Green Network - Cloud: Allocation of Resources And Migration In Cloud Network

J.Sivaranjani[1], R.Malar[2]
[1, 2] Department of Computer Science and Engineering
[1, 2] Indira Institute of Engineering and Technology, Pandur, Thiruvallur

Abstract*- In computing clouds, burstiness of a virtual machine (VM) workload widely exists in real applications, where spikes usually occur aperiodically with low frequency and short duration. This could be effectively handled through dynamically scaling up/down in a virtualization-based computing cloud; however, to minimize energy consumption, VMs are often highly consolidated with the minimum number of physical machines (PMs) used. In this case, to meet the dynamic runtime resource demands of VMs in a PM, some VMs have to be migrated to some other PMs. i.e., reserving a certain amount of extra resources on each PM to avoid live migrations, and propose a novel server consolidation algorithm, QUEUE. We first model the resource requirement pattern of each VM as a two-state Markov chain to capture burstiness, then we design a resource reservation strategy for each PM based on the stationary distribution of a Markov chain. Finally, we present QUEUE, a complete server consolidation algorithm with a reasonable time complexity. Achieves a better balance between performance and energy consumption in comparison with other commonly-used consolidation algorithms.*

Keywords- Bursty workload, Markov chain, Resource reservation, Server consolidation.

I. INTRODUCTION

CLOUD computing has been gaining more and more trac-tion in the past few years, and it is changing the way we access and retrieve information [1]. The recent emergence of virtual desktop [2] has further elevated the impor-tance of computing clouds.

As a crucial technique in modern computing clouds, virtualization enables one phys-ical machine (PM) to host many performance isolated vir-tual machines (VMs). It greatly benefits a computing cloud where VMs running various applications are aggregated together to improve resource utilization. It has been shown in previous work [3] that, the cost of energy consumption, e.g., power supply, and cooling, occupies a significant frac-tion of the total operating costs in a cloud.

Therefore, making optimal utilization of underlying resources to reduce the energy consumption is becoming an important issue [4], [5]. To cut back the energy consumption in clouds, server consolidation is proposed to tightly pack VMs to reduce the number of running PMs; however, VMs' performance may be seriously affected if VMs are not appropriately placed, especially in a highly consolidated cloud.We observed that the variability and burstiness of VM workload widely exists in modern computing clouds, as evidenced in prior studies [4], [6], [7], [8], [9]. Take a typical web server for example, burstiness may be caused by flash crowed with bursty incoming requests.

We all know that VMs should be provisioned with resources commensurate with their workload requirements [10], which becomes more complex when considering workload variation. As shown in Fig. 1, two kinds of resource provisioning strate-gies are commonly used to deal with workload burstiness provisioning for peak workload and provisioning for nor-mal workload. Provisioning for peak workload is favour-able to VM performance guarantee, but it undermines the advantage of elasticity from virtualization and may lead to low resource utilization [1], [8], [9].

On the other hand, live migra-tion moves some VM(s) to a relatively idle PM, when local resizing is not able to allocate enough resources.

However, in a highly consolidated computing cloud where resource contention is generally prominent among VMs, live migra-tion may cause significant service downtime; furthermore, it also incurs noticeable CPU usage on the host PM [12], which probably degrades the co-located VMs' performance.

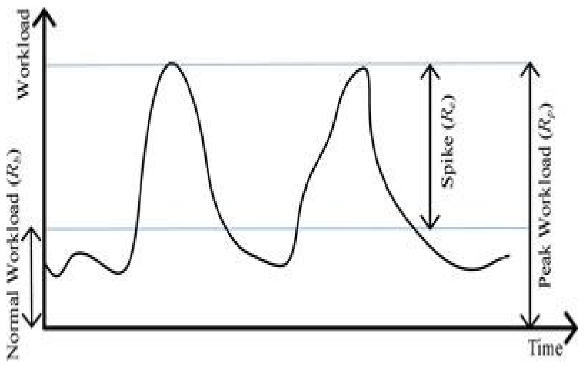


Figure 1. An example of workload with bursty spikes.

