [44], i.e., even when the server is overloaded, the *DM* scheduling can guarantee the SLAs of higher priority jobs rather than violating all jobs' SLAs.

Duda and Cheriton [41] proposed a novel *BVT* scheduling algorithm for either uniprocessor or multiprocessors platform. In the long term, the *BVT* scheduling tries to share the CPU usage among jobs proportionally by their weights. On the other hand, in the short term, some latency-sensitive jobs are allowed to warp back in their virtual time (i.e., execution time of jobs) so that the jobs can be served earlier, i.e., the latency-sensitive jobs can borrow virtual time from their future CPU usage. By implementation, the *BVT* scheduler sorts the jobs with their virtual time and dispatches them with the smallest virtual time first.

The Credit Scheduler [42], [45] is the default CPU scheduler in a Xen-based server. It tries to automatically balance the load from VMs across all available physical CPUs on a symmetric multiprocessor server. Each VM is assigned a weight and a cap. The cap is an absolute value defining the amount of CPU resource that one VM receives (i.e., one VM's CPU capacity) and the weight is a relative value which is proportional to the CPU resource that the VM receives. In each time slice, the scheduler transforms the weight into a credit allocation for each VM. Once a VM is assigned the CPU resource, its credits will be consumed. The priority of a VM can be one of two values: over or under, which indicates the VM's credit is depleted or not, respectively. So, if the VM is out of credits, it only runs when other higher priority VMs (i.e., VMs with credits remaining) have completed their execution.

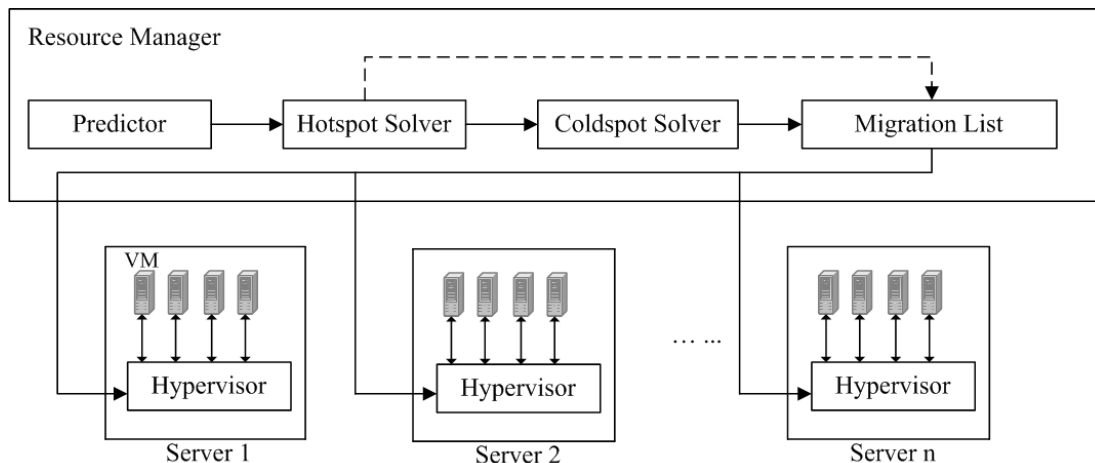


Fig. 2. VM workload based resource manager architecture [12].

b) Memory resource scheduling: In order to share a server's memory among its hosting VMs, the hypervisor of the server virtualizes the physical memory by adding an extra level of address translation, i.e., a mapping function between a virtual address (which is used by VMs) and a physical address (which is a software abstraction in order to provide the hypervisor with the status of the hardware memory). In general, there is a tradeoff between VMs' performance isolation and efficient memory utilization in the memory resource scheduling, i.e., static memory scheduling (reserve the amount of memory to VMs to be applied in the Xen-based server [4]) performs better performance isolation and incurs lower control overhead, but it may result in lower memory utilization and VMs' performance degradation. On the other hand, dynamic memory scheduling among VMs can improve the memory utilization since the scheduler can automatically assign the memory to VMs to accommodate their memory demands, but the performance isolation may be violated. Different from the CPU resource, which can be immediately assigned to the jobs according to the real-time information (such as the deadlines of the jobs), adjusting the memory size to the hosting VMs needs to modify the physical memory address among the VMs, which is not as flexible as the CPU resource scheduling. Thus, normally, the memory resource scheduling period is longer than the CPU resource scheduling period and the memory resource scheduler relies on the historical memory usage of the hosting VMs (i.e., non-real time information) to estimate the VMs' future memory usage.

Waldspurger [46] proposed a ballooning technique in the memory management in order to achieve efficient memory utilization and guarantee memory performance isolation. Each VM in a server is installed a balloon module, which is to allocate the VM's pages and map them into physical memory. The hypervisor of a server can increase or reclaim the memory of one VM by implementing balloon inflating or deflating operation. The memory allocation strategy in the ballooning technique is based on each VM's memory usage and its share-based entitlement, i.e., if the VM has less idle memory and shares more memory pages with other VMs (content-based page sharing is provisioned in the ballooning technique, i.e.,

if some VMs try to access the same contents in the memory pages, one of the VM can share its memory pages with others), it would be assigned more memory in the next time slice, and vice versa.

Heo *et al.* [47] proposed a dynamic memory allocation strategy based on the memory usage of each VM. Initially, the hypervisor set two values (i.e., U^{max} and U^{min}) to ensure the minimum and maximum amount of memory assigned to each VM. The memory usage of each VM is monitored and once the VM's memory usage is below a predefined threshold in the previous time slice, the hypervisor would revoke its free memory and assign it to the other VMs. Basically, the hypervisor dramatically assigns extra memory to VMs, which are underprovisioned, and slowly revokes the idle memory of VMs, which are overprovisioned.

Lu and Shen [48] presented that VMs' memory usage may not be related to VMs' performance, i.e., the memory usage based allocation strategies may probably degrade the VMs' QoS. They proposed to use the VM page miss ratio (the number of page misses under the new VM memory allocation divided by that under its baseline allocation) as a parameter to determine the VM memory allocation and guarantee the VM's performance consequently. Specifically, each VM is assigned a baseline memory allocation initially (baseline memory allocation ensures each VM's bound performance). The hypervisor of a server allocates the remaining memory resources to VMs so that the overall page misses (i.e., the geometric mean of each VM's page miss ratio) is reduced. As a consequence, the VM with a higher page miss ratio would obtain more memory resource as compared to the VM with a lower page miss ratio.

c) Network I/O resource scheduling: Network I/O virtualization technologies enable VMs share the network resource of a server to improve the network I/O utilization and provide flexible connectivity. In general, the network I/O virtualization architecture is shown in Fig. 3 where each VM has at least one Virtual Network Interface Card (vNIC) to communicate with other VMs. Virtual bridge (such as Open vSwitch [49], VMware's vNetwork distributed vswitch [50] and Cisco's Nexus 1000V [51]) is a software layer located between vNICs of VMs and Physical Network Interface Cards

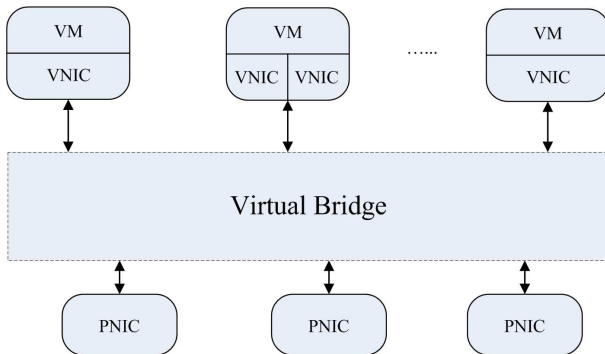


Fig. 3. Network I/O virtualization Architecture.

(*PNICs*). It provides packet forwarding between VNICs and PNICs based on forwarding tables. Also, it supplies flexible connectivity management (e.g., Open vSwitch supports both VLANs and GRE tunnels [52]), traffic statistics collection and QoS enforcement, etc. One of the challenges in the network I/O virtualization technologies is to allocate the network resource to VMs so that 1) VMs can fairly share the network resource, 2) the network resource usage is improved, and 3) the minimum QoS is guaranteed among VMs [53], [54]. However, different from the CPU and memory scheduling, scheduling the network resource in a server is not sufficient to achieve the goals mentioned above, because the traffic of a VM may go through not only PNICs in a server but also switches in a data center, i.e., if only the allocation strategy at the access point (server side) is considered, the network resource will be overprovisioned in a server's PNICs (because the bottleneck would be at the aggregation or core switches in the data center). Popa *et al.* [53] presented that it is impossible to achieve network-level proportional resource allocation (i.e., end-to-end data rate is proportional among different flows based on the flows' weights), guarantee minimum QoS of each VM and maximize network resource utilization simultaneously, because there are two kinds of tradeoffs among these three goals, i.e., 1) the tradeoff between network-level proportional resource allocation and high utilization, and 2) the tradeoff between network-level proportional resource allocation and minimum QoS guarantee of each VM. Therefore, designing an optimal network I/O resource scheduling in a data center is very challenging.

By applying the models proposed in [55]–[57], it is easy to achieve flow-based network resource fair-sharing in a single link (i.e., fair-sharing the network resource in a single server or a single switch). The authors in [58]–[60] tried to fairly assign the network resource to flows in a congested link by notifying the source VMs to fairly reduce their transmission rates. Sun and Ansari [61] proposed the persistence proportional sharing at network-level (*PPS-N*) algorithm by fairly assigning the bandwidth (i.e., network resource) to flows at the congested links. On the other hand, rather than providing fair resource allocation, Ballani *et al.* [62] proposed a virtual cluster structure in order to guarantee the network performance among different tenants (i.e., minimum QoS guarantee) by reserving the bandwidth to VMs. Guo *et al.* [63] assumed that all the

aggregate and core switches in a data center are non-blocking, and so they tried to design a server-based fair bandwidth allocation. They modeled the bandwidth competition among VMs as a cooperative bargaining game, and tried to 1) guarantee the minimum bandwidth allocation to VMs based on their base bandwidth requirement, and 2) proportionally share the remaining bandwidth based on their weights. Pascal and Gulati [64] also considered a server-based fair bandwidth allocation, but they considered not only the minimum QoS guarantee and proportional sharing, but also tried to provide performance isolation among VMs in a server by limiting the maximum bandwidth allocation to VMs. To improve the network resource utilization, Sun and Ansari [61] proposed the bandwidth efficiency persistence proportional sharing in network level (*BEPPS-N*) algorithm. By allowing each PNIC running in the work-conserving mode, the algorithm tries to assign the bandwidth to all the flows in a link as much as possible, thereby achieving max-min fairness. Raiciu *et al.* [65] demonstrated that enabling the multipath forwarding¹ in a data center network can significantly increase the throughput and resource utilization of the network.

Note that the network I/O resource scheduling strategies mentioned above are to allocate the bandwidth to the VMs once the location of each VM is determined (i.e., each VM has already been assigned to the corresponding PM). However, the locations of the VMs can significantly affect the performance of the network resource scheduling strategy. For instance, suppose there are two VMs, i.e., VM 1 and VM 2, and each PM can only host one of the two VMs; meanwhile, assume the traffic demand from VM 1 to VM 2 is 1 unit and the residual link capacity of the rack switch and aggregate switch is 1 unit and 0.9 unit, respectively. As shown in Fig. 4(a), for a bad VM placement, the network will be congested on the aggregate links no matter what kind of network I/O resource scheduling strategy is applied. For a good VM placement, as shown in Fig. 4(b), QoS of each VM will be satisfied. In Section VI-A, we will discuss resource allocation by jointly optimizing the VM placement and bandwidth (i.e., network I/O resource) scheduling in detail.

III. PM WORKLOAD BASED RESOURCE MANAGEMENT

PM workload based resource management is to predict each PM's workload based on its historical data trace and perform PM based resource allocation. The platform of PM workload based resource management is similar to that of VM workload based resource management (Fig. 1), but the LRM only traces its PM's workload rather than all VMs on the PM. PM workload based resource management also consists of two parts: PM workload prediction and PM workload based resource allocation.

A. PM workload prediction

PM workload is defined as the average resource utilization of a PM during a fixed time period. Types of resources

¹Multipath forwarding refers to routing the traffic load of the VM pair through multiple paths in order to balance the load among the paths.

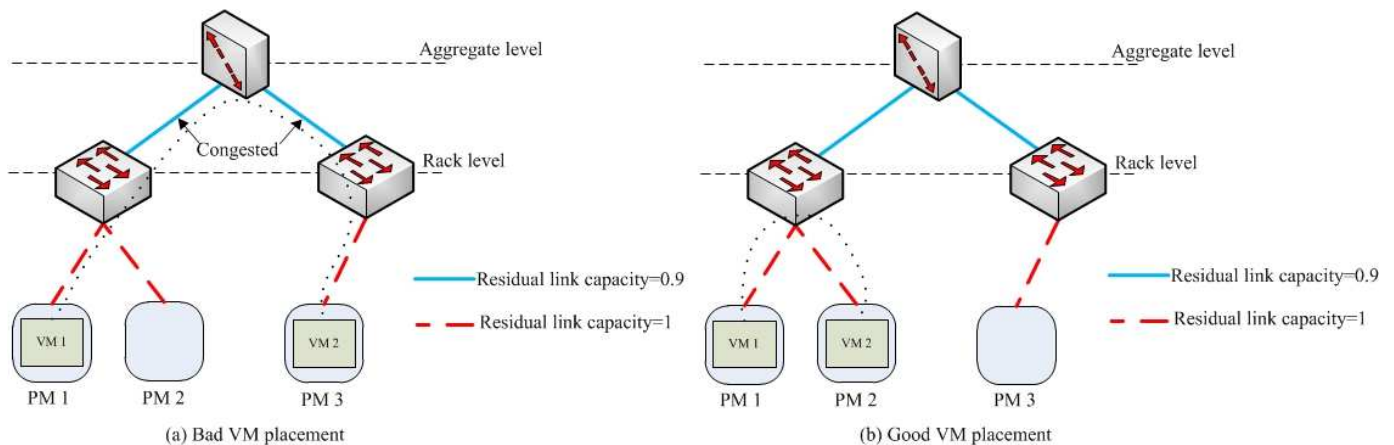


Fig. 4. The illustration of effects different VM placement methods on the performance of VMs.

include CPU cycle, memory, and network I/O resources. So, PM workload based prediction is to estimate the resource utilization of the PM for the next time period.

Nidhi and Rao [16], [17] argued that the characteristics of a PM's workload in a data center is dynamically changing over time because the PM workload is considered as the synthetic workload of hosting VMs and live VM migration happens frequently in the data center. Therefore, a specific workload prediction algorithm can hardly predict accurately all the time, i.e., different prediction models and parameters may need to be adopted in order to accommodate the new characteristics of the PM's workload. The authors proposed an ensemble learning approach to predict the PM's workload quickly and accurately. The basic idea of the approach is to form a set of base workload learners, i.e., individual workload prediction algorithms, to estimate the PM's workload based on the historical data separately. Then, the final workload prediction value is calculated as the weighted average of predictions of all learners in the set. The learner's weight is updated periodically by computing the difference of the actual value with the estimated value, and the learner whose estimated value is closer to the actual value will get a larger weight, and vice versa. The experimental result shows that the proposed approach can achieve an accuracy of 87.8%.

Similar to the VM workload, the PM workload also exhibits non-stationary feature over time, due to reasons like changes of the service function of a PM (e.g., a file server is reconfigured as a web server), etc. LRM in the PM workload prediction only needs to upload the information about resource utilization of the PM rather than every VM in the PM, and thus alleviates the network traffic for uploading the resource utilization data traces to GRM. In addition, the computational load of the prediction algorithm is lighter since the number of PMs in the data center is much less than the number of VMs.

B. Server workload based resource allocation

GRM does not have insight on the application's performance because only the resource utilization of all PMs is uploaded to GRM and there is no solid evidence showing the relationship between the PM resource utilization and the application's SLA.

Thereby, an assumption is made such that when the PM's average resource utilization approaches a predefined upper bound (e.g., 90%), the applications running on the PM are considered to have violated their SLAs. Therefore, similar to the VM workload based resource mechanism, GRA in the PM workload based resource allocation mechanism tries to avoid triggering the PM overload in the data center to guarantee applications' SLAs.

