Towards Multi-Resource Physical Machine Provisioning for IaaS Clouds

Lei Wei, Bingsheng He
School of Computer Engineering
Nanyang Technological University
Singapore

Chuan Heng Foh
Centre for Communication Systems Research
University of Surrey
UK

***Abstract*—Virtualization has been an enabling technology for IaaS (Infrastructure as a Service) Clouds. Physical machine (PM) provisioning is a key problem for IaaS cloud providers on their resource utilization and quality of service to users. Proper provisioning is able to ensure the service quality while conserving unnecessary power consumption from over-provisioned PMs. However, the effectiveness of PM provisioning in current IaaS providers such as Amazon and Rackspace is severely limited by that they offer virtual machines with proportional resource provisioning on different resource types (including CPU, memory and disk etc). Such a rigid offering cannot satisfy diversified user applications in the cloud, and can cause significant over-provision on PMs in order to satisfy users' requirement on all resource types. This paper argues a more flexible approach that IaaS providers should offer virtual machines with flexible combinations on multiple resource types. We further formulate the problem of multiple resource virtual machine allocations for IaaS clouds, and develop analytical models to predict the suitable number of PMs while satisfying a predefined quality-of-service requirement. Experiments show that the proposed approach can significantly increase the resource utilization, with a reduction on the number of active PMs by 27% on average.**

I. Introduction

Infrastructure as a Service (IaaS) is one of the most basic cloud-service models. Enabled by the virtualization technology, IaaS cloud providers offer users with resources in the form of virtual machines (VMs). Physical Machine (PM) provisioning is an important and challenging problem for IaaS clouds. The number of active PMs is a direct indicator of total energy consumption [8], [5]. Since PMs are generally not energy-proportionality as they still consume power even when they are idle [1], switching idle PMs to an inactive mode (i.e. power off or hibernate) is an effective approach for energy saving. Energy consumption is believed to surpass the hardware cost of IT equipment in data centers [3]. Moreover, provisioning a suitable number of active PMs while satisfying the predefined quality-of-service level is able to minimize the cost of cloud providers. On the one hand, under-provisioning can result in massive violations on service level agreement (SLAs) and lead to customer loss. On the other hand, over-provisioning causes low utilization of resources and results in energy waste. Therefore, this paper investigates whether and how we improve the effectiveness of PM provisioning for IaaS cloud providers.

Many research efforts have been devoted to the problem of PM provisioning. Many approaches have been developed based on mathematical models [2], [5], and some other works are based on heuristic approaches [6], [8]. We refer readers to a recent survey [4] for more recent resource provisioning and management techniques. Heuristic based methods cannot precisely predict actual required resources according to the pre-defined quality-of-service requirement. Despite many studies on addressing the problem of PM provisioning on IaaS clouds, their effectiveness is severely limited by the virtual machine offering given by current IaaS cloud providers. Major IaaS providers such as Amazon EC2 and Rackspace offer the virtual machines with proportional resource provisioning on different resource types (including CPU, memory and disk etc). For example, Amazon's virtual machines almost double the CPU capability, memory capacity and storage from m1.small, m1.medium, m1.large to m1.xlarge, and the prices double accordingly. Under this scheme, cloud providers need to perform resource provisioning according to the dominant resource type. Therefore, essentially, cloud providers perform PM provisioning with a single resource provisioning scheme (SRP). In SRP, cloud providers provision VMs to users according to one single type of dominant resource (e.g., number of CPU cores or RAM size), the amounts of other resources are proportional to the dominant resource.

The advantage of SRP is that it makes the resource allocation and VM scheduling simple. However, SRP has two major drawbacks. First, it does not fit the diversified resource requirements from different applications in the cloud. According to the case studies in major cloud vendors (such as Amazon[1]), applications from different sectors have been deployed to IaaS clouds. They can have different and also dynamic requirements on different resource types. Some of them are CPU bound and some others are memory bound or I/O bound. Second, there is a severe drawback in the resource utilization. SRP provisions sufficient resources according to the resource type with maximum requirement, which leads to significant PM over-provisioning.