II. THE PROPOSED APPROACH

In this paper, we propose to reserve some extra resources on each PM to accommodate bursty workload [13]. In doing so, when a resource spike occurs, VMs can be quickly recon-figured to the new level of resource requirement through local resizing with minimal overheads, instead of being migrated to some other PMs. Hence, the number of live migrations could be reduced considerably and the overall performance of a computing cloud could be improved.

Specifically, we investigate the problem of minimizing the amount of extra resources reserved on each PM during server consolidation while the overall performance is proba-bilistically guaranteed. By “probabilistically guaranteed”, we mean that, the fraction of time within which the aggre-gated workloads of a PM exceed its physical capacity is not larger than a threshold. Imposing such a threshold rather than conducting live migration upon PM’s capacity over-flow is a way to tolerate minor fluctuations of resource usage (like the case of CPU usage) and to break the tradeoff between utilization and performance. Then, our problem can be formulated as an optimization, wherein the goal is to minimize the amount of resource reserved on each PM, and the constraint is that the capacity violation ratio of every PM is not larger than a predetermined threshold.We use a two-state Markov chain to capture the bursti-ness of workload [7], and also shows how to learn the chain parameters. Inspired by the serving windows in queueing theory [14], we abstract the resources reserved on each PM for workload spikes as blocks. Denoting by uðtÞ the number of busy blocks at time t on a PM, we show that a sequence of uð0Þ, u ð1Þ, u ð2Þ; . . . has the Markov property, namely that, the next state only depends on the current state and not on the past sequence of states. Then we develop a novel server consolidation algorithm, QUEUE, based on the sta-tionary distribution of this Markov chain. We also show how to further improve the effectiveness of QUEUE with more careful treatment of heterogenous workload spikes. Simulation and testbed results show that, QUEUE improves the consolidation ratio by up to 45 percent with large spike size and around 30 percent with normal spike size compared with the strategy that provisions for peak workload, and achieves a better balance between perfor-mance and energy consumption in comparison with other commonly-used consolidation algorithms.

The contribu-tions of our paper are three-fold.

1) To the best of our knowledge, we are the first to quantify the amount of reserved resources with con-sideration of workload burstiness. We propose to use the two-state Markov chain model to capture workload burstiness, and we present a formal prob-lem description and its NP-completeness.
2) We develop a novel algorithm, QUEUE, for bursti-ness-aware resource reservation, based on the sta-tionary distribution of a Markov chain. We also show how to cope with heterogeneous spikes to further improve the performance of QUEUE.
3) Extensive simulations and testbed experiments.

III. RELATED WORK

Most of prior studies [3], [15], [16] on server consolidation focused on minimizing the number of active PMs from the perspective of bin packing (BP). A heterogeneity-aware resource management system for dynamic capacity provi-sioning in clouds was developed in [17]. Stable resource allocation in geographically-distributed clouds was consid-ered in [18]. Network-aware virtual machine placement was considered in [19]. Scalable virtual network models were designed in [8], [20] to allow cloud tenants to explicitly specify computing and networking requirements to achieve predictable performance.

In a computing cloud, burstiness of workload widely exists in real applications, which becomes an inevitable characteristic in server consolidation [1], [4], [6], [7], [21. Different from them, in our model a lower limit of provisioning is set at the normal workload level which effectively prevents VM interfer-ence caused by unpredictable behaviors from co-located VMs.

Markov chain was used to inject burstiness into a tradi-tional benchmark in [7]. Several works [5], [28], [29] studied modeling and dynamic provisioning of bursty workload in cloud computing. A previous study [30] proposed to reserve a constant level of hardware resource on each PM to tolerate workload fluctuation; but how much resource should be reserved was not given. To the best of our knowl-edge, we are the first to quantify the amount of reserved resources with consideration on various, but distinct, workload burstiness.