Although PM workload based resource management does not need to monitor every VM's resource utilization in a PM and thus reduces the total control overhead from hypervisor, it is difficult to design the PM workload based resource allocation strategy because GRA in PM workload based resource management is VM-agnostic, i.e., GRA is unaware of the VM-level information, and so traditional live VM migration strategies to minimize the number of awaked PMs cannot be implemented in the PM workload based resource management.

Nidhi and Rao [66], [67] tried to minimize the energy consumption in a PM cluster by shutting down the low-utilized PMs or switching them to low-power-mode for a period if applications running on these PMs are non-critical. Gmach *et al.* [68] predicted the workload traces from different PMs and checked whether these workload traces can be consolidated into a smaller number of PMs, i.e., the total resource utilization is improved by awaking a smaller number of PMs.

For the local resource allocation, PM workload based LRM can adopt the same strategies proposed in the VM workload based LRM.

IV. APPLICATION WORKLOAD BASED RESOURCE MANAGEMENT

Application based resource management is to predict each application's workload and assign the necessary resource to the application based on the estimated workload so that the application's SLA can be satisfied. Application workload based resource management platform is shown in Fig. 5. Normally, an application running in a data center comprises different application-tiers. An application-tier can be considered as an individual component or function in the application. For instance, an application can be separated into three tiers,

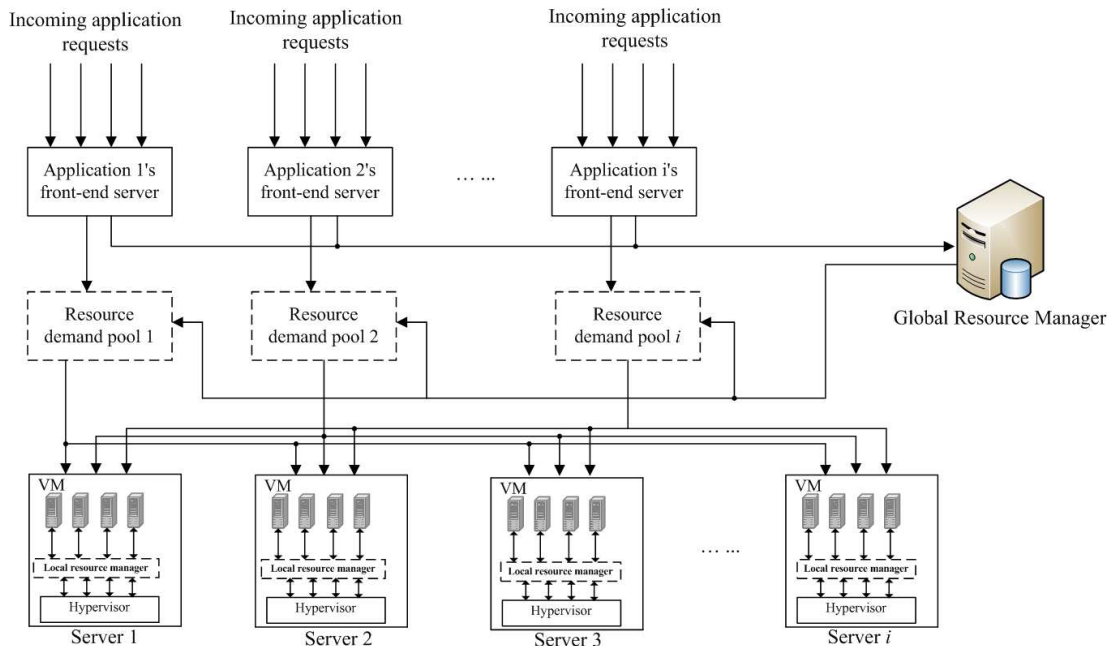


Fig. 5. Application based resource management platform.

which are running various tasks in different VMs: a front-end tier (e.g., a HTTP server), a client-server processing tier, and/or an enterprise database tier [69], [70]. The front-end tier receives the application requests from users, and the requests are processed in the dedicated back-end tier (including client-server processing tier and/or an enterprise database). Normally, all the application requests go through the front-end tier, and so the front-end tier has to sense the application workload and report the workload data traces to GRM. GRM, which has the same two modules mentioned in Fig. 1, analyzes every application/application-tier workload data trace, predicts the future workload and decides the minimum amount of resource demand of each application/application-tier based on the estimated workload. Each application/application-tier has a resource demand pool, which defines the minimum amount of resource demands. Afterwards, GRM maps all the resource demand pools into physical servers in order to maximize the total resource utilization in the data center. LRM is also applied in the platform to optimize the server resource utilization in a fine scale time domain.

A. Application workload prediction

The application workload is referred to as the number of incoming requests of the specific application. Recently, many studies focused on predicting the application workload in the data center. Prevost *et al.* [71] assumed that each application is accessed via a set of VMs, which are assigned different IP addresses, and predicted the number of requests for each IP address. They used FIR Wiener Filter to predict the workload of web servers (e.g., WWW server at NASA) by utilizing the historical workload data traces with a sliding window of size N . FIR Wiener Filter is a tool to minimize the mean square error between the actual workload and predicted

workload during look-ahead interval by solving the Yule-Walker equations. Wei *et al.* [72] argued that the application workload data trace in the data center exhibits periodical nature, and so they applied the ARIMA model to do short-term application workload prediction; the results showed that there is less than 10% underestimate or overestimate prediction error on average. Tom *et al.* [73] formulated the web server's workload in the data center as a time series, which exhibits short-term random fluctuations. However, the time series also expresses prominent periodic patterns in a diurnal cycle. They provided both long term (in days) and short term (in minutes) application workload predictions. For the long term prediction, the load is modeled as a dynamic harmonic regression. For the short term prediction, the autoregressive model is applied for the workload prediction.

Other than the server and VM workload prediction, the application workload does not predict the resource utilization directly. The number of application/application-tier requests is much easier to be estimated than VM and server workload since the application workload depends on human activities and exhibits periodical characteristic over time. Moreover, the application workload prediction algorithm is unaware of the hardware deployment in the data center, i.e., the server heterogeneity does not affect the performance of the application prediction module. However, applications sometimes suffer the flash crowd load, which is unpredictable [74], and result in applications' SLA violation. Meanwhile, the application workload prediction is to estimate the number of application requests in the next prediction period, which is not directly related to resource demand of the application, i.e., GRM has no insight on how much resources that the application really needs. Therefore, GRA in the application workload based resource management would design much more complicated resource allocation strategy than GRA in

the other two managements mentioned earlier.

B. Application workload based resource allocation

Similar to the VM based resource allocation module, the application workload based resource allocation strategy is also implemented by two components: GRA and LRA which are located in GRM and LRM, respectively.

1) *Application workload based global resource manager:* Application workload based GRM is to predict the workload demands and assign resources to applications in GRA. Normally, SLA in application workload based resource management is defined as the average application response time in terms of the waiting time plus the service time. In order to maximize the resource utilization and satisfy applications' SLAs, minimum resources assigned to each application should be decided (i.e., if the minimum resource is allocated to the application, the average response time for handling a request during a specific time period is equal to the predefined SLA). Then, GRM should map the minimum resource demands of all applications into physical servers. So, GRA comprises three mapping functions as shown in Fig. 6.

a) *The mapping function from the average arrival rate to the average service rate:* The first mapping function is to decide the minimum average service rate (the average number of requests to be served during one time slot) needed based on the estimated average arrival rate of the application (i.e., the number of application requests during one time slot) and the predefined SLA. Studies [72], [75]–[78] have applied queuing models to formulate the relationship between the estimated average arrival rate and average service rate demand of given SLA for each application/application-tier.

Kimish *et al.* [79] argued that an application running in a particular server follows the M/M/1 queueing model, i.e., the application arrival and service rates are all Poisson distributions. Liu *et al.* [84] also assumed the inter-arrival time of the request sequence of the whole application follows an exponential distribution. Meanwhile, they considered the service time sequence of the application requests to be exponentially distributed no matter how many VMs or servers are serving the application. Studies [75], [76], [78] considered that one application can be separated into different tiers and formulated the queuing model based on each application-tier rather than the whole application. Wang *et al.* [75] and Italo *et al.* [76] assumed the application configuration shown in Fig. 7, i.e., all the applications can be separated into M tiers, each of which is served by one VM. Meanwhile, all the application requests from the clients should be served by all the application-tiers, i.e., the departure rate of application-tier $i - 1$ is the arrival rate of application-tier i . Based on the above assumption, each VM that serves the application-tier is assumed to follow the M/M/1 queueing model with FCFS scheduling, and so the entire application is modeled as an interconnected network of M M/M/1 queues, one for each tier. In other words, the average application response time is the sum of time delays in every M/M/1 queue model from all tiers. Massimiliano *et al.* [77] argued that not all the applications' workflows pass through tier by tier, i.e., different application

requests may not need to process through all the tiers (all the application's functions) in the application configuration. Peter *et al.* [78] modeled the application as a Directed Acyclic Graph (DAG) where nodes represent application-tiers (i.e., application functions) and edges represent the relationships between tiers (i.e., if the output of one application-tier is the input of another application-tier, then there is a directed line between the two tiers). Associated with each edge is a rate indicating the number of requests of the destination node triggered per request of the source node. According to the application configuration, each tier's requests served by a VM are modeled as a M/G/1/PS queue, i.e., the average service rate of each tier's requests running on the VM is assumed to have an arbitrary distribution, the service discipline is assumed to be processor sharing, and the average request arrival rates on each tier are assumed to have a Poisson distribution. Wood *et al.* [19] modeled one application served by one VM as a G/G/1 queueing system, and so the VM's average service rate is not only determined by the application's average arrival rate but also decided by the variance of application request inter-arrival times and server time.

b) *The mapping function from the average service rate demand to the minimum resource demand:* The second mapping function is to find the relationship between the average application service rate demand and the minimum resource demand. So, by combining with the first mapping function, the relationship between the average arrival rate of the application/application-tier and the minimum resource demand is established by the given application's SLA.

It is normally assumed that only one VM serves one application/application-tier, and most studies tried to determine the minimum size of the VM. The average service time of the VM which is serving a particular application-tier is assumed to be linearly proportional to the amount of resources allocated to the VM [76]. Kimish *et al.* [79] considered the relationship between the average service rate and the minimum resource demand as a black box, and the application/application-tier's performance is tested under different resource provisioning before it is deployed in the data center, i.e., the data center provides limited types of VMs/servers to host the applications (different types of VMs/servers have different resource configurations which are similar to the types of instances provided by Amazon EC2 [80] and Windows Azure [81]), and the application/application-tier is first run in every type of VM/PM and tested with every type of VM/PM's average service rate. Therefore, the average application/application-tier's service rate of different types of VM/PM is obtained and GRA can select different types of VM/PM according to the average application/application-tier's arrival rate.

Instead of mapping the application service rate to the minimum resource, some studies directly mapped the arrival rate to the minimum resource under the constraint of the SLA in terms of the response time. In doing so, the first mapping function becomes unnecessary. Van *et al.* [82] claimed that the response time is linearly proportional to the application arrival rate when the CPU utilization is 100% and different types of CPU exhibit different linear curves to indicate the relationship between the response time and arrival rate by given the SLA.

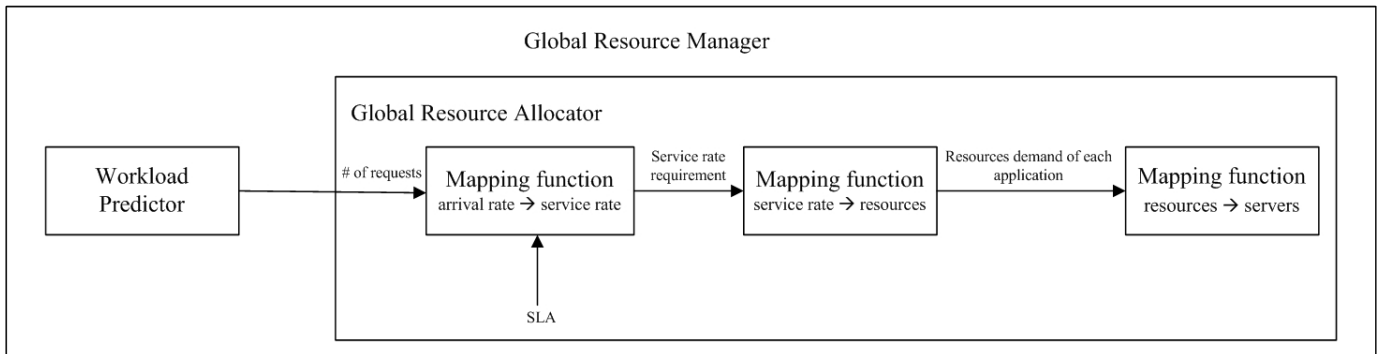


Fig. 6. Global resource manager configuration.

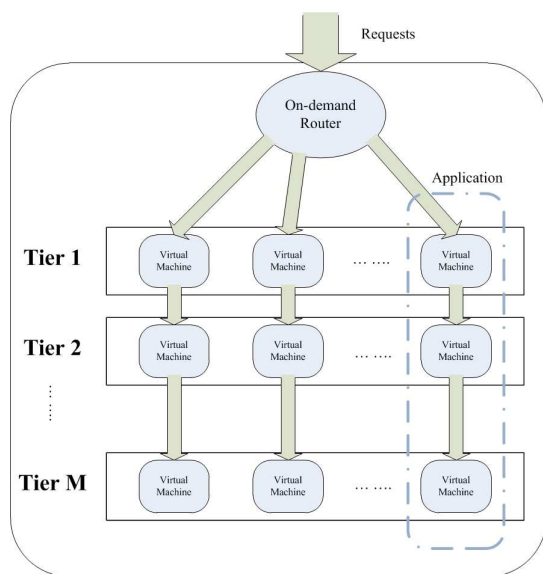


Fig. 7. Application configuration in data center [75].

So, based on different arrival rates, we can select different types of CPUs to meet the application's SLA in terms of response time. Ying *et al.* [83] showed that the application arrival rate is proportional to the application resource demand and provided an empirical formula to setup the relationship among the arrival rate, resource demand and application performance. Wang *et al.* [69] proposed that the average arrival rate of each application-tier is linearly proportional to the CPU consumption (i.e., the number of CPU cycles consumed by running a specific application-tier) over the average response time, and the coefficient of the linear function is calculated based on the historical data traces, which include arrival rates and response time of different application-tiers, CPU consumption and CPU capacity over different time slots. Via experiments, Liu *et al.* [84] demonstrated that when only one VM serves the application/application-tier, the average response time is a convex function of the CPU utilization for a given application/application-tier average arrival rate, but if the application/application-tier is deployed in more than one VMs, the mapping function is very complicated, and so pre-testing the application performance under different resource provisioning is the only way to solve the mapping function.

All in all, the best solution to establish the mapping function between the arrival rate and the minimum resource demand is to test the application or application-tier over different particular hardware configurations.

c) *The mapping function from the minimum resource demand to servers:* By building the first two mapping functions, the relationship among the minimum resource demand, SLA and the arrival rate of application/application-tier is established, i.e., the expression of $resources\ demand = f(SLA, arrival\ rate)$ for each application/application-tier is known, but assigning the minimum resources to the applications/application-tiers is not equal to maximizing the resource utilization. Finding the minimum number of awaked servers that can provide all the applications/application-tiers' minimum resource demand is the optimal solution to reach the maximum resource utilization in the data center. Therefore, mapping the minimum resource demand to the physical servers efficiently is an important step in resource management.

Recently, a third mapping function, the VM placement problem, has been formulated as a bin packing problem [84]–[86], which can be depicted as follows:

- **Given:** 1) N number of VMs and the resource demands of each VM d_i^r (where $1 \leq i \leq N$ and r denotes different types of resource demands, i.e., $r \in \{cpu, mem, bandwidth, disk\}$); 2) the resource capacity of the servers in the data center C^r (note that the servers in the data center are considered to have the same configurations and we will discuss the VM placement problem among heterogeneous servers in Section V-B1).
- **Obtain:** The VM location indicator variable, $x_{i,j}$ (i.e., $x_{i,j} = 1$ indicates the VM is placed in the j th server; else, $x_{i,j} = 0$).
- **Objective:** Minimize the total number of awaked servers.
- **Constraints:** 1) each VM is placed in one server, i.e., $\forall i, \sum_j x_{i,j} = 1$; 2) the capacity of each server is not violated, i.e., $\forall j, \forall r, \sum_i d_i^r x_{i,j} \leq C^r$.

The bin packing problem is a widely accepted NP-hard problem [87] and many traditional heuristic algorithms (such as First Fit, Best Fit, Best Fit Decreasing, etc.) have been proposed to solve it. However, the efficiency of the bin packing based VM placement strategy excessively depends on the predication of the VM resources, i.e., if the VM resource

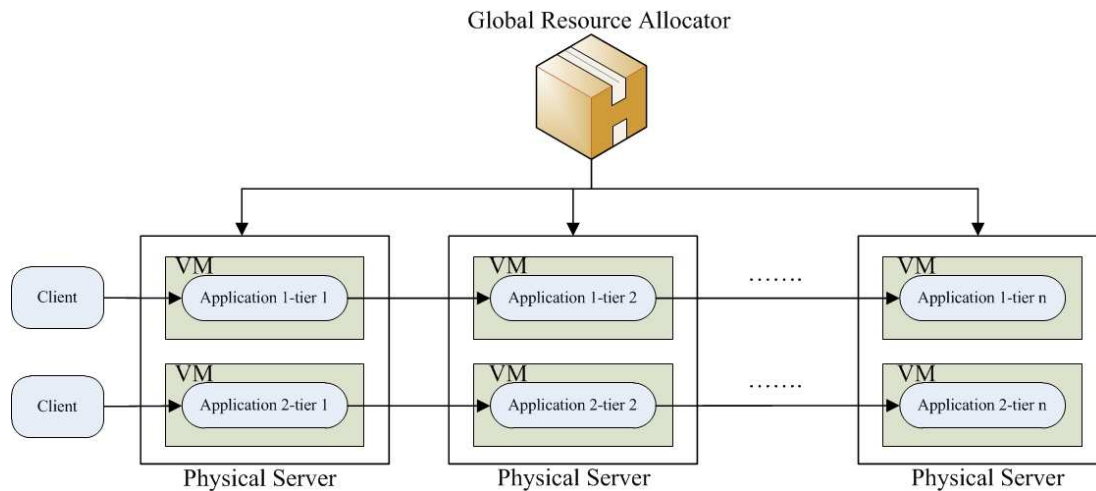


Fig. 8. An example of n application-tier resource allocation.

demands are under-estimated/over-estimated, the VM placement strategy may result in SLA violation/resource utilization degradation.

Jin *et al.* [88] proposed that VM resource demands are divided into deterministic (such as CPU resource) and stochastic resource demands (such as bandwidth resource). The deterministic resource demands are static during each time slot, while the stochastic resource demands varies and the distribution of each stochastic resource demand r follows a normal distribution $N(\mu_i^r, \delta_i^{r^2})$, where μ_i^r is the expectation of stochastic resource r 's demand for VM i , and $\delta_i^{r^2}$ is the variance of stochastic resource r 's demand for VM i . Thus, the authors formulated the stochastic VM placement problem in which the objective and the variables are the same as the traditional bin packing VM placement. Yet, in addition to guaranteeing that the hosting VMs' total deterministic resource demands do not exceed the server's capacity, the formulation includes one more constraint that for each stochastic resource demand, the resource underprovisioning probability (that the hosting VMs' total resource demand exceeds the server's capacity) is less than α . Other studies [89], [90] formulated the VM placement problem as a stochastic integer programming by considering the VM resource demands and the VM cost (e.g., the electricity price) as stochastic values. However, their objective is to minimize the total cost of serving the VMs rather than minimizing the total number of awaked servers.

Considering the VM placement as a bin packing problem or stochastic programming leads to rearranging all the VMs' placements for each time slot. Thus, in order to improve the complexity as well as reduce the number of migrations, many studies design generic algorithms to find the sub-optimal solution of the VM placement [91]–[93]. The basic idea of these algorithms is to only rearrange the hotspot servers' (i.e., servers cannot satisfy resource demands of their hosting VMs) VMs to the suitable places. Specifically, the algorithms try to migrate the VMs from hotspot servers to the lightly loaded servers. The lightly loaded servers are ordered by a certain criteria (e.g., CPU usage, memory usage or some resource load indicator function). The VM migration attempts are made,

starting with the first lightly loaded server (which has the most amount of idle resource) and continuing until the set of hotspot servers is exhausted or there is no lightly loaded server left. Liu *et al.* [84] proposed that instead of supplying one large VM (which can accommodate the minimum resource demand) serving one application/application-tier, provisioning more smaller size VMs, which can be collaboratively working for the same application/application-tier, is more intriguing since higher resource utilization in terms of a smaller number of awaked servers, can be reached, thereby a novel VM-splitting and assignment heuristic algorithm was proposed. Other studies [69], [76], [94] argued that optimizing VM placement globally would drain the network resource and degrade the VMs' performance. This is because implementing VM placement would trigger live migration of a huge amount of VMs that incurs huge bandwidth and causes service interruption. So, they tried to optimize the resource allocation locally, i.e., they assigned resources to different VMs (which may serve different applications) within one server. For instance, in Fig. 8, there are two applications and each can be separated into n -tier and distributed into n physical servers. GRA tries to allocate resources to VMs on each server in order to satisfy the whole applications' SLA. Although it is not an optimal resource allocation solution and somehow cannot guarantee applications' SLA, this method relaxes the complexity of the VM placement problem and avoids live VM migration.

2) *Application workload based local resource manager:* Similar to the other two resource management schemes, applying the application workload based LRM in each server can potentially improve the resource utilization and enhance applications' QoS by adjusting the resource allocation among VMs in a small time scale. The VM workload based local resource scheduling strategies mentioned earlier can also be implemented in the application workload based LRM. On the other hand, different from VM based LRM, application based LRM can acquire the application level information (e.g., average arrival rate of an application/application-tier, applications' SLA, etc.) from its GRM to deploy its unique

local resource scheduling strategy in a server.

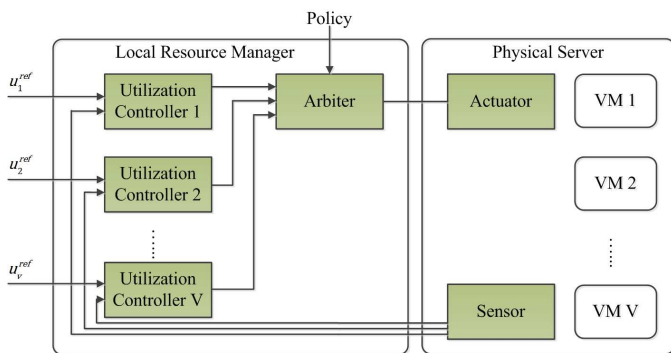


Fig. 9. Local resource manager architecture [69].

Wang *et al.* [69] constructed LRM shown in Fig. 9. Each VM is running one application-tier and the sensor in the server senses each VMs CPU utilization in each time slot. The utilization controller compares the recent CPU utilization with the CPU utilization demand μ_v^{ref} (i.e., $\frac{\text{arrival rate}}{\text{service rate}}$), which is given by GRM. If the recent CPU utilization is larger than the CPU utilization demand, the Arbiter would increase the CPU entitlement of the specific VM, and vice versa. Also, the Arbiter should consider some policies (e.g., priorities of different application-tiers) enforced in the VMs. The Actuator (which is considered as a LRM) is to implement the CPU entitlement quota of each VM in the real physical CPU. Instead of measuring each VM's CPU utilization, Wang *et al.* [62] proposed to monitor the performance of each application in a server (each VM is used to host one application) periodically. The resource controller in the server is to compare each application's performance with its predefined SLA and make the resource allocation decision, i.e., if the current application performance is worse than its SLA, then it would have more chance to obtain more CPU entitlement, and vice versa. Ying *et al.* [83] traced each VM's resource demand at each time slot and adjusted the local resource provisioning to minimize the sum of the application utility function, which is determined by the resource demand, resource provision, SLA of the application, and the application priority.

V. COMPARISONS OF DIFFERENT RESOURCE MANAGEMENT MECHANISMS IN THE HETEROGENEOUS DATA CENTER

In a real data center, it is difficult to achieve optimal resource management because of the complicated data center environment, and most of the studies discuss the resource management mechanism in a homogeneous data center, which does not exhibit all the realistic heterogeneity features that will be delineated in the following. In this section, we will first explain the heterogeneity characteristics in the real data center and then analyze the difficulty of implementing different resource management mechanisms in the heterogeneous data center.

A. Heterogeneity characteristics of a data center

We provide a brief overview of four heterogeneity features of a data center that present impediment to resource manage-

ment:

Resource heterogeneity: Hardware provided by the data center is heterogeneous [6], [95], [96], i.e., configurations of servers in the data center are not identical. For instance, different servers may be equipped with different types of CPU, and different amount of RAM and hard disk. Resource heterogeneity is very common in a large-scale data center because of necessary and frequent replacements of out-of-order servers and installation of new state-of-the-art servers that are more powerful than the existing ones to accommodate the dramatic increment of resource demands in the data center [79], [97]–[99]. Meanwhile, the high performance servers are equipped with more energy efficient components (e.g., energy efficient CPU, memory and NIC) to serve applications so that the hybrid (in the sense of having energy and non-energy efficient servers) data center can decrease the OPEX of data center vendors.

Application type heterogeneity: Applications running in the data center may rely on different types of resource to determine their performance [6]. For example, CPU-intensive applications (e.g., GZIP data compression [100], scientific computing [99], etc.) consume more CPU resource than other types of resource to enhance their performance, and other memory (Multigrid application [101], multimedia applications [102], etc.) or network I/O (e.g., web services [103]) intensive applications' performance depends on the amount of memory or bandwidth assigned to them. Therefore, application type heterogeneity affects the performance of different applications differently, and presents great challenges to resource allocation among different applications in the data center.

SLA heterogeneity: SLA is a QoS contract signed between the service provider (i.e., the SaaS provider who rents resources in the data center) and the infrastructure provider (i.e., the data center provider). In other words, the service provider requires its service to be guaranteed with a certain kind of QoS. The infrastructure provider, on the other hand, tries to manage its resources and meets the QoS as much as it can; otherwise, it would pay a penalty. Basically, SLA defines the QoS metric, i.e., the QoS related cost function [104], [105]. However, different service providers have their own definitions of QoS [13], [106]. Some studies define QoS as the average response time that one application request is fulfilled by the VM(s) in the data center [77], [79], [84], [107], [108] or the probability of the average response time needs to be achieved [76], while some define QoS as the average packet loss percentage as their performance metric [13], [109], and some depict QoS as the throughput (i.e., requests per second [110] or the number of completed jobs per time slot [111], [112]). SLA heterogeneity results in selecting different models to estimate the application performance, thus leading to complicated resource management.

Workload heterogeneity: Workload in the data center exhibits spatial and temporal dynamics. Spatial dynamics refers to different features exhibited by different workload data traces of VMs/servers/applications, and these features can be characterized by three parts: usage mode, intensity and duration. Dynamic usage mode among data traces indicates that workloads of different VMs/servers/applications' exhibit different

TABLE I
THE COMPARISON OF THREE RESOURCE MANAGEMENT MECHANISMS IN THE HETEROGENEOUS DATA CENTER

	Resource heterogeneity	Application type heterogeneity	SLA heterogeneity	Workload heterogeneity
VM workload prediction	H	L	N	H
VM workload based resource allocation	N	L	N	N
PM workload prediction	H	L	N	H
PM workload based resource allocation	N	L	N	N
Application workload prediction	N	N	N	L
Application workload based resource allocation	L	L	L	N

characteristics, e.g., some express a seasonal, weekly or diurnal cycle over time [72], [73], [113] and some workload data traces show sudden huge spikes, i.e., sudden workload surges [74], [114], which are considered as unpredictable workload and degrade the application performance. Dynamic intensity implies that the ratio of the peak workload to the average workload is different among different data traces. Dynamic duration implies that the length of the workload data trace varies, i.e., the makespan of one application/application-tier running on a specific hardware is dynamically changed, e.g., the durations of some short-term jobs running in particular VMs are less than 15 minutes, but some jobs run longer than 300 minutes [115].

On the other hand, temporal dynamics means that the workload data trace of a VM/PM/application may change its features over time. Temporal dynamics often exists in the VM/PM workload data trace because of changes of the ownership of VMs, VM migration and server consolidation. Therefore, owing to the heterogeneity of workload features, there is no unique model that can predict every object's future workload accurately, thus rendering resource allocation inefficient.

B. Comparison of three resource management mechanisms in heterogeneous data center

In this section, we will discuss how the different kinds of heterogeneities in the data center degrade the performance of the three resource management mechanisms. We measure the degree of performance degradation into three levels: H (High, i.e., the heterogeneity feature degrades the performance of the corresponding resource management mechanism significantly and there is no suitable solution to solve the problem), L (Low, i.e., the heterogeneity feature may complicate the corresponding resource management mechanism, but can be solved by applying suitable methods), N (None, i.e., the heterogeneity feature does not affect the performance of the corresponding resource management mechanism). Table I summarizes the comparison results.

1) *The impact of the resource heterogeneity:* Resource heterogeneity makes each server's resource capacity heterogeneous, and thus degrades the prediction accuracy of VM/PM workload prediction (mentioned in Section II). Furthermore, resource heterogeneity degrades the performance of the VM/PM workload based resource allocation strategy because the strategy tries to find two or more VMs/servers

such that the sum of their resource utilization is less than a threshold when performing VM/PM consolidation, and if the servers' resource capacities are different, the summation of resource utilization makes no sense (i.e., summation of resource utilization would probably reach 100% because of the different capacities of servers). For instance, suppose there are two PMs, i.e., PM 1 and PM 2, and the CPU capacity of PM 1 is two times higher than that of PM 2 (for simplicity, we only consider CPU resource in this example); meanwhile, as shown in Fig. 10, initially (i.e., in time slot t_0), each PM hosts only one VM, i.e., VM A and VM B, respectively, and the CPU utilization of VM A and VM B is 80% and 60%, respectively. In time slot t_1 , the CPU utilization of VM A drops to 20%, and so VM A can be consolidated into PM 2 if we only consider the sum of the VMs' CPU utilization being less than a threshold (assumed to be 90%). However, the CPU utilization of VM A will be higher than 20% after VM A has been migrated into PM 2, whose capacity is half that of PM 1, and so it is possible that the CPU utilization of PM 2 will exceed the threshold, and will result in the SLA violation.

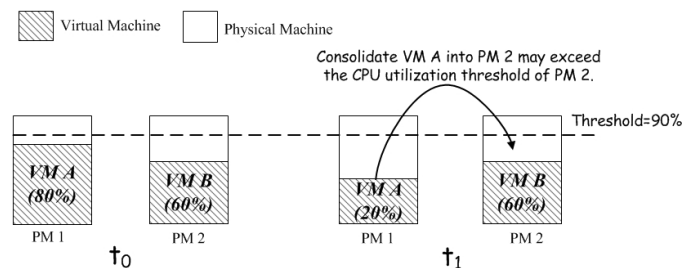


Fig. 10. Effects of the resource heterogeneity on the performance of the VM workload based resource allocation strategy.

One solution to relax the problem resulting from resource heterogeneity is to predict the amount of resources that each VM/PM needs rather than the resource utilization, but it is a big challenge to precisely predict the exact resource demand of each VM/PM, especially to estimate the CPU resource demand. Zhang *et al.* [98] separated the heterogeneous servers into N sub-clusters and each sub-cluster comprises a number of homogeneous servers. As shown in Fig. 11, the proposed resource management model has two levels of management. In the global resource management, instead of predicting the resource utilization of VMs and servers, the global scheduler first predicts the resource usage of each sub-cluster and subsequently calculates their residual resources. Based

on the estimated residual resources of each sub-cluster, the global scheduler distributes the incoming application workload among the sub-clusters. The amounts of assigned application workloads for each sub-cluster are proportional to their estimated residual resources. In sub-cluster resource management, the local scheduler first maps the resource utilization of the sub-cluster into the scheduling delay (i.e., the scheduling delay is expressed as a linear function of the resource utilization for each sub-cluster). Then, the local scheduler tries to minimize the number of awake servers in its sub-cluster and minimize the SLA (in terms of the upper bound of the scheduling delay) cost simultaneously.