IV. MODELING VIRTUAL MACHINE WORKLOAD

Two-State Markov Chain

It has been well recognized in previous studies [4], [6], [7] that VM workload is time-varying with bursty spikes, as shown in Fig. 1. Several works [9], [10], [22], [23], [24], [25] modeled the workload of a VM as a random variable,

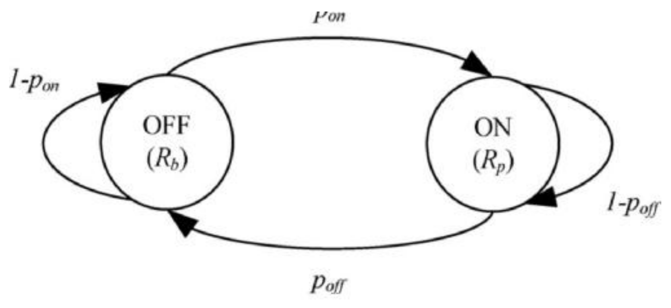


Figure 2. Two-state Markov chain. The “ON” state represents peak work-load (Rp) while the “OFF” state represents normal workload (Rb). pon and poff are the state switch probabilities.

which follows the Bernoulli distribution in [9] or normal distribution in [10], [24], [25]. Different from these works, we model the workload of a VM as a two-state Markov chain, which takes the additional dimension of time into consideration, and thus describes the characteristics of spikes more precisely.

We denote the resource require-ments of peak workload, normal workload, and workload spike by R_p, R_b, and R_e, respectively, where R_e ¼ R_p R_b as demonstrated in Fig. 1. The “ON” state represents peak workload while the “OFF” state represents normal work-load. We use p_{on} and p_{off} to denote the state switch proba-bilities. More specifically, if a VM is in the ON state, then the probability of it switching to OFF at the next time is p_{off} , and remaining ON is 1 p_{off} . Similarly if a VM is in the OFF state, then the probability of it switching to ON at next time is p_{on} and remaining ON is 1 p_{on}. We emphasize that this model is able to describe the characteristics of spikes precisely—intuitively, R_e denotes the size of a spike, and p_{on} denotes the frequency of spike occurrence. Thus, each VM can be described by a four-tuple

$$V_i \text{ ¼ } p_{on}^i;\ p_{off}^i;\ R_b^i;\ R_e^i\ ;\ 81\quad i\quad n; \qquad (1)$$

where n is the number of VMs.

Learning Model Parameters

This section provides a simple strategy for cloud tenants to generate model parameters for their VM workload. It con-sists of two phases.

First, a cloud tenant must have the workload traces and guarantees that they will be consistent with the realistic deployment in computing clouds.

Second, given a VM workload trace, a cloud tenant gen-erates a four-tuple.

$$p_{on} \text{ ¼ } \frac{S_{FN}}{S_{FN} \text{ þ } S_{FF}}\ ;\ \text{and } p_{off} \text{ ¼ } \frac{S_{NF}}{S_{NF} \text{ þ } S_{NN}}\ :$$

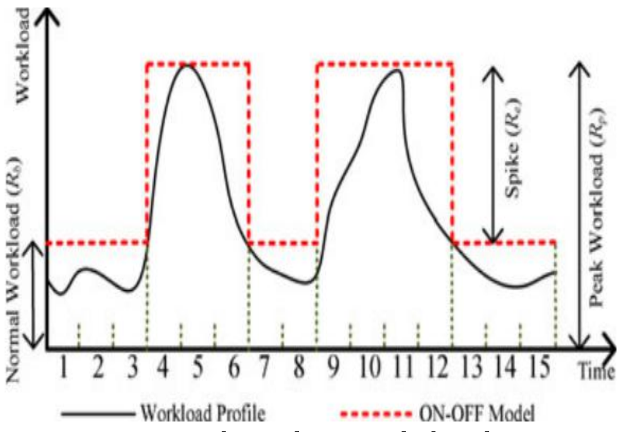


Figure 3. Given the predetermined Rb and Re, we conservatively round the solid black curve up to the dashed red curve, based on which we can calculate pon and poff .

Potential Benefits

The two-state Markov chain model allows cloud tenants to flexibly control the tradeoff between VM performance and deployment cost through adjusting Rb and Re.