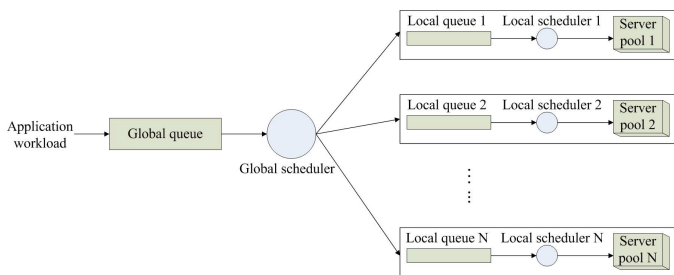


Fig. 11. A data center resource management model [98].

On the other hand, in the application workload based resource management, GRM predicts the number of incoming requests for each application, and thus resource heterogeneity does not affect the application workload prediction accuracy. However, resource heterogeneity complicates the mapping function from the minimum resource demands to physical servers (i.e., the third mapping function mentioned in Section IV-B). Placing the minimum resource demands in terms of VMs into the minimum number of physical servers with heterogeneous features can be formulated as a vector bin packing problem, which is proven to be NP hard [117]. The heterogeneous features of physical servers are depicted as the different capacities and efficiency (in terms of the amount of power consumption by running a unit of workload). It is obvious that GRM prefers to pick the physical servers with higher capacity to host the suitable VMs in order to minimize the number of the awake servers. However, it is not the best solution if the objective is to maximize the resource utilization of the awake servers (note that minimizing the number of the awake servers is not equivalent to maximizing the resource utilization of the data center in the heterogeneous data center) or maximize the energy efficiency of the data center. For instance, suppose there are three clusters of PMs (PMs from the same cluster have the same configurations), denoted as PM A, PM B and PM C, and the capacity of each PM in cluster PM A, PM B and PM C is 1 unit, 0.5 unit and 0.8 unit, respectively; meanwhile, the energy efficiencies of the PMs from different clusters are related as follows: PM C > PM B > PM A. Assume four VMs are to be allocated in the PMs, i.e., VM 1, VM 2, VM 3 and VM 4, and the resource demands of VM 1, VM 2, VM 3, and VM 4 are 0.5, 0.5, 0.5, and 0.4 unit, respectively. Fig. 12 illustrates different strategies for awaking different PMs to host the VMs (note that we consider the resource utilization threshold of each PM to be 100% in

this example) in order to achieve different objectives, i.e., minimizing the number of the awake PMs, maximizing the resource utilizations of the awake PMs and maximizing the energy efficiencies of the awake PMs. However, most of the recent studies [86], [118], [119] preferred to first pick the PMs with higher power efficiency in hosting the suitable VMs in order to improve the energy efficiency.

2) *The impact of the application type heterogeneity:* Application type heterogeneity results in different applications/VMs/PMs requiring different demands of different resource types, and thus leads to the unbalanced demands among different types of resource in a server, i.e., some servers may have higher CPU demands but lower memory and disk demands, and some may have lower CPU demands but higher memory and disk demands. Thus, it is necessary to consider the features of multi-dimensional resource demands to improve the resource utilization, reduce the energy consumption of the data center as well as guarantee the applications' SLA [120], [121]. Obviously, application type heterogeneity increases the complexity of VM/PM workload prediction because GRM needs to estimate multi-dimensional instead of one-dimensional resource utilization of VMs/servers and set up a different prediction model to estimate each dimensional resource utilization of each VM/PM. Yet, application type heterogeneity does not affect the application workload prediction, which is to estimate the average number of incoming application requests.

Moreover, application type heterogeneity complicates the applications/VMs/PMs based resource allocation, i.e., it is more complicated to map the resource demands into the minimum number of physical servers by considering the multi-dimensional nature of the resource demands (this is proven to be NP hard [122], [123]). The heuristic solutions of VM, server and application workload based multi-dimensional resource allocation are similar (if only application type heterogeneity is considered), which can be divided into three categories:

Single dimensional mapping heuristics: the basic idea of this type of solutions is to map the multi-dimensional resource demands into single dimension, and thus perform VM/PM consolidation. Wood *et al.* [19] defined the *volume* of each VM/PM as the product of the VM/PM's CPU, network and memory utilization, i.e.,:

$$v_i = \frac{1}{1 - u_i^{cpu}} \times \frac{1}{1 - u_i^{net}} \times \frac{1}{1 - u_i^{men}}, \quad (1)$$

where v_i is the *volume* of VM/PM i , and u_i^{cpu} , u_i^{net} and u_i^{men} are the CPU, network and memory utilization of VM/PM i , respectively. Thus, the larger value of v_i implies the higher multidimensional resource utilization of VM/PM i . Thereby, the suitable VMs in the server with the highest *volume* are migrated to the servers with lower *volume* value. Arzuaga and Kaeli [124] defined the Virtual Server Load (VSL) as the resource demand of each physical server. Denote j as the index of the VMs which are running in server i , r as the index of different types of resource (e.g., $r \in \{cpu, memory, disk\}$), $u_{j,r}^i$ as the resource r usage of VM j in server i , and c_r^i as resource r capacity of server i , respectively. Then, the VSL

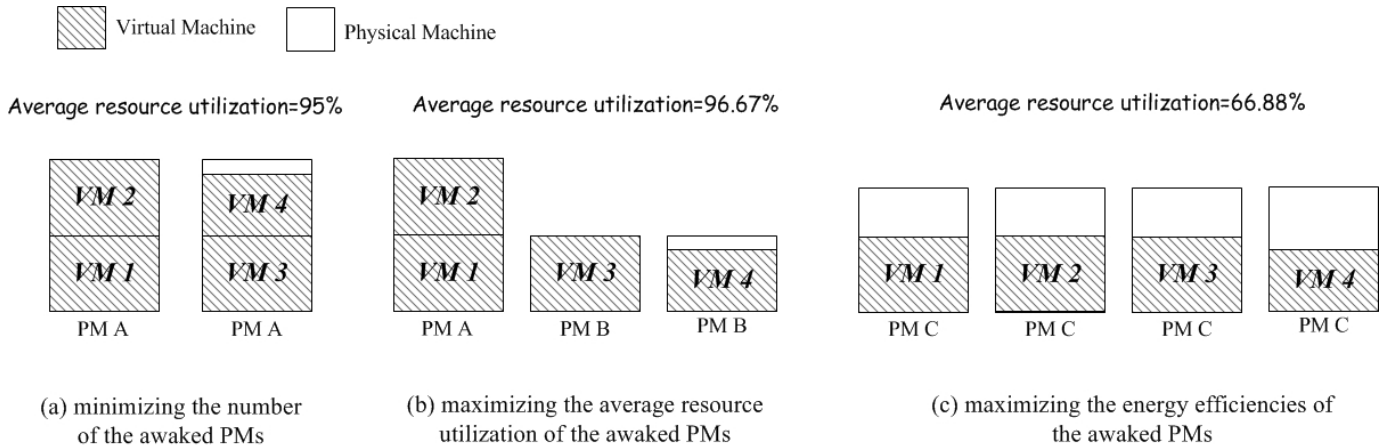


Fig. 12. The illustration for awaking different types of PMs to host the VMs in order to achieve different objectives.

of server i can be expressed as follows:

$$VSL_i = \sum_r \left(w_r \times \frac{\sum_j u_{j,r}^i}{c_r^i} \right), \quad (2)$$

where w_r is a weight associated with resource r . Based on the definition of VSL, a load-balancing VM migration framework is proposed to balance the VSL values among servers by migrating suitable VMs from the servers with larger VSL values to the servers with smaller VSL values.

Although mapping the multi-dimensional resource demands into one dimension decrease the complexity of VM migration/placement, it sometimes misleads the real resource demands of VMs/servers, and thus results in inappropriate VM migration/placement [86], [125].

Multi-dimension aware heuristics: Rather than ignoring the relationships across multi-dimensions, multi-dimension aware heuristics is to consider the server's complimentary demands for different dimensions simultaneously during VM migration/placement. Singh *et al.* [126] proposed the VectorDot scheme to balance the multi-dimensional workload among physical servers. The basic idea of VectorDot is to place the VM (which has, for instance, high CPU demands but low memory demands) to the suitable server (which has low CPU utilization but high memory utilization), i.e., the VM's resource demands is complementary to the resource utilization of the target server. In order to measure the complementarity of the VM among servers, the dot product is defined in the paper. Thus, the VM is migrated to the available server (i.e., the server with enough space to hold the VM) with higher dot product value. Norm-based Greed scheme [127] proposed the similar idea, however, the method defined resource distance metric (i.e., $\sum_r w_r u_{j,r} h_{i,r}$, where w_r is a weight of resource r , $u_{j,r}$ is VM j 's demands of resource r , and $h_{i,r}$ is server i 's residual capacity of resource r), rather than the dot product to measure the complementarity between a VM and a server.

3) *The impact of the SLA heterogeneity:* The application SLA heterogeneity does not affect the performance of the VM/PM/application workload prediction algorithm, because the variation of the VM/PM/application's resource demands is

not determined by the SLA. Also, VM/PM workload based resource allocation is not impacted by the SLA, since no matter what kind of QoS contract is signed in the SLA, once the SLA is violated, the average resource utilization of the server must have exceeded a predefined threshold in the VM/PM workload based resource allocation. In other words, ensuring every dimensional resource utilization of the server less than a threshold can satisfy the SLA.

The application SLA heterogeneity, however, complicates the application workload based resource allocation because different types of SLA may generate different mapping functions to generate the average service rate based on the application average arrival rate, and calculate different minimum resource requirement of the application consequently. For instance, if the SLA is depicted as the application's average response time, normally, different queuing models are applied to formulate the relationship between the average service rate and the average arrival rate of the application; however, if the SLA is defined as the average packet loss percentage or the average throughput of the application, it is difficult to map the average arrival rate into the average service rate of the application by applying queuing models. Therefore, when applying application workload based resource management, suitable models should be designed and investigated in order to establish the relationship between the average arrival rate and the actual resource demands based on different types of SLAs. Song *et al.* [13], [128] proposed that the performance of the application (which can be defined as the application's average response time, throughput, average packet loss percentage, etc.) is a linear function with respect to the application average arrival rate and the amount of resources assigned to the application, and the coefficients of the linear function can be measured with plentiful experiments. Thereby, by obtaining the function, it is easy to calculate the minimum resource provisioning based on the application average arrival rate and the SLA. Rather than modeling the different mapping functions in terms of $resource\ demand = f(SLA, arrival\ rate)$, Zhan *et al.* [112] separated the applications with different SLAs into different priority levels, which are based on the requirement of the response time. For example, a web server application

(who needs immediate response) has a higher priority than a MapReduce application (whose SLA is defined as the application's throughput), and thus the web server application can be assigned with more resources and the MapReduce applications can be queued if the resources are not available.

4) *The impact of the workload heterogeneity*: The workload prediction accuracy does not determine or affect the resource allocation strategy applied in GRM of the data center, but it is the main factor in determining the efficiency of GRM. Meanwhile, workload heterogeneity is the key jeopardy in degrading the performance of the prediction algorithm. As mentioned in the previous section, VM/PM/application workload in the data center exhibits spatial and temporal dynamics. In order to solve the spatial dynamics problem among different workload time series, it is necessary to build the prediction model individually by analyzing the features of corresponding VM/PM/application workload time series. For instance, if the workload time series has the clear seasonal component with low noise (i.e., fewer random fluctuations), it is better to apply Moving Average (MA) prediction model to estimate the future workload [129], [130]; on the other hand, if the workload time series has some noise and changes within trend, but no seasonal behavior, then the Simple Exponential Smoothing model may perform better [132]; moreover, if the workload time series has the clear seasonal component with some noise, then GRM can apply the Auto Regressive Integrated Moving Averages (ARIMA) stochastic process model to predict the workload [72], [135]. Vazquez *et al.* [133] evaluated the accuracy of several common workload prediction models by testing the models to forecast the real cloud computing workloads including Google cluster [7] and Intel Netbatch [134]. Also, the complexity and the prediction period of different prediction models should be considered in selecting a suitable model to estimate the workload. For instance, if the workload is a seasonal time series, then ARIMA can perform long-term workload estimation at the expenses of higher complexity, while MA can only precisely predict the short-term (i.e., next one or two time slots) workload at the gain of lower complexity. Herbst *et al.* [136] theoretically analyzed different prediction models for application scenarios as well as their pros and cons with respect to the complexity, the historical data requirements (i.e., how many historical data points are needed to predict the future data points) and the prediction period.

The temporal dynamics implies that the features of a specific workload time series may vary over time. For instance, the workload has clear seasonal components initially, but, as the time passes by, the workload time series loses the seasonal component but exhibits active periods with clear trends. The temporal dynamics of the workload may degrade the prediction accuracy significantly because the historical workload data set cannot reflect the features of workload time series in the future. The VM/PM workload exhibits temporal dynamics most of the time. This is because, as mentioned previously, the application workload mainly depends on the human activities, which exhibit periodic features most of time; however, the VM/PM workload is not only determined by the human activities but also affected by the VM/PM based resource allocation, and

thus the application/application-tier running in the VMs/PMs changes over time. For example, suppose there are two applications, namely, app-A and app-B, with different features (i.e., their workload time series have different features), and each of them has two VMs initially. If app-A's workload drops and only needs one VM, and meanwhile, if app-B's workload increases and requires more resources, then GRM will release one of the VMs from app-A and re-assign it to app-B to satisfy its resource demands. Eventually, the features of the reassigned VM's workload time series is altered over time (because the application running in the reassigned VM is changed), thus degrading the workload prediction accuracy. One way to solve the accuracy degradation of the VM/PM workload prediction is to fix the number of VMs/servers for serving a specific application, i.e., no matter whether the application's workload in the data center increases/decreases, GRM can only adjust the size of VMs/servers without changing the number of VMs/servers serving the application. Herbst *et al.* [136] proposed a new workload classification and forecasting system to solve the VM/PM workload temporal dynamics problem. The basic idea of the proposed workload classification and forecasting system, which is quite similar to [17], is to automatically select the most suitable prediction model (which yields the minimum Mean Absolute Scaled Error) from the model set in each time slot. Thus, once the features of the workload time series are changed, the system can adaptively choose the suitable prediction model after a certain time period. However, the drawback of the proposed method is high complexity, i.e., two or more prediction models need to be executed in parallel to compare the prediction accuracy.

VI. OPEN ISSUES

Several works have contributed to maximize the resource utilization by applying the resource management schemes as presented in the previous sections. However, there are still some issues which need to be addressed. This section discuss several open issues and possible research directions in the resource management of the data center.

A. Network aware resource management

With an explosive growth of data center traffic, network bandwidth constraint becomes increasingly more critical. Traditional resource management only tries to assign the PM resources to the VMs in order to guarantee the corresponding SLAs while ignoring the resource management at the network layer. As mentioned in Sec. II-B2, provisioning sufficient resources to the VMs in the local PMs may not satisfy the SLAs if the resource is under-provisioned at the network layer. In other words, allocating insufficient network resources to the VMs may degrade the performance of the VMs, thus violating the applications' SLAs. For example, traditionally, the web service application comprises three tiers, i.e., a front-end Web server tier, a middle application tier, and a backend database/storage tier. It is desirable to guarantee efficient bandwidth provisioning among the three tiers' communications to avoid application performance degradation. The optimal

network aware resource management is to minimize the number of the awaked PMs while guaranteeing each link in the network not to be congested, i.e., the network resources are not under-provisioned. However, the VM placement problem is coupled with the network resource allocation problem, i.e., a bad VM placement may incur insufficient networking resource provisioning to the VMs; this coupling thus complicates the network aware resource management.

Many studies have considered the network aware resource management problem as the Virtual Data Center (VDC) embedding problem. Specifically, each application is running on a Virtual Data Center (VDC), which consists of not only a number of VMs but also the virtual switches and virtual links in providing the virtual network connectivities among VMs, and so allocating the resources to the application can be considered as embedding the VDC into the physical data center. Basically, the framework of the VDC embedding problem is shown in Fig. 13. Different VMs, which communicate with each other to serve the same application, form a VDC and every VDC has its own IP address space². VMs within the same VDC can communicate with each other just as they are in the same layer-2 Ethernet. VMs in different VDCs can communicate through layer-3 gateways. The VDC predictor estimates its VDC resource demands, which include the number of VMs, the resource capacity of each VM, and the bandwidth demand matrix among the VMs. The VDC solver is a centralized resource manager in the data center. On the one hand, the VDC solver receives the information of the physical data center, such as the topology of the data center and the status and resource utilization of the PMs and switches; on the other hand, it generates solutions for efficiently provisioning the resources to each VDC.