When a tenant wants to maximize VM performance, the ten-ant should choose a large Rb and a small Re. As we will show later in this paper, there may be multiple workload spikes that share some common physical resources. Thus, when the aggregated amount of workload spikes that simulta-neously occur is larger than the amount of the shared common resources, capacity overflow happens and VM per-formance is probably affected.

When a tenant wants to minimize deployment cost, the tenant should choose a small Rb and a large Re. By ”deployment cost”, we mean the fee which is paid by a cloud tenant to a cloud provider. Since physical resources are opportunistically shared among multiple workload spikes, the charge for workload spike should be smaller than that for normal workload [9]. Therefore, decreasing Rb helps tenants to reduce the deployment cost.

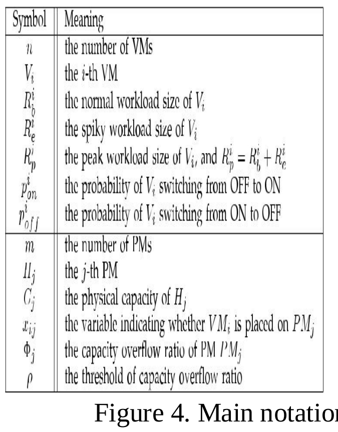
Our model is also a tradeoff between modeling complexity and precision. We could model time-varying workload by three-state or even more states of Markov chain, which should capture the workload bustiness more precisely; however, the complexity in learning model parameters and allocating physical resources increases as well, which may complicate the interactions between cloud providers and tenants.

IV. PROBLEM FORMULATION

We consider a computing cloud with one-dimensional resource; for scenarios with multi-dimensional resources, we provide a few remarks in Section 8. There are m physical machines in the computing cloud, and each PM is described by its physical capacity

$$H_j ¼ ðC_jÞ; 81 \quad j \quad m: \qquad (2)$$

We use a binary matrix X ¼ ½x$_{ij}$&$_{n\ m}$ to represent the results of placing n VMs on m PMs: x$_{ij}$ ¼ 1, if V$_i$ is placed on H$_j$, and 0 otherwise. We assume that the workloads of VMs are mutually independent. Let W$_i$ðtÞ be the resource



Symbol	Meaning
n	the number of VMs
V_i	the i-th VM
R_b^i	the normal workload size of V_i
R_e^i	the spiky workload size of V_i
R_p^i	the peak workload size of V_i, and $R_p^i = R_b^i + R_e^i$
p_{on}^i	the probability of V_i switching from OFF to ON
p_{off}^i	the probability of V_i switching from ON to OFF
m	the number of PMs
H_j	the j-th PM
C_j	the physical capacity of H_j
x_{ij}	the variable indicating whether VM_i is placed on PM_j
Φ_j	the capacity overflow ratio of PM PM_j
ρ	the threshold of capacity overflow ratio

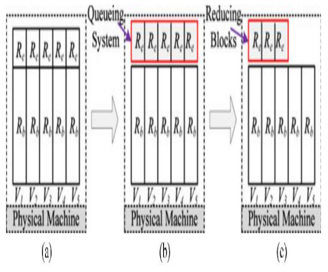

Queueing System
Reducing Blocks
R_e R_e R_e R_e R_e
R_b R_b R_b R_b R_b
V_1 V_2 V_3 V_4 V_5
Physical Machine
(a)
(b)
(c)

Figure 4. Main notations for quick reference.

requirements of V$_i$ at time t. According the Markov chain model, we have

$$W_iðtÞ ¼ \begin{cases} R^i_b & \text{if } V_i \text{ is in the “OFF” state at time t;} \\ R_p^i & \text{if } V_i \text{ is in the “ON” state at time t:} \end{cases}$$

Then, the aggregated resource requirement of VMs on PM H$_j$ is $\sum^n_{i ¼1} x_{ij} W_i ðtÞ$.

Let P_{CO}^t j indicate whether the capacity overflow happens on PM H$_j$ at time t, i.e.,

$$CO_j^t ¼ \begin{cases} 0 & \text{if } \sum^N_{i¼1} x_{ij} W_i ðtÞ > C_j; \\ 1 & \text{otherwise:} \end{cases}$$

Intuitively, the results of VM placement should guarantee that the capacity constraint is satisfied on each PM at the beginning of the time period of interest, i.e.

$$CO^0_j ¼ 0; 81 \quad j \quad m:$$

We now can define our metric for probabilistic perfor-mance guarantee—capacity overflow ratio (COR), which is the fraction of time that the aggregated workloads of a PM exceed its physical capacity. Denoting the capacity overflow ratio of PM H$_j$ as F$_j$, we have

$$F_j ¼ \frac{1 \quad \sum_{t}^{T} CO_j^t}{P \quad T};$$

V. BURSTINESS-AWARE RESOURCE RESERVATION

Overview of QUEUE

We propose reserving a certain amount of physical resour-ces on each PM to accommodate workload spikes. The main idea is to abstract the reserved spaces as blocks.
However, we may find that a certain number of blocks are idle for the majority of the time in Fig. 5b, so we can reduce the number of blocks while only incurring very few capacity violations.

Therefore, our goal becomes reserving minimal number of blocks on each PM while the perfor-mance constraint in Eq. (3) is still satisfied.

Resource Reservation Strategy for a Single PM

In this section, We focus on resource reservation for a single PM. For the sake of convenience, we set the size of each block as the size of the maximum spike of all co-located VMs on a PM. In Section 6, we will present how to cope with heteroge-nous workload spikes in an effort to further improve the per-formance of QUEUE. We also assume that all VMs have the same state switch probabilities, i.e., pion ¼ pon and pioff ¼ poff , for all 1 i n. In Section 8, we will show how to cluster VMs when they have different state switch probabilities.

Suppose there are k VMs on the PM of interest and ini-tially each VM Vi occupies Rib resources. We initialize the number of blocks reserved on this PM as k, and our objec-tive is to reduce the number of blocks to K (K < k), while the capacity overflow ratio F does not exceed the threshold r. Let uðtÞ be the number of busy blocks at time t, implying that, there are uðtÞ VMs in the ON state and ðk uðtÞÞ VMs in the OFF state. Let OðtÞ and IðtÞ denote the number of VMs that switch state from ON to OFF (i.e., VMs that leave the queueing system) and from OFF to ON (i.e., VMs that enter the queueing system) at time t, respectively.

VI. COPING WITH HETEROGENOUS SPIKES

In this section, we present how to improve the consolidation performance of QUEUE through more careful treatment of heterogenous workload spikes. In doing so, a total of 28 units of resour-ces are reserved, which is less than that in the previous case.We, therefore, have the following intuition: on each PM, we can try to partition the co-located VMs into several groups and consider them separately, so as to improve QUEUE by reducing the amount of resources reserved for workload spikes.A key problem in achieving our goal is how to partition a set of k VMs into non-overlapped groups, i.e., how to parti-tion an integer k, which is an interesting and important problem in number theory [32].

VII. PERFORMANCE EVALUATION

In this section, we conduct extensive simulations and testbed experiments to evaluate the proposed algorithms under different settings and reveal insights of the proposed design performance.

Simulation Setup

Two commonly-used packing strategies are considered here, which both use the First Fit Decrease heuristic for VM placement. The first strategy is to provision VMs for peak workload (FFD by Rp), while the second is to provision VMs for normal workload (FFD by Rb). Provisioning for peak workload is usually applied for the initial VM place-ment [1], where cloud tenants choose the peak workload as the fixed capacity of the VM to guarantee application per-formance. On the other hand, provisioning for normal workload is usually applied in the consolidation process, since at runtime the majority of VMs are in the OFF state, i.e., most of the VMs only have normal workloads.