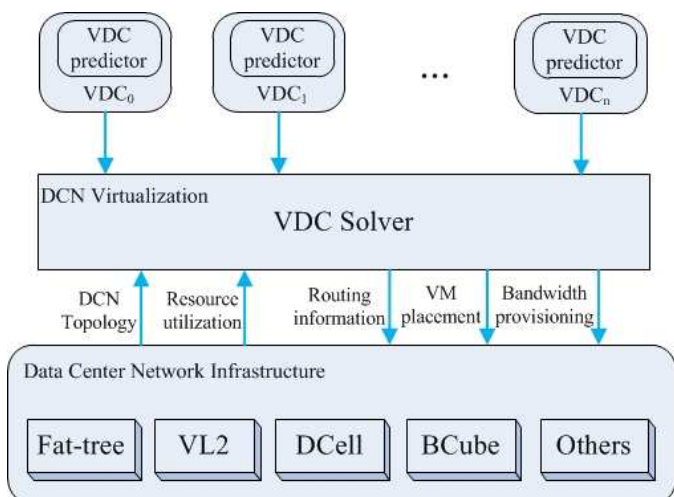


Fig. 13. The framework of the VDC embedding.

Based on the VDC embedding framework, Rabbani *et al.* [137] proposed a heuristic algorithm to efficiently solve the VDC embedding problem, which is proven to be NP-hard [137], [138]. The algorithm comprises three steps, i.e., VM

mapping, virtual switch (*vswitch*) mapping and virtual link (*vlink*) mapping:

- **VM mapping:** The basic idea of VM mapping in the heuristic algorithm is to balance the resource demands among the PMs. Specifically, each VM selects the PM with the minimum *PM cost* among the feasible PMs, which have enough residual resources³ to host the VM. The *PM cost* of PM i , denoted as $PM_{cost}(i)$, is defined as follows:

$$PM_{cost}(i) = \frac{b(i)}{|r(i) - \bar{r}|}, \quad (3)$$

where $r(i)$ is the residual CPU resource of PM i , \bar{r} is the average residual CPU resource among PMs and $b(i)$ is the occupied bandwidth resource of PM i . Thus, a smaller value of $PM_{cost}(i)$ indicates that the PM has less occupied bandwidth and more residual CPU resource.

- **vswitch mapping:** In the second step, the heuristic algorithm tries to map each *vswitch* in VDC into the physical switch with the minimum *communications cost* among the feasible physical switches, which have enough residual capacity to host the resource demands of the *vswitch*. The *communications cost* of physical switch j , denoted as $pswitch_{cost}(j)$, is defined as follows:

$$pswitch_{cost}(j) = \sum_{k=1}^N h(j, k) b(k), \quad (4)$$

where N is the number of the VMs that are connected to the *vswitch* in the VDC, $h(j, k)$ is the number of hops between VM j and physical switch k , and $b(k)$ is the bandwidth demand of VM k . The intuition behind the *vswitch* mapping is that it prefers to map the *vswitch* into the lower layer physical switch that generates a fewer number of hops and thus results in a lower cost.

- **vlink mapping:** In the third step, the heuristic algorithm tries to map each *vlink* into the corresponding physical link. The basic idea is to find the shortest physical path between the two VMs among the available physical paths such that the residual bandwidth capacity of each selected physical link is no less than the bandwidth demand of the *vlink*.

Guo *et al.* [138] proposed a similar algorithm to solve the VDC embedding problem. Specifically, in order to reduce the complexity for solving the VM embedding problem, the neighboring PMs are grouped into clusters of different sizes before any VDC allocation takes place. Thus, each VDC only needs to search for the suitable PM cluster to host itself rather than searching the entire data center. In the first step, the VDC solver would select a suitable PM cluster to host the VDC. The number of PMs of the selected PM cluster should be larger than the number of VMs in the VDC and the aggregate ingress and egress bandwidth of the PM cluster should be larger than those of the VDC. In the second step, the VDC solver tries to map the VMs in the VDC into the PMs in the selected PM cluster. The basic idea of the VM mapping is very

²Note that IP address spaces of different VDCs may overlap.

³The paper [137] only considers the CPU and bandwidth resource demands of VMs when performing VM mapping.

similar to [137], which is to allocate the VMs into feasible PMs (which have enough residual resources to host the VMs) while balancing the residual bandwidth among PMs. In the third step, the VDC solver tries to allocate physical paths for all the VM-pairs, and at the same time tries to find the shortest available path between the two VMs by applying the Breadth First Search algorithm.

The intuition behind the two mentioned VDC embedding heuristic algorithms are very similar, but the complexity of the algorithms are very high, i.e., if the VDC solver cannot find a feasible physical path (i.e., the residual bandwidth of all the physical paths is less than the bandwidth demand of the VM pair) for a VM pair in the last step, the algorithms need to return to the first step to find another feasible VM mapping. This kind of back-tracking leads to high time complexity. In order to reduce the complexity of the algorithm, studies [139], [140] proposed the affinity-aware VM allocation method to place the affine VMs (i.e., the VMs with a large amount of communications or data exchanges among them) within the same ToR (*Top of Rack*) PMs as much as possible. This is because the communications links that only traverse the ToR switch have lower blocking probability as compared to the communications links that need to traverse the aggregation or core switches. Different from the VDC embedding based resource management (which first maps the VMs into feasible PMs, and then determines the bandwidth allocation for each VM pair), bandwidth demands of the VM pairs determine the VM mapping in the affinity-aware VM allocation method. However, the proposed affinity-aware VM allocation method cannot guarantee that all the VM pairs can acquire sufficient bandwidth provisioning on the corresponding physical path. Meng *et al.* [141] proposed a traffic-aware VM placement algorithm to reduce the network cost. The basic idea of the algorithm is to partition VMs into VM-clusters according to the traffic between different VMs as well as the data center network characteristics, and then place the VM-clusters into different slot-clusters (i.e., PM clusters) to minimize the network cost. However, the algorithm does not consider the capacity of the slot-cluster, i.e., some VM-clusters may not find the suitable slot-clusters to host them because of the capacity limitations of the slot-clusters.

Kliazovich *et al.* [142] demonstrated that there is a tradeoff between minimizing the number of awaked PMs in the data center and avoiding the network congestion (which tries to minimize the maximum link utilization). Specifically, in order to avoid network congestion, the bandwidth-aware VMs should be distributed among the PMs as much as possible. However, this methodology contradicts the energy efficient resource management strategy, which tries to concentrate all the VMs within the minimum number of PMs. Many studies proposed different methodologies to optimize the tradeoff. Jiang *et al.* [143] tried to jointly optimize the VM placement and the routing path selection among VM pairs in order to optimize the tradeoff. Specifically, they formulated the objective function f as follows:

$$f = \frac{1}{L} \sum_{l=1}^L h_l + \alpha \frac{1}{M} \sum_{m=1}^M g_m, \quad (5)$$

where L is the total number of the physical links in the data center, h_l is the cost of link l (which is a convex function of the link utilization), M is the total number of the PMs available in the data center, g_m represents the status of PM m (i.e., if PM m is awaked, $g_m = 1$; otherwise, $g_m = 0$), and α is a weighting factor in order to allow operators to freely adjust the tradeoff between the *link cost* (i.e., $\frac{1}{L} \sum_{l=1}^L h_l$) and the *capacity cost* (i.e., $\frac{1}{M} \sum_{m=1}^M g_m$). Thus, the joint optimization problem, denoted as P0, can be formulated as follows:

- **Given:** 1) the resource demands for each VM, 2) the available resources for each PM, and 3) the bandwidth demand for each VM pair.
- **Obtain:** 1) the VM location indicator variable $x_{i,m}$ (i.e., $x_{i,m} = 1$ indicates VM i is in PM m ; else $x_{i,m} = 0$); 2) the routing indicator variable $y_{i,j}^p$ (i.e., $y_{i,j}^p = 1$ indicates the traffic from VM i to VM j is routed on path p ; else, $y_{i,j}^p = 0$).
- **Objective:** Minimize f .
- **Constraints:** 1) each VM is placed in one PM; 2) the resource capacity of each PM is not violated; 3) the traffic of each PM pair is routed on one physical path.

In order to efficiently solve the optimization problem, the authors leveraged the idea of Markov chain approximation method [144] to obtain an approximated solution of P0.

Belabel *et al.* [145] argued that applying the virtual bridging⁴ and multipath forwarding (mentioned in Sec. II-B2c) techniques in the routing optimization can achieve better tradeoff between the *link cost* and the *capacity cost*. Note that recent Ethernet switching solutions in the data center network, such as Provider Backbone Bridges with Traffic Engineering (*PBB-TE*) [146], the Shortest Path Bridging (*SPB*) protocol [147] and the Transparent Interconnection of a Lot of Links (*TRILL*) protocol [148], can enable multipath forwarding of the Ethernet frames, which can potentially provide better load balancing among different paths. In order to enable the multipath forwarding in solving the joint optimization problem (i.e., P0), the authors declared the variable $y_{i,j}^p$ in P0 as a non-negative real variable, rather than a binary variable.

Meanwhile, in order to enable virtual bridging in solving the joint optimizing problem, the authors proposed that each PM can be considered as a switch to route the traffic of VM pairs. Note that enabling the virtual bridging function in the hypervisor may consume extra resources of the PM for routing the traffic, but this may reduce the available resources for each PM to host the VMs. Thus, the second constraint in P0 needs to be modified. Specifically, suppose each VM is characterized by a $|\mathcal{D}|$ -dimensional (note that \mathcal{D} is a set of resource dimensions and each dimension corresponds to a different resource type such as CPU, memory or disk space) resource demand vector $\mathbf{r}_i = [r_{i,1}, \dots, r_{i,|\mathcal{D}|}]$, where $r_{i,d}$ is the VM i 's resource demand for dimensional d ($d \in \mathcal{D}$), then,

⁴Virtual bridging is to offload the traffic (generated from the VM pair) switching operations from the access and aggregate switches to the software hypervisor level in the PM, if the two VMs are in the same PM.

for each PM m , we have the following constraint:

$$\forall d \in \mathcal{D}, \tau^d \sum_i r_i^{traffic} x_{i,m} + \sum_i r_{i,d} x_{i,m} \leq C_m^d, \quad (6)$$

where $r_i^{traffic}$ is the amount of traffic demand for VM i , τ^d is a coefficient that maps the traffic demand of VM i into the resource demand (for routing the traffic demand using virtual bridge in the PM) for dimensional d , and C_m^d is the resource capacity of PM m for dimensional d . The new joint optimization problem has been proven to be NP-hard [145]. In order to find the suboptimal solution, the authors mapped the joint optimization problem into the single source facility location problem (*SSFLP*), which is a well-studied problem, and applied the repeated matching heuristic algorithm [149], which can reach good optimality gaps for solving *SSFLP*, to generate the suboptimal values of $x_{i,m}$ and $y_{i,j}^p$.

The above works tried to propose heuristic algorithms to solve the network aware resource management problem and each of them has its tradeoff between the complexity and the performance. However, a number of VM migrations among PMs are introduced for implementing the network aware resource management algorithms, and the VM migrations consume a huge amount of the bandwidth resource in the switches [150], which may significantly increase the traffic load of the data center network, thus resulting SLA violation. Thus, it is necessary to design an optimal network aware resource management by considering the bandwidth demands introduced in the VM migrations.

B. Resource management in the green data center

In order to reduce the OPEX and CO_2 footprints for running the data center, the concept of green data center is introduced, i.e., the data center is both powered by renewable energy as well as brown energy [151]–[155]. Renewable energy is generated at the site of the data center, and once the renewable energy is insufficient or unavailable to satisfy the energy demands of the data center, brown energy, which acts as backup energy supplement, is obtained from the electrical grid (note that the renewable energy could be drawn from the nearby renewable power plants to power the data center, but this kind of renewable energy is not free for the data center providers. Thus, from minimizing the OPEX point of view, we consider green energy as the on-site renewable energy and brown energy as the energy from the electrical grid or the renewable power plants). The goal of the resource management in the green data center is to optimally allocate resources to applications in order to minimize the brown energy usage, while guaranteeing the applications' SLAs.

Normally, the power supply system of a green data center is shown in Fig. 14, in which the on-site green energy collector locally extracts energy from the green energy source and converts it into electrical power, the charge controller regulates the electrical power from the green energy collector, and the inverter converts the electrical power between AC and DC. The smart meter records the electric energy from the power grid and renewable power plants consumed by the data center. Many studies [152], [153] argued that the introduced

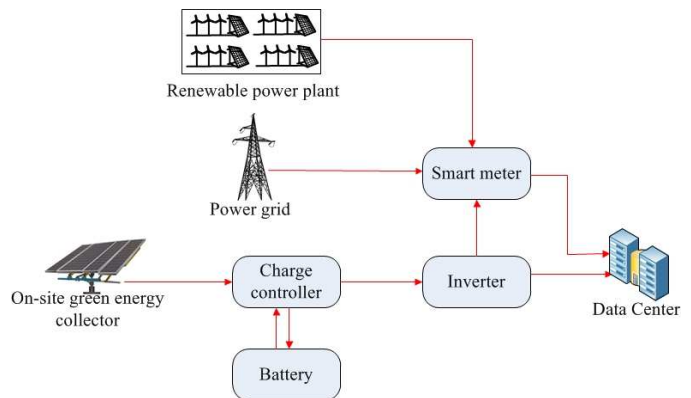


Fig. 14. The power supply system of a green data center.

batteries in the green data center may induce many problems: 1) batteries incur extra energy losses because of self-discharge; 2) equipping batteries increases the CAPEX of the green-powered system; 3) batteries contain chemicals, which are harmful to the environment. Therefore, it is beneficial to build a battery-absent green data center. Since green energy cannot be "banked", in order to minimize the brown energy usage, the generated green energy should be fully utilized in each time slot. Therefore, it is necessary to adjust the resource provisioning based on the amount of the available green energy of the data center. GreenSlot [156], [157] and GreenHadoop [153] systems have been proposed to schedule the workload based on the green energy supply. The jobs in the data center are grouped into two types: deferrable jobs (which have loose deadline, such as batch processing jobs) and non-deferrable jobs (which need to be handled immediately, such as web services). The basic idea of the two proposed systems is to buffer the deferrable jobs if the green energy is insufficient and execute them before their deadlines.

Although there are many disadvantages to equip the system with battery, in reality, data centers have already been equipped with uninterruptible power supplies (UPSs) to protect against possible power failures [158]. Moreover, though batteries consume extra energy by self-discharging, they can bank the excessive green energy instead of wasting it. Thus, it is realistic to consider the green data center as a battery enabled system. Since green energy can be banked into batteries (i.e., the green energy provisioning can be adjusted over time), fully utilizing the generated green energy may not be the optimal solution in the battery enabled green data center. For instance, green energy can be banked (i.e., not fully utilized) at the current time slot and utilized at latter time slots when the green energy generation is not sufficient to satisfy the energy demands of the data center and the electricity price is relatively high. Therefore, designing an efficient green energy provisioning strategy is critical to minimize the brown energy cost of the resource management in the data center. However, it is difficult to design an optimal green energy provisioning strategy, because the current green energy provisioning decision is coupled with the future decisions (e.g., the current green energy provisioning decision may leave insufficient battery capacity, and so the green energy generated in the near future

may be wasted). Inspired by the green energy supplement strategy in cellular networks [159], the green energy provisioning strategy and the resource management in the data center can be designed in a similar manner. Specifically, GRM predicts the green energy generation, the resource demands of the data center as well as the electricity price in each fine-grained time slot (e.g., one hour) during one coarse-grained time slot (e.g., one day) based on different kinds of prediction models. Then, according to these prediction results, an offline green energy provisioning strategy is designed to minimize the electricity cost by determining the amount of green energy available for the data center as well as the amount of the energy stored in the battery in each fine-grained time slot during one coarse-grained time slot. However, the actual green energy generation, resource demands and electricity may differ from the prediction results; an online resource management should be designed to dynamically adjust the resource provisioning to serve the application workloads in order to maintain the amount of energy stored in the battery to be no less than the value generated from the offline green energy provisioning strategy, while guaranteeing the application SLA. For instance, if the green energy generation is over-estimated/under-estimated, less/more resource should be provisioned to serve the workload in order to maintain the amount of energy level in the battery (which are pre-determined by the offline green energy provisioning strategy).

C. Cooling aware resource management

Generally, the architecture of a data center is depicted in Fig. 15, where the IT equipment includes the computing (PMs) and communication (switches) resources in the data center, the power infrastructure generates and delivers the power to the IT equipment and the cooling supply system, and the cooling supply system produces and delivers the cooling resources to remove the heat from the IT equipment. Normally, the cooling supply system comprises two major components: cooling resource generator and cooling resource distributor. Cooling resource generator, i.e., the outside air economizer and water economizer in Fig. 15, is to chill down the returned hot air/water from the Computer Room Air Conditioning (CRAC) units and send back the cooling air/water to the CRAC units. Cooling resource distributor, i.e., CRACs in Fig. 15, is to disperse the cooled air/water to the IT equipment and collect the returned hot air/water.

The cooling supply system is very important to ensure proper operation of the IT equipment. However, the energy consumption of the cooling supply system is tremendous; it was reported that about 35%–55% of the total energy consumption of the data center is consumed by the cooling supply system [160]. Moreover, the report of U.S. Environmental Protection Agency also stated that the cooling supply system consumes about 50% of the total energy consumption in the data center [161]. Therefore, minimizing the energy consumption of the cooling supply system can significantly reduce the data center's total energy consumption.

The cooling supply system is to ensure the IT equipment under redline temperature [163], and the temperature of the IT

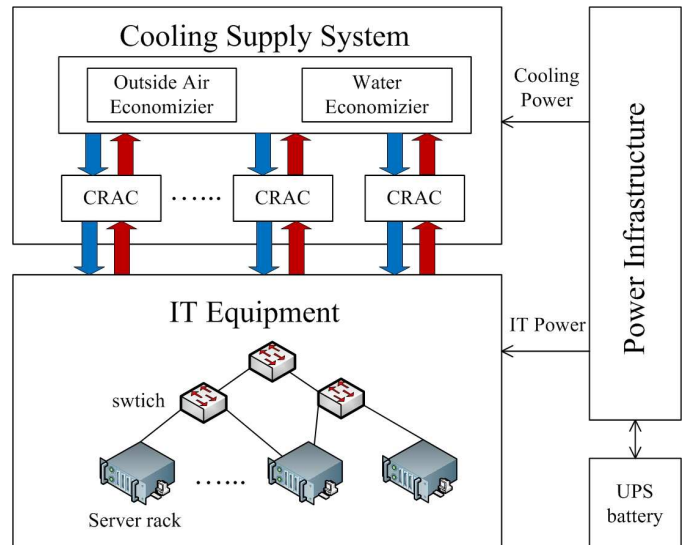


Fig. 15. The architecture of a data center.

equipment depends on the amount of resource provisioning. Therefore, Liu *et al.* [162] modeled the power consumption of the cooling supply system $p^{cooling}$ as a function of IT power consumption d (which is determined by the amount of resource provisioning) and the environment parameters, i.e., $p^{cooling} = kd^3$, where k is a coefficient which is proportional to the temperature difference between the returned hot air from the IT equipment (denoted as t_{RA}) and the outside cooling air (denoted as t_{OA}). In other words, $p^{cooling} = \alpha(t_{RA} - t_{OA})d^3$, where α is a parameter decided by the air flow rate. Abdulla *et al.* [164] also modeled the cooling supply system $p^{cooling}$ as a nonlinear function with respect to the power consumption of the IT equipment as well as the environment parameters (e.g., t_{RA} , the air flow rate and the air density). We define the cooling power efficiency η as the IT power consumption d divided by the cooling power consumption $p^{cooling}$. In other words, the amount of heat generated by the IT equipment, which consumes η units of power, can be removed by the cooling system using 1 unit of power. Thus, intuitively, when η is high (i.e., the value of $\alpha(t_{RA} - t_{OA})$ is small), in order to minimize the energy consumption of the cooling supply system, GRM should assign more resources to serve the applications, and vice versa. Moreover, if green energy is considered in the cooling aware resource management, in order to achieve the objective (i.e., minimize the brown energy usage while guaranteeing the applications' SLAs), the resource provisioning strategy would be much more complicated by incorporating the green energy generation and cooling power efficiency into the resource management.

VII. CONCLUSION

We have presented an overview on different kinds of resource management mechanisms for a data center to maximize the resource utilization. Resource management basically comprises two components: Global Resource Manager (GRM) and Local Resource Manager (LRM). GRM can essentially optimize the coarse-grained resource allocation from the global

point of view. LRM locally adjusts the resource allocation in a small time scale. Our studies indicate that proactive resource allocation by predicting the future workload can enhance the performance of resource management. Based on predicting the workloads of different objects (VM, PM or application), we have categorized the resource management mechanisms into three types: VM, PM, and application based resource management mechanisms. We have reviewed different prediction algorithms and allocation strategies recently proposed for the three management mechanisms.

Most of the resource management mechanisms have been implemented based on the assumption of the data center being homogeneous. Qualitative comparisons have been made among the three resource management mechanisms if the data center is heterogeneous. We have investigated four heterogeneity features existed in recent data centers and discuss the degree of performance degradation when three kinds of resource management mechanisms are implemented in a heterogeneous data center. It is challengeable to design an optimal resource management by considering all the heterogeneity features of a data center.

REFERENCES

- [1] M. Armbrust, *et al.*, "Above the Clouds: A Berkeley View of Cloud Computing," Dept. Electrical Engineering and Computer Sciences, University of California, Berkeley, CA, Rep. UCB/EECS, 2009.
- [2] W. Song, Z. X., Q. Chen, and H. Luo, "Adaptive resource provisioning for the cloud using online bin packing," *IEEE Transactions on Computers*, vol. 63, no. 11, pp. 2647–2660, 2014.
- [3] B. Wadhwa and A. Verma, "Energy and carbon efficient VM placement and migration technique for green cloud datacenters," in *2014 Seventh International Conference on Contemporary Computing (IC3)*, Noida, India, Aug. 7–9, 2014, pp. 189–193.
- [4] B. Paul, *et al.*, "Xen and the art of virtualization," *ACM SIGOPS Operating Systems Review*, vol. 37, no. 5, pp. 164–177, 2003.
- [5] B. Robert, L. Y. Chen, and E. Smirni, "Data centers in the wild: A large performance study," IBM, Zurich, Switzerland, Rep., Z1204–002, 2012.
- [6] C. Reiss, *et al.*, "Heterogeneity and dynamicity of clouds at scale: Google trace analysis," in *Proceedings of the Third ACM Symposium on Cloud Computing*, San Jose, CA, Oct. 14–17, 2012, pp. 7.
- [7] Traces of Google workloads. [Online]. Available: <http://code.google.com/p/googleclusterdata/>
- [8] H. Chen, P. Xiong, K. Schwan, A. Gavrilovska, and C. Xu, "A cyber-physical integrated system for application performance and energy management in data centers," in *2012 International Green Computing Conference (IGCC)*, San Jose, CA, Jun. 4–8, 2012, pp. 1–10.
- [9] M. Lin; A. Wierman, L. L. H. Andrew, and E. Thereska, "Dynamic right-sizing for power-proportional data centers," in *2011 Proceedings IEEE INFOCOM*, Shanghai, China, Apr. 10–15, 2011, pp. 1098–1106.
- [10] Z. Xu, and W. Liang, "Minimizing the Operational Cost of Data Centers via Geographical Electricity Price Diversity," in *2013 IEEE Sixth International Conference on Cloud Computing (CLOUD)*, Santa Clara, CA, Jun. 28–Jul. 3, 2013, pp. 99–106.
- [11] S. Ren, and Y. He, "COCA: Online distributed resource management for cost minimization and carbon neutrality in data centers," in *2013 International Conference for High Performance Computing, Networking, Storage and Analysis (SC)*, Denver, CO, Nov. 17–22, 2013, pp. 1–12.
- [12] X. Zhen, W. Song, and Qi Chen, "Dynamic resource allocation using virtual machines for cloud computing environment," *IEEE Transactions on Parallel and Distributed Systems*, vol. 24, no. 6, pp. 1107–1117, 2013.
- [13] Y. Song, Y. Sun, and W. Shi, "A two-tiered on-demand resource allocation mechanism for VM-based data centers," *IEEE Transactions on Services Computing*, vol. 6, no. 1, pp. 116–129, 2013.
- [14] W. Zhao, Z. Wang, and Y. Luo, "Dynamic memory balancing for virtual machines," *ACM SIGOPS Operating Systems Review*, vol. 43, no. 3, pp. 37–47, 2009.
- [15] N. M. K. Varma, D. Min, and E. Choi, "Diagnosing CPU utilization in the Xen virtual machine environment," in *2011 6th International Conference on Computer Sciences and Convergence Information Technology (ICCIT)*, Jeju Island, Korea, Nov. 29–Dec. 1, 2011, pp. 58–63.
- [16] S. Nidhi, and S. Rao, "Online Ensemble Learning Approach for Server Workload Prediction in Large Data centers," in *IEEE 11th International Conference on Machine Learning and Applications (ICMLA)*, Boca Raton, FL, Dec. 12–15, 2012, vol. 2, pp. 68–71.
- [17] S. Nidhi, and S. Rao, "Ensemble Learning for Large-Scale Workload Prediction," *IEEE Transactions on Emerging Topics in Computing*, vol. 2, no. 2, pp. 149–165, 2014.
- [18] L. Lu, and E. Smirni, "Effective resource and workload management in data centers," in *2014 IEEE Network Operations and Management Symposium (NOMS)*, Krakow, Poland, May. 5–9, 2014, pp. 1–7.
- [19] T. Wood, P. J. Shenoy, A. Venkataramani, and M. S. Yousif, "Black-box and Gray-box Strategies for Virtual Machine Migration," in *2007 4th USENIX Symposium on Networked Systems Design & Implementation (NSDI)*, Cambridge, MA, Apr. 11–13, vol. 7, pp. 17–30.
- [20] J. Jheng, F. Tseng, H. Chao, and L. Chou, "A novel VM workload prediction using Grey Forecasting model in cloud data center," in *2014 IEEE International Conference on Information Networking (ICOIN)*, Phuket, Thailand, Feb. 10–12, 2014, pp. 40–45.
- [21] Q. Kashifuddin, Y. Li, and A. Sohn, "Workload Prediction of Virtual Machines for Harnessing Data Center Resources," in *2014 IEEE 7th International Conference on Cloud Computing (CLOUD)*, Anchorage, AK, Jun. 27–Jul. 2, 2014, pp. 522–529.
- [22] NASA & World Cup data trace. [Online]. Available: <http://www.ircache.net/>.
- [23] W. Alan, Jack B. Swift, Harry L. Swinney, and John A. Vastano, "Determining Lyapunov exponents from a time series," *Physica D: Nonlinear Phenomena*, vol. 16, no. 3, pp. 285–317, 1985.
- [24] Z. Gong, X. Gu, and John Wilkes, "Press: Predictive elastic resource scaling for cloud systems," in *2010 IEEE International Conference on Network and Service Management (CNSM)*, Niagara Falls, Canada, Oct. 25–29, 2010, pp. 9–16.
- [25] N. Hiep, Z. Shen, X. Gu, S. Subbiah, and J. Wilkes, "Agile: Elastic distributed resource scaling for infrastructure-as-a-service," in *2013 Proc. of the USENIX International Conference on Automated Computing (ICAC13)*, San Jose, CA, Jun. 26–28, 2013, pp. 69–82.
- [26] K. Arijit, X. Yan, S. Tao, and N. Anerousis, "Workload characterization and prediction in the cloud: A multiple time series approach," in *2012 IEEE Network Operations and Management Symposium (NOMS)*, Maui, Hawaii, Apr. 16–20, 2012, pp. 1287–1294.
- [27] L. Chen, and Haiying Shen, "Consolidating complementary VMs with spatial/temporal-awareness in cloud data centers," in *2014 Proceedings IEEE INFOCOM*, Toronto, Canada, Apr. 27–May. 2, 2014, pp. 1033–1041.
- [28] R. N. Calheiros, *et al.*, "CloudSim: a toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms," *Software: Practice and Experience*, vol. 41, no. 1, pp. 23–50, 2011.
- [29] N. Bobroff, A. Kochut, and K. Beaty, "Dynamic placement of virtual machines for managing sla violations," in *10th IFIP/IEEE International Symposium on Integrated Network Management*, Munich, Germany, May. 21–25, 2007, pp. 119–128.
- [30] A. Beloglazov and R. Buyya, "Energy Efficient Resource Management in Virtualized Cloud Data Centers," in *2010 10th IEEE/ACM International Conference on Cluster, Cloud and Grid Computing (CCGrid)*, Melbourne, Australia, May. 17–20, pp. 826–831.
- [31] D. Gupta, R. Gardner, and L. Cherkasova, "Xenmon: Qos monitoring and performance profiling tool," Hewlett-Packard Labs, Tech. Rep. HPL-2005-187, 2005.
- [32] Infrastructure, VMware. (2006). "Resource management with VMware DRS," VMware Whitepaper. [Online]. Available: https://www.vmware.com/pdf/vmware_drs_wp.pdf
- [33] Microsoft Performance and Resource Optimization. [Online]. Available: <http://technet.microsoft.com/enus/library/cc917965.aspx>.
- [34] HP Process Resource Manager. [Online]. Available: <http://h30081.www3.hp.com/products/prm/index.html>.
- [35] IBM Redbook, "Advanced POWER Virtualization on IBM System p5: Introduction and Configuration," Jan. 2007. [Online]. Available: <http://ps-2.kev009.com/basil.holloway/ALL%20PDF/sg247940.pdf>
- [36] F. Farahnakian, T. Pahikkala, P. Liljeberg, and J. Plosila, "Hierarchical Agent-Based Architecture for Resource Management in Cloud Data Centers," in *2014 IEEE 7th International Conference on Cloud Computing (CLOUD)*, Anchorage, AK, Jun. 27–Jul. 2, 2014, pp. 928–929.