We consider both the situations without and with live migration, where different metrics are used to evaluate the runtime performance. For experiments without live migra-tion, where only local resizing is allowed to dynamically provision resources, we use the capacity overflow ratio defined in Section 4 as the performance metric. Next, in our testbed experiments, we add live migration to our system to simulate a more realistic computing cluster, in which the number of migrations reflects the quality of performance, and the number of active PMs reflects the level of energy consumption.

Simulation Results

We first evaluate the computation cost of our algorithm briefly, and then quantify the reduction of the number of running PMs, as well as compare the runtime performance with two commonly-used packing strategies.To investigate the performance of our algorithm in vari-ous settings, three kinds of workload patterns are used for each experiment: Rb ¼ Re, Rb > Re and Rb < Re, which denote workloads with normal spike size, small spike size, and large spike size, respectively. It will be observed later that, the workload pattern of VMs does affect the packing result, number of active PMs, and number of migrations.According to the results in Section 5.3, the time complex-ity of QUEUE is Oðd4 þ n log n þ mnÞ. In

Fig. 10, we present the experimental computation cost of QUEUE with reason-able d and n values. We see that, our algorithm incurs very few overheads with moderate n and d values. The cost vari-ation with respect to n is not even distinguishable in the mil-lisecond-level.

We mention that, there are very few PMs with CORs slightly higher than r in each experiment. This is because a Markov chain needs some time to enter into its stationary distribution. Though we did not theoretically evaluate whether the chain constructed in Section 5 is rapid-mixing, in our experiments, we find that the time period before the chain enters into its stationary distribution is very short.

Testbed Experiment

We use Xen Cloud Platform (XCP) 1.3 [35] as our testbed to enable live migration in our system. XCP is an open-source cloud platform of its commercial counterpart XenServer. Our proposed scheme can be easily integrated into any existing enterprise-level computing cloud since it simply computes the amount of reserved resources on each PM. A total of 15 machines (Intel Core i5 Processor with four 2.8 GHz cores and 4 GB memory) are used. Ubuntu 12.04 LTS Server Edition is installed both on the PMs and VMs. The resource type in QUEUE can be any one-dimen-sional resource such as CPU, memory, disk I/O, network bandwidth, or any combination of them that can be mapped to one dimension. The architecture of our testbed. We developed three main modules: consolidation module, per-formance monitor, and schedule module. QUEUE is imple-mented in the consolidation module. To construct the MinN array, QUEUE gets VM specifications from cloud users and the predetermined system parameters (e.g., r, and d) from XCP API. The MinN array can be reused as long as all of k, pon, poff , and r remain unchanged. The FidOptPat algorithm (Algorithm 3) may be time-consuming.

We are also interested in studying the effect of different workload patterns, thus, Rb and Re are classified into three types: small (S), medium (M), and large (L). A certain amount of users can be accommodated for each size—400 for small, 800 for medium and 1,600 for large. Fig. 15 shows the details of various workload patterns in our testbed experiments

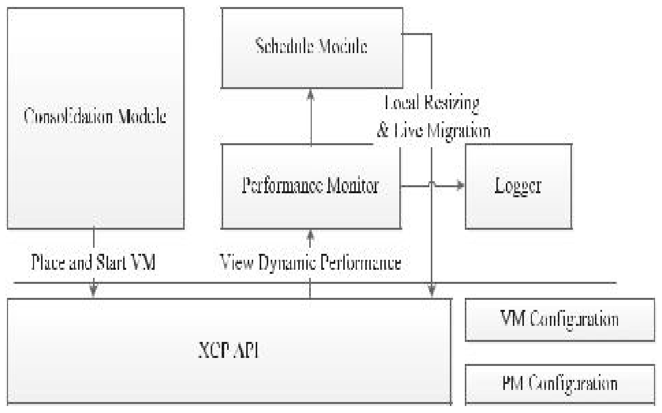


Figure 5. The architecture of our testbed

VIII. IMPLEMENTATION

A modular design reduces complexity, facilities change and results in easier implementation by encouraging parallel development of different part of system. Software with effective modularity is easier to develop because function may be compartmentalized and interfaces are simplified. Software architecture embodies modularity that is software is divided into separately named and addressable components called modules that are integrated to satisfy problem requirements.