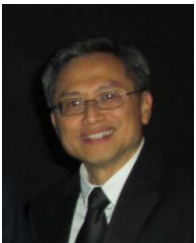
- [37] M. Sameep, and A. Neogi, "Recon: A tool to recommend dynamic server consolidation in multi-cluster data centers," in *2008 IEEE Network Operations and Management Symposium (NOMS)*, Bahia, Brazil, Apr. 7–11, 2008, pp. 363–370.
- [38] X. Meng, *et al.*, "Efficient resource provisioning in compute clouds via vm multiplexing," in *2010 ACM Proceedings of the 7th international conference on Autonomic computing*, Washington, DC, Jun. 7–11, 2010, pp. 11–20.
- [39] A. Strunk, and W. Dargie, "Does live migration of virtual machines cost energy?" in *2013 IEEE 27th International Conference on Advanced Information Networking and Applications (AINA)*, Barcelona, Spain, Mar. 25–28, 2013, pp. 514–521.
- [40] I. M. Leslie, *et al.*, "The design and implementation of an operating system to support distributed multimedia applications," *IEEE Journal on Selected Areas in Communications*, vol. 14, no. 7, pp. 1280–1297, 1996.
- [41] K. J. Duda, and D. R. Cheriton, "Borrowed-virtual-time (BVT) scheduling: supporting latency-sensitive threads in a general-purpose scheduler," *ACM SIGOPS Operating Systems Review*, vol. 33, no. 5, pp. 261–276, 1999.
- [42] Credit Scheduler-Xen. [Online]. Available <http://wiki.xen.org/wiki/CreditScheduler>.
- [43] C. Tseng, and P. Huang, "An Efficient Virtual CPU Scheduling Algorithm for Xen Hypervisor in Virtualized Environment," in *Proceedings of the 2nd International Conference on Information Technology and Computer Science*, Berkeley, CA, Jan. 10–12, 2013, vol. 25, pp. 334–338.
- [44] Z. Zheng, *et al.*, "The multi-processor load balance scheduler based on XEN," in *Proceedings of IEEE International Conference on Systems and Informatics (ICSAI)*, Yantai, China, May. 19–20, 2012, pp. 905–909.
- [45] L. Cherkasova, D. Gupta, and A. Vahdat, "Comparison of the three cpu schedulers in xen," *SIGMETRICS Performance Evaluation Review*, vol. 35, no. 2, pp. 42–51, Sep. 2007.
- [46] C. A. Waldspurger, "Memory Resource Management in VMware ESX Server," in *Proc. Fifth Symp. Operating Systems Design and Implementation (OSDI 02)*, Boston, MA, Dec. 9–11, 2002, pp. 181–194.
- [47] J. Heo, X. Zhu, P. Padala, and Z. Wang, "Memory overbooking and dynamic control of Xen virtual machines in consolidated environments," in *IFIP/IEEE International Symposium on Integrated Network Management*, Long Island, NY, Jun. 1–5, 2009, pp. 630–637.
- [48] P. Lu, and K. Shen, "Virtual Machine Memory Access Tracing with Hypervisor Exclusive Cache," in *Usenix Annual Technical Conference*, Santa Clara, CA, Jun. 17–22, 2007, pp. 29–43.
- [49] Open vSwitch. [Online]. Available: <http://openvswitch.org/>.
- [50] VMware vNetwork Distributed Switch: Migration and Configuration. [Online]. Available: <http://www.vmware.com/files/pdf/vsphere-vnetwork-ds-migration-configuration-wp.pdf>.
- [51] Cisco Nexus 1000V Series Switches. [Online]. Available: <http://www.scs.co.kr/download/Cisco%20Nexus%201000V%20Series%20Switches.pdf>.
- [52] B. Pfaff, *et al.*, "Extending Networking into the Virtualization Layer," in *2009 ACM Workshop on Hot Topics in Networks (HotNets)*, New York City, NY, Oct. 22–23, 2009.
- [53] L. Popa, *et al.*, "FairCloud: sharing the network in cloud computing," in *Proceedings of the ACM SIGCOMM 2012 conference on Applications, technologies, architectures, and protocols for computer communication*, Helsinki, Finland, Aug. 13–17, 2012, pp. 187–198.
- [54] J. P. Billaud, and A. Gulati, "hClock: Hierarchical QoS for packet scheduling in a hypervisor," in *Proceedings of the 8th ACM European Conference on Computer Systems*, Prague, Czech Republic, Apr. 14–17, 2013, pp. 309–322.
- [55] J. C. Bennett, and H. Zhang, "WF2Q: worst-case fair weighted fair queueing," in *IEEE INFOCOM '96*, San Francisco, CA, Mar. 24–28, 1996, vol. 1, pp. 120–128.
- [56] A. Demers, S. Keshav, and S. Shenker, "Analysis and simulation of a fair queueing algorithm," in *ACM SIGCOMM Computer Communication Review*, vol. 19, no. 4, pp. 1–12, 1989.
- [57] P. Goyal, H. M. Vin, and H. Chen, "Start-time fair queueing: a scheduling algorithm for integrated services packet switching networks," in *ACM SIGCOMM Computer Communication Review*, vol. 26, no. 4, pp. 157–168, 1996.
- [58] A. Kabbani, *et al.*, "Af-qcn: Approximate fairness with quantized congestion notification for multi-tenanted data centers," in *2010 IEEE 18th Annual Symposium on High Performance Interconnects (HOTI)*, Mountain View, CA, Aug. 18–20, 2010, pp. 58–65.
- [59] R. Pan, L. Breslau, B. Prabhakar, and S. Shenker, "Approximate fairness through differential dropping," *ACM SIGCOMM Computer Communication Review*, vol. 33, no. 2, pp. 23–39, 2003.
- [60] Y. Zhang and N. Ansari, "Fair Quantized Congestion Notification in Data Center Networks," *IEEE Transactions on Communications*, vol. 61, no.11, pp. 4690–4699, Nov. 2013.
- [61] X. Sun and N. Ansari, "Improving Bandwidth Efficiency and fairness in cloud computing," in *2013 IEEE Global Communications Conference (GLOBECOM)*, Atlanta, GA, Dec. 9–13, 2013, pp. 2313–2318.
- [62] H. Ballani, P. Costa, T. Karagiannis, and A. Rowstron, "Towards predictable datacenter networks," in *ACM SIGCOMM Computer Communication Review*, vol. 41, no. 4, pp. 242–253, 2011.
- [63] J. Guo, F. Liu, D. Zeng, J. C. S. Lui, and H. Jin, "A cooperative game based allocation for sharing data center networks," in *2013 Proceedings IEEE INFOCOM*, Turin, Italy, Apr.14–19, 2013, pp. 2139–2147.
- [64] B. J. Pascal, and A. Gulati, "hClock: Hierarchical QoS for packet scheduling in a hypervisor," in *Proceedings of the 8th ACM European Conference on Computer Systems*, Prague, Czech Republic, Apr. 14–17, 2013, pp. 309–322.
- [65] C. Raiciu, *et al.*, "Improving datacenter performance and robustness with multipath TCP," *ACM SIGCOMM Computer Communication Review*, vol. 41, no. 4, pp. 266–277, 2011.
- [66] N. Singh, and S. Rao, "Energy optimization policies for server clusters," in *2010 IEEE Conference on Automation Science and Engineering (CASE)*, Toronto, Canada, Aug. 22–24, 2010, pp. 293–300.
- [67] N. Singh, and S. Rao, "Modeling and reducing power consumption in large IT systems," in *2010 4th Annual IEEE Systems Conference*, San Diego, CA, Apr. 05–08, 2010, pp. 178–183.
- [68] D. Gmach, J. Rolia, L. Cherkasova, and A. Kemper, "Workload analysis and demand prediction of enterprise data center applications," in *IEEE 10th International Symposium on Workload Characterization (IISWC)*, Boston, MA, Sep. 27–29, 2007, pp. 171–180.
- [69] Z. Wang, *et al.*, "AppRAISE: application-level performance management in virtualized server environments," *IEEE Transactions on Network and Service Management*, vol. 6, no. 4, pp. 240–254, 2009.
- [70] U. Bhuvan, *et al.*, "An analytical model for multi-tier internet services and its applications," *ACM SIGMETRICS Performance Evaluation Review*, vol. 33, no. 1, pp. 291–302, 2005.
- [71] J. J. Prevost, K. Nagothu, B. Kelley, and M. Jamshidi, "Load prediction algorithm for multi-tenant virtual machine environments," in *2012 IEEE World Automation Congress (WAC)*, Puerto Vallarta, Mexico, Jun. 24–28, 2012, pp. 1–6.
- [72] F. Wei, Z. Lu, J. Wu, and Z. Cao, "RPPS: A novel resource prediction and provisioning scheme in cloud data center," in *2012 IEEE Ninth International Conference on Services Computing (SCC)*, Honolulu, Hawaii, Jun. 24–29, 2012, pp. 609–616.
- [73] V. Tom, P. Aggarwal, X. Wang, and T. Li, "Hierarchical forecasting of web server workload using sequential monte carlo training," *IEEE Transactions on Signal Processing*, vol. 55, no. 4, pp. 1286–1297, 2007.
- [74] Z. Hui, G. Jiang, K. Yoshihira, and H. Chen, "Proactive Workload Management in Hybrid Cloud Computing," *IEEE Transactions on Network and Service Management*, vol. 11, no. 1, pp. 90–100, 2014.
- [75] X. Wang, D. Lan, X. Fang, M. Ye, and Y. Chen, "A resource management framework for multi-tier service delivery in autonomic virtualized environments," in *2008 IEEE Network Operations and Management Symposium (NOMS)*, Bahia, Brazil, Apr. 7–11, 2008, pp. 310–316.
- [76] C. Italo, J. Almeida, V. Almeida, and M. Santos, "Self-adaptive capacity management for multi-tier virtualized environments," in *2007 10th IFIP/IEEE International Symposium on Integrated Network Management (IM'07)*, Munich, Germany, May. 21–25, 2007, pp. 129–138.
- [77] M. Massimiliano, *et al.*, "Green web services: Improving energy efficiency in data centers via workload predictions," in *2013 IEEE 2nd International Workshop on Green and Sustainable Software (GREENS)*, San Francisco, CA, May. 18–26, 2013, pp. 8–15.
- [78] D. Peter, *et al.*, "Modellus: Automated modeling of complex internet data center applications," *ACM Transactions on the Web (TWEB)*, vol. 6, no. 2, pp. 8, 2012.
- [79] P. Kimish, M. Annavaram, and M. Pedram, "NFRA: Generalized Network Flow-Based Resource Allocation for Hosting Centers," *IEEE Transactions on Computers*, vol. 62, no. 9, pp. 1772–1785, 2013.
- [80] Amazon EC2 Instances. [Online]. Available: <http://aws.amazon.com/ec2/instance-types/>
- [81] Microsoft Azure Instances. [Online]. Available: <http://azure.microsoft.com/en-us/pricing/details/virtual-machines/>
- [82] H. N. Van, H. Frederic, D. Tran, and J. Menaud, "Autonomic virtual resource management for service hosting platforms," in *Proceedings of IEEE 2009 ICSE Workshop on Software Engineering Challenges of Cloud Computing*, Vancouver, Canada, May. 23, 2009, pp.1–8.

- [83] S. Ying, Y. Sun, H. Wang, and X. Song, "An adaptive resource flowing scheme amongst VMs in a VM-based utility computing," in *2007 7th IEEE International Conference on Computer and Information Technology (CIT)*, Aizu-Wakamatsu, Fukushima, Oct. 16–19, 2007, pp. 1053–1058.
- [84] L. Liu, *et al.*, "A novel performance preserving VM Splitting and Assignment Scheme," in *2014 IEEE International Conference on Communications (ICC)*, Sydney, Australia, Jun. 10–14, 2014, pp. 4215–4220.
- [85] M. Kevin, J. Filliben, and C. Dabrowski, "Comparing vm-placement algorithms for on-demand clouds," in *2011 IEEE Third International Conference on Cloud Computing Technology and Science (CloudCom)*, Athens, Greece, Nov. 29–Dec. 1, 2011, pp. 91–98.
- [86] Y. Zhang and N. Ansari, "Heterogeneity aware dominant resource assistant heuristics for virtual machine consolidation," in *IEEE Global Communications Conference (GLOBECOM)*, Atlanta, GA, Dec. 09–13, 2013, pp. 1297–1302.
- [87] A. Caprara, "Properties of some ILP formulations of a class of partitioning problems," *Discrete applied mathematics*, vol. 87, no. 1, pp.11–23, 1998.
- [88] H. Jin, D. Pan, J. Xu, and N. Pissinou, "Efficient VM placement with multiple deterministic and stochastic resources in data centers," in *2012 IEEE Global Communications Conference (GLOBECOM)*, Anaheim, CA, Dec. 3–7, 2012, pp. 2505–2510.
- [89] S. Chaisiri, B.S. Lee, and D. Niyato, "Optimal virtual machine placement across multiple cloud providers," in *2009 IEEE Asia-Pacific Services Computing Conference (APSCC)*, Biopolis, Singapore, Dec. 7–11, 2009, pp. 103–110.
- [90] J. Chase, R. Kaewpuang, Y. Wen, and D. Niyato, "Joint virtual machine and bandwidth allocation in software defined network (SDN) and cloud computing environments," in *2014 IEEE International Conference on Communications (ICC)*, Sydney, Australia, Jun. 10–14, pp. 2969–2974.
- [91] M. Cardosa, A. Singh, H. Pucha, and A. Chandra, "Exploiting Spatio-Temporal Tradeoffs for Energy-Aware MapReduce in the Cloud," *IEEE Transactions on Computers*, vol. 61, no. 12, pp. 1737–1751, 2012.
- [92] J. Xu, and J. A. B. Fortes, "Multi-Objective Virtual Machine Placement in Virtualized Data Center Environments," *Green Computing and Communications (GreenCom)*, 2010 *IEEE/ACM Int'l Conference on & Int'l Conference on Cyber, Physical and Social Computing (CPSCom)*, Hangzhou, China, Dec. 18–20, 2010, pp. 179–188.
- [93] F. Machida, M. Kawato, and Y. Maeno, "Redundant virtual machine placement for fault-tolerant consolidated server clusters," in *2010 IEEE Network Operations and Management Symposium (NOMS)*, Osaka, Japan, Apr. 19–23, 2010, pp.32–39.
- [94] P. Pradeep, *et al.*, "Adaptive control of virtualized resources in utility computing environments," *ACM SIGOPS Operating Systems Review*, vol. 41, no. 3, pp. 289–302, 2007.
- [95] K.Su, *et al.*, "Affinity and Conflict-Aware Placement of Virtual Machines in Heterogeneous Data Centers," in *2015 IEEE Twelfth International Symposium on Autonomous Decentralized Systems (ISADS)*, Taichung, Taiwan, Mar. 25–27, 2015, pp. 289–294.
- [96] J. Hwang, S. Zeng, F. Y. Wu, and T. Wood, "Benefits and challenges of managing heterogeneous data centers," in *2013 IFIP/IEEE International Symposium on Integrated Network Management (IM 2013)*, Ghent, Belgium, May. 27–31, 2013, pp.1060–1065.
- [97] E. Rotem, *et al.*, "Energy management of highly dynamic server workloads in an heterogeneous data center," in *2014 24th International Workshop on Power and Timing Modeling, Optimization and Simulation (PATMOS)*, Palma de Mallorca, Spain, Sept. 29–Oct. 1, 2014, pp. 1–5.
- [98] S. Zhang, Y. Liu, B. Wang, and R. Zhang, "Analysis and modeling of dynamic capacity provisioning problem for a heterogeneous data center," in *2013 Fifth International Conference on Ubiquitous and Future Networks (ICUFN)*, Da Nang, Vietnam, Jul. 2–5, 2013, pp. 785–790.
- [99] S. Zhang, Y. Liu, B. Wang, and R. Zhang, "A Novel Resource Allocation Algorithm for a Heterogeneous Data Center," in *2013 International Conference on Information Science and Applications (ICISA)*, Suwon, Korea, Jun. 24–26, 2013, pp.1–4.
- [100] ESX Server Performance and Resource Management for CPU-Intensive Workloads, VMware white paper. [Online]. Available: http://www.vmware.com/pdf/ESX2_CPU_Performance.pdf.
- [101] Y. Park, R. Scott, and S. Sechrest, "Virtual Memory versus File Interface for Large, Memory-Intensive Scientific Applications," in *Proceedings of the 1996 ACM/IEEE Conference on Supercomputing*, Pittsburgh, PA, Nov. 17–22, 1996, pp.53–53.
- [102] C. Lee, M. C. Chen, and R. Chang, "HiPEC: high performance external virtual memory caching," in *Proceedings of the 1st USENIX conference on Operating Systems Design and Implementation*, Monterey, CA, Nov. 14–17, 1994, pp. 1–12.
- [103] V. Udaykiran, P. Mohapatra, R. Iyer, and K. Kant, "Improving cache performance of network intensive workloads," in *Proceedings of 2001 IEEE International Conference on Parallel Processing*, Valencia, Spain, Sep. 3–7, 2001, pp. 87–94.
- [104] A. Gambi, M. Pezze, and M. Young, "SLA Protection models for virtualized data centers," in *SEAMS '09 ICSE Workshop on Software Engineering for Adaptive and Self-Managing Systems*, Vancouver, BC, May 18–19, 2009, pp. 10–19.
- [105] Z. Zhu, J. Bi, H. Yuan, and Y. Chen, "SLA Based Dynamic Virtualized Resources Provisioning for Shared Cloud Data Centers," in *2011 IEEE International Conference on Cloud Computing (CLOUD)*, Washington, DC, Jul. 4–9, 2011, pp. 630–637.
- [106] H. Qian, and D. Medhi, "Data center resource management with temporal dynamic workload," in *2013 IFIP/IEEE International Symposium on Integrated Network Management*, Ghent, Belgium, May. 27–31, 2013, pp. 948–954.
- [107] Z. I. M. Yusoh, and M. Tang, "A penalty-based grouping genetic algorithm for multiple composite SaaS components clustering in Cloud," in *2012 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*, Seoul, Korea, Oct. 14–17, 2012, pp. 1396–1401.
- [108] Y. Chen, *et al.*, "Managing server energy and operational costs in hosting centers," *ACM SIGMETRICS Performance Evaluation Review*, vol. 33, no. 1, pp. 303–314, 2005.
- [109] L. Dan, *et al.*, "RDCM: Reliable data center multicast," in *2011 Proceedings IEEE INFOCOM*, Shanghai, China, Apr. 10–15, 2011, pp. 56–60.
- [110] D. Fesehaye, and K. Nahrstedt, "SCDA: SLA-aware cloud datacenter architecture for efficient content storage and retrieval," in *2014 IEEE 7th International Conference on Cloud Computing (CLOUD)*, Anchorage, AK, Jun. 27–Jul. 2, pp. 120–127.
- [111] K. Gaj, *et al.*, "Performance evaluation of selected job management systems," in *IEEE International Parallel and Distributed Processing Symposium*, Fort Lauderdale, FL, Apr. 15–19, 2002, vol. 2, pp. 0254–0254.
- [112] J. Zhan, *et al.*, "Cost-aware cooperative resource provisioning for heterogeneous workloads in data centers," *IEEE Transactions on Computers*, vol. 62, no. 11, pp. 2155–2168, 2013.
- [113] G. Anshul, Y. Chen, D. Gmach, M. Arlitt, and M. Marwah, "Minimizing data center SLA violations and power consumption via hybrid resource provisioning," in *2011 International Green Computing Conference and Workshops (IGCC)*, Orlando, FL, Jul. 25–28, 2011, pp. 1–8.
- [114] G. S. Kumar, S. K. Gopalaiyengar, and R. Buyya, "SLA-based resource provisioning for heterogeneous workloads in a virtualized cloud datacenter," *Algorithms and Architectures for Parallel Processing*, Melbourne, Australia, Oct. 24–26, 2011, pp. 371–384.
- [115] Z. Liu, and S. Cho, "Characterizing machines and workloads on a Google cluster," in *2012 41st International Conference on Parallel Processing Workshops (ICPPW)*, Pittsburgh, PA, Sep. 10–13, 2012, pp. 397–403.
- [116] Z. Wang, X. Zhu, and S. Singhal, "Utilization and SLO-based control for dynamic sizing of resource partitions," in *16th IFIP/IEEE International Workshop on Distributed Systems: Operations and Management*, Barcelona, Spain, Oct. 24–26, 2005, pp. 133–144.
- [117] S. Lee, *et al.*, "Validating heuristics for virtual machines consolidation," Microsoft Research, MSR-TR-2011-9, 2011.
- [118] L. Chen, *et al.*, "MTAD: A Multitarget Heuristic Algorithm for Virtual Machine Placement," *International Journal of Distributed Sensor Networks*, 2014.
- [119] A. Beloglazov, J. Abawajy, and R. Buyya, "Energy-aware resource allocation heuristics for efficient management of data centers for cloud computing," *Future generation computer systems*, vol. 28, no. 5, pp. 755–768, 2015.
- [120] N.T. Hieu, M. Di Francesco, and A. Yla-Jaaski, "A virtual machine placement algorithm for balanced resource utilization in cloud data centers," in *2014 IEEE 7th International Conference on Cloud Computing (CLOUD)*, Anchorage, AK, Jun. 27–Jul. 2, 2014, pp. 474–481.
- [121] Y. Feng, B. Li, and B. Li, "Bargaining towards maximized resource utilization in video streaming datacenters," in *2012 Proceedings IEEE INFOCOM*, Orlando, FL, Mar. 25–30, 2012, pp. 1134–1142.
- [122] X. Sun, *et al.*, "Multi-dimensional resource integrated scheduling in a shared data center," in *2011 31st IEEE International Conference on Distributed Computing Systems Workshops (ICDCSW)*, Minneapolis, MN, Jun. 20–24, 2011, pp. 7–13.
- [123] M.H. Ferdous, M. Murshed, R.N. Calheiros, and R. Buyya, "Virtual machine consolidation in cloud data centers using ACO metaheuristic," in *Euro-Par 2014 Parallel Processing*, Springer International Publishing, pp. 306–317, 2014.

- [124] E. Arzuaga and D. R. Kaeli, "Quantifying load imbalance on virtualized enterprise servers," in *Proceedings of the first joint WOSP/SIPEW international conference on Performance engineering*, San Jose, CA, Jan.28–30, 2010, pp. 235–242.
- [125] M. Mishra and A. Sahoo, "On theory of vm placement: Anomalies in existing methodologies and their mitigation using a novel vector based approach," in *2011 IEEE International Conference on Cloud Computing (CLOUD)*, Washington, D.C., Jul. 4–9, 2011, pp. 275–282.
- [126] A. Singh, M. Korupolu, and D. Mohapatra, "Server-storage virtualization: integration and load balancing in data centers," in *Proceedings of the 2008 ACM/IEEE conference on Supercomputing*, Austin, TX, Nov. 15–21, 2008, pp. 1–12.
- [127] S. Lee, *et al.*, "Validating heuristics for virtual machines consolidation," Microsoft Research, MSR-TR-2011-9, 2011.
- [128] Y. Song, *et al.*, "A service-oriented priority-based resource scheduling scheme for virtualized utility computing," in *Proc. Intl Conf. High Performance Computing (HiPC)*, Bangalore, India, Dec. 17–20, 2008, pp. 220–231.
- [129] S.H. Lim, B. Sharma, B.C. Tak, and C.R. Das, "A dynamic energy management scheme for multi-tier data centers," in *2011 IEEE International Symposium on Performance Analysis of Systems and Software (ISPASS)*, Austin, TX, Apr. 10–12, 2011, pp. 257–266.
- [130] C.T. Huynh and E.N. Huh, "Prediction technique for resource allocation in Micro Data Center," in *IEEE International Conference on Information Networking (ICOIN)*, Siem Reap, Cambodia, Jan. 12–14, pp. 483–486.
- [131] K. Le, R. Bianchini, M. Martonosiz, and T.D. Nguyen, "Cost- and energy-aware load distribution across data centers," in *22nd ACM Symposium on Operating Systems Principles*, Big Sky, MT, Oct. 11–14, 2009.
- [132] D. Prangchumpol, S. Sanguansintukul, and P. Tantasanawong, "Improving Heterogeneous Workload Performance in Server Virtualization Based on User Behaviors," *Journal of Convergence Information Technology*, vol. 7, no. 16, pp. 544–552, 2012.
- [133] C. Vazquez, R. Krishnan, and E. John, "Time Series Forecasting of Cloud Data Center Workloads for Dynamic Resource Provisioning," *Journal of Wireless Mobile Networks, Ubiquitous Computing, and Dependable Applications (JoWUA)*, vol. 6, no. 3, pp. 87–110, 2015.
- [134] O. Shai, E. Shmueli, and D. Feitelson, "Heuristics for resource matching in intel's compute farm," in *Proc. of the 17th International Workshop on Job Scheduling Strategies for Parallel Processing (JSSPP13)*, Boston, MA, May, 2014, pp. 116–135.
- [135] R. Calheiros, E. Masoumi, R. Ranjan, and R. Buyya, "Workload Prediction Using ARIMA Model and Its Impact on Cloud Applications' QoS," *IEEE Transactions on Cloud Computing*, vol. 3, no. 4, pp. 449–458, Oct.–Dec. 2015.
- [136] N.R. Herbst, N. Huber, S. Kounev, and E. Amrehn, "Self-adaptive workload classification and forecasting for proactive resource provisioning," *Concurrency and Computation: Practice and Experience*, vol. 26, no. 12, pp. 2053–2078, 2014.
- [137] M.G. Rabbani, *et al.*, "On tackling virtual data center embedding problem," in *2013 IFIP/IEEE International Symposium on Integrated Network Management (IM 2013)*, Ghent, Belgium, May 27–31, 2013, pp. 177–184.
- [138] C. Guo, *et al.*, "SecondNet: a data center network virtualization architecture with bandwidth guarantees," in *Proceedings of the 6th International Conference*, Bordeaux, France, Jun. 28–Jul. 01, 2011, pp. 15–15.
- [139] K. Su, L. Xu, C. Chen, W. Chen, and Z. Wang, "Affinity and conflict-aware placement of virtual machines in heterogeneous data centers," in *2015 IEEE Twelfth International Symposium on Autonomous, Taichung, Taiwan*, Mar. 25–27, 2015, pp. 289–294.
- [140] J. Chen, *et al.*, "AAGA: Affinity-Aware Grouping for Allocation of Virtual Machines," in *013 IEEE 27th International Conference on Advanced Information Networking and Applications (AINA)*, Barcelona, Spain, Mar. 25–28, 2013, pp. 235–242.
- [141] X. Meng, V. Pappas, and L. Zhang, "Improving the scalability of data center networks with traffic-aware virtual machine placement," in *2010 Proceedings IEEE INFOCOM*, San Diego, CA, Mar. 15–19, 2010, pp. 1–9.
- [142] D. Kliazovich, P. Bouvry, and S. U. Khan, "DENS: data center energy-efficient network-aware scheduling," *Cluster computing*, vol. 16, no. 1, pp. 65–75, 2013.
- [143] J.W. Jiang, T. Lan, S. Ha, M. Chen, and M. Chiang, "Joint VM placement and routing for data center traffic engineering," in *2012 Proceedings IEEE INFOCOM*, Orlando, FL, Mar. 25–30, 2012, pp. 2876–2880.
- [144] M. Chen, S.C. Liew, Z. Shao, and C. Kai, "Markov approximation for combinatorial network optimization," in *2010 Proceedings IEEE INFOCOM*, San Diego, CA, Mar. 15–19, 2010, pp. 1–9.
- [145] D. Belabed, S. Secci, G. Pujolle, and D. Medhi, "Striking a Balance Between Traffic Engineering and Energy Efficiency in Virtual Machine Placement," *IEEE Transactions on Network and Service Management*, vol. 12, no. 2, pp. 202–216, 2015.
- [146] IEEE Standard for Local and Metropolitan Area Networks–Virtual Bridged Local Area Networks Amendment 10: Provider Backbone Bridge Traffic Engineering, IEEE Std. 802.1Qay–2009, Aug. 2009.
- [147] D. Allan, *et al.*, "Shortest path bridging: efficient control of larger ethernet networks," *IEEE Communications Magazine*, vol. 48, no. 10, pp. 128–135, 2010.
- [148] J. Touch and R. Perlman, "Transparent interconnection of lots of links (TRILL): Problem and applicability statement," RFC 5556, 2009.
- [149] A. Reinert, B. Sans, and S. Secci, "Design optimization of the petaweb architecture," *IEEE/ACM Transactions on Networking*, vol. 17, no. 1, pp. 332–345, 2009.
- [150] A. Stage and T. Setzer, "Network-aware migration control and scheduling of differentiated virtual machine workloads," in *Proceedings of the 2009 ICSE workshop on software engineering challenges of cloud computing*, Vancouver, Canada, May 23, 2009, pp. 9–14.
- [151] I. Goiri, *et al.*, "Parasol and greenswitch: Managing datacenters powered by renewable energy," in *ACM SIGARCH Computer Architecture News*, vol. 41, no. 1, pp. 51–64, 2013.
- [152] R. Bianchini, "Leveraging renewable energy in data centers: present and future," in *Proceedings of the 21st international symposium on High-Performance Parallel and Distributed Computing*, Delft, Netherlands, Jun. 18–22, 2012, pp. 135–136.
- [153] I. Goiri, *et al.*, "GreenHadoop: leveraging green energy in data-processing frameworks," in *Proceedings of the 7th ACM european conference on Computer Systems*, Bern, Switzerland, Apr. 10–13, 2012, pp. 57–70.
- [154] A. Kiani and N. Ansari, "Towards Low-Cost Workload Distribution for Integrated Green Data Centers," *IEEE Communications Letters*, vol. 19, no. 1, pp. 26–29, Jan. 2015.
- [155] Y. Zhang and N. Ansari, "On Architecture Design, Congestion Notification, TCP Incast and Power Consumption in Data Centers," *IEEE Communications Surveys and Tutorials*, vol. 15, no. 1, pp. 39–64, First Quarter, 2013.
- [156] I. Goiri, *et al.*, "Greenslot: scheduling energy consumption in green datacenters," in *Proceedings of 2011 International Conference for High Performance Computing, Networking, Storage and Analysis*, Seattle, WA, Nov. 12–18, 2011, p. 20.
- [157] I. Goiri, *et al.*, "Matching renewable energy supply and demand in green datacenters," *Ad Hoc Networks*, vol. 5, pp. 520–534, 2015.
- [158] W. Deng, *et al.*, "Multigreen: Cost-minimizing multi-source datacenter power supply with online control," in *Proceedings of the fourth international conference on Future energy systems*, Berkeley, CA, May. 22–24, 2013, pp. 149–160.
- [159] T. Han and N. Ansari, "On Optimizing Green Energy Utilization for Cellular Networks with Hybrid Energy Supplies," *IEEE Transactions on Wireless Communications*, vol.12, no.8, pp.3872–3882, August 2013.
- [160] X. Zhang, *et al.*, "Energy Consumption Modeling of Data Center IT Room with Distributed Air Flow," in *20th IEEE International Conference on Emerging Technologies and Factory Automation (ETFA'2015)*, Luxembourg, Sep. 8–11, 2015.
- [161] R. Brown, "Report to congress on server and data center energy efficiency: Public law 109–431," Lawrence Berkeley National Laboratory, 2008.
- [162] Z. Liu, *et al.*, "Renewable and cooling aware workload management for sustainable data centers," in *ACM SIGMETRICS Performance Evaluation Review*, vol. 40, no. 1, pp. 175–186, 2012.
- [163] J. Yao, *et al.*, "Adaptive Power Management Through Thermal Aware Workload Balancing in Internet Data Centers," *IEEE Transactions on Parallel and Distributed Systems*, vol. 26, no. 9, pp. 2400–2409, Sept. 2015.
- [164] M.A. Abdulla, S. Pasricha, A. Maciejewski, and H.J. Siegel, "Power and thermal-aware workload allocation in heterogeneous data centers," *IEEE Transactions on Computers*, vol. 64, no. 2, pp. 477–491, 2015.



Xiang Sun [S'13] received a B.E. degree in electronic and information engineering and an M.E. degree in technology of computer applications from Hebei University of Engineering, Hebei, China. He is currently working towards the Ph.D. degree in electrical engineering at the New Jersey Institute of Technology (NJIT), Newark, New Jersey. His research interests include mobile edge computing, big data networking, green edge computing and communications, and cloud computing.



Nirwan Ansari [S'78, M'83, SM'94, F'09] is Distinguished Professor of Electrical and Computer Engineering at the New Jersey Institute of Technology (NJIT). He has also been a visiting (chair) professor at several universities such as High-level Visiting Scientist at Beijing University of Posts and Telecommunications.

Professor Ansari is authoring *Green Mobile Networks: A Networking Perspective* (John Wiley, 2016) with T. Han, and co-authored two other books.

He has also (co-)authored more than 500 technical publications, over one third published in widely cited journals/magazines. He has guest-edited a number of special issues covering various emerging topics in communications and networking. He has served on the editorial/advisory board of over ten journals. His current research focuses on green communications and networking, cloud computing, and various aspects of broadband networks.

Professor Ansari was elected to serve in the IEEE Communications Society (ComSoc) Board of Governors as a member-at-large, has chaired ComSoc technical committees, and has been actively organizing numerous IEEE International Conferences/Symposia/Workshops. He has frequently delivered keynote addresses, distinguished lectures, tutorials, and invited talks. Some of his recognitions include IEEE Fellow, several Excellence in Teaching Awards, best paper awards, the NCE Excellence in Research Award, the ComSoc AHSN TC Outstanding Service Recognition Award, the NJ Inventors Hall of Fame Inventor of the Year Award, the Thomas Alva Edison Patent Award, Purdue University Outstanding Electrical and Computer Engineer Award, and designation as a COMSOC Distinguished Lecturer. He has also been granted over 30 U.S. patents.

He received a Ph.D. from Purdue University in 1988, an MSEE from the University of Michigan in 1983, and a BSEE (summa cum laude with a perfect GPA) from NJIT in 1982.



Ruopeng Wang is a senior software engineer at Huawei, which is a leading global information and communications technology "ICT" solutions provider. In the past four years, his major work focuses on cloud computing, network function virtualization (NFV), and algorithms (resource allocation, prediction, etc.). He earned a Ph.D. degree in Space Physics from Wuhan University in 2012.