Modularity is the single attribute of software that allows a program to be intellectually manageable. The five important criteria that enable us to evaluate a design method with respect to its ability to define an effective modular design are: Modular decomposability, Modular Comps ability, Modular Understandability, Modular continuity, Modular Protection.

Our implementation of the proposed scheme consists of six modules: User Registration, Cloud Server Deployment, Intermediate Server Deployment, Green Computing Setup, Migration of Virtual Server, Cache Server Implementation.

IX. CONCLUSION

In a highly consolidated computing cloud, the VM performance is prone to degradation without an appropriate VM placement strategy, if various and distinct burstiness exists. To alleviate this problem, we have to activate more PMs, leading to more energy consumption. To balance the performance and energy consumption with respect to bursty workload, we propose to reserve a certain amount of resources on each PM that form a queuing system to accommodate burstiness. To quantify the amount of reserved resources is not a trivial problem. In this paper, we propose a burstiness-aware server consolidation algorithm based on the two-state Markov chain. We use a probabilistic performance constraint and show that the proposed algorithm is able to guarantee this performance constraint.

X. FUTURE ENHANCEMENT

Possible future research topics for the dynamic migration problem in the cloud computing platform include, the generation, learning and mutation method of Bucket Code can be further improved to reduce its asymmetry property. One example is calculating proper probability for generating the first part of the Bucket Code, proposing a better way for codes to learn from each other instead of just learn from the best one.

REFERENCES

[1] M. Armbrust, A. Fox, R. Griffith, A. D. Joseph, R. Katz, A.Konwinski, G. Lee, D. Patterson, A. Rabkin, I. Stoica, and M. Zaharia, "A view of cloud computing," Commun. ACM, vol. 53, no. 4, pp. 50–58, 2010.

[2] M.-H. Oh, S.-W. Kim, D.-W. Kim, and S.-W. Kim, "Method and architecture for virtual desktop service," U.S. Patent 20 130 007 737, 2013.

[3] M. Marzolla, O. Babaoglu, and F. Panzieri, "Server consolidation in clouds through gossiping," in Proc. IEEE

Int. Symp. World Wireless, Mobile Multimedia Netw., pp. 1–6.

[4] W. Vogels, “Beyond server consolidation,” ACM Queue, vol. 6, no. 1, pp. 20–26, 2008.

[5] N. Bobroff, A. Kochut, and K. Beaty, “Dynamic placement of virtual machines for managing SLA violations,” in Proc. IFIP/IEEE Int. Symp. Integr. Netw. Manag., 2007, pp. 119–128.

[6] S. Kandula, S. Sengupta, A. Greenberg, P. Patel, and R. Chaiken, “The nature of data center traffic: Measurements & analysis,” in Proc. ACM 9th ACM SIGCOMM Conf. Internet Meas. Conf., 2009, pp. 202–208.

[7] N. Mi, G. Casale, L. Cherkasova, and E. Smirni, “Injecting realistic burstiness to a traditional client-server benchmark,” in Proc. IEEE 6th Int. Conf. Auton. Comput., 2009, pp. 149–158.

[8] D. Xie, N. Ding, Y. C. Hu, and R. Kompella, “The only constant ischange: Incorporating time-varying network reservations in data centers,” in Proc. ACM SIGCOMM, 2012, pp. 199–210.

[9] S. Zhang, Z. Qian, J. Wu, S. Lu, and L. Epstein, “Virtual network embedding with opportunistic resource sharing,” IEEE Trans. Parallel Distrib. Syst., vol. 25, no. 3, pp. 816–827, Mar. 2014.

[10] M. Wang, X. Meng, and L. Zhang, “Consolidating virtual machines with dynamic bandwidth demand in data centers,” in Proc. IEEE INFOCOM, 2011, pp. 71–75.

[11] A. Verma, G. Kumar, and R. Koller, “The cost of reconfiguration in a cloud,” in Proc. 11th Int. Middleware Conf. Ind. Track, 2010, pp. 11–16.