To address the shortcomings of SRP, we propose a dynamic multiple resource provisioning (DMRP) approach to optimize the resource utilization for IaaS cloud providers. DMRP allows

[1] http://aws.amazon.com/solutions/case-studies/

978-1-4799-2003-7/14/$31.00 ©2014 IEEE

TABLE I
NOTATIONS OF MODEL

N	Number of PMs in the considered data center
K	Number of resource types
L	Number of options for each resource type
R_i	Total amount for type-i resource in a PM, $i = 1, 2, ..., K$
r_i	Available resource for type-i resource in a PM
V	Total number of VM types
v_j	Number of type-j VMs in a PM, $j = 1, 2, ..., V$
b_j	Requested resource vector for type-j VM request, $j = 1, 2, ..., V$
b_{ji}	Requested resource amount of type-i resource for type-j VM request, $j = 1, 2, ..., V$
λ_j	Arrival rate of type-j requests
μ_j	Service rate of type-j requests
P_B	Blocking probability
α	The predefined P_B threshold

cloud providers to offer flexible resource combinations of different resource types for users. With such an flexibility in acquiring VMs, the resource provision of VMs can better match different resource requirements from users, especially when users' applications have unproportional requirements on different resource types.

DMRP potentially can give a good and tight provisioning according to user requirement. However, provisioning the suitable number of active PMs for dynamic workloads in DMRP becomes much more complicated than SRP. The major challenge is that multiple dimensional resource types make the states of resource allocations in IaaS clouds grow in an exponential manner. It is difficult to study in a precise prediction model. Based on Markov Chain, we develop an analytical model to dynamically predict the number of active PMs. Particularly, we define *blocking probability* (the probability that a VM request would be blocked) to quantify to describe the quality-of-service level. Our analytical model can guide the PM provisioning by controlling the blocking probability in a predefined threshold. The workload prediction uses the existing method Weighted Moving Average (WMA).

We perform some preliminary studies by simulating an IaaS cloud with four basic workload patterns. The experimental experiments show that 1) our analytical model is highly accurate in predicting the suitable number of PMs, with an error smaller than 3%, and 2) the proposed approach can significantly increase the resource utilization, with a reduction on the number of active PMs by 27% on average than SRP.

Organization. The rest of the paper is organized as follows. We give an overview on DMRP in Section II, followed by the analytical model in Section III. We present the experimental results in Section IV, and conclude in Section V.

II. SYSTEM OVERVIEW

In this section, we present the application scenario and the design overview of DMRP. The parameters throughout the paper are listed in Table I.

DMRP is an online dynamic multiple resource provisioning system for IaaS cloud providers. We formulate new notions of VM offerings to users and also define the quality-of-service level. Moreover, it has a model-guided PM provisioning to minimize the number of active PMs while satisfying the predefined quality-of-service level.

New notions of VM offerings. As traditional fixed type provisioning in SRP scheme cannot support multiple resource provisioning, it is necessary to develop a new VM offering approach to cover the flexible provisioning according to resource demands of different resource types. DMRP offers a series of amounts of resource for each resource type and let users pick one from each resource type. We depict the above scenario with the following simple illustrating example. A cloud system may identify three resources (i.e. $K = 3$), namely CPU (c), memory (m), and disk space (d) for the charging, which can be expressed by a triple tuple $< c, m, d >$. Each PM may possess a total of $< 16(cores), 32(GB), 512(GB) >$ of CPU, memory, and disk space, respectively. Each of these resources may allows four options (i.e. $L = 4$), which are $[1, 2, 4, 8]$ for CPU, $[2, 4, 8, 16]$ for memory, and $[32, 64, 128, 256]$ for disk space. This configuration permits a total of 64 ($V = L^K$) different types of VMs, namely $< 1, 2, 32 >$, $< 1, 2, 64 >$, $< 1, 2, 128 >$,..., $< 8, 16, 256 >$.

There are three issues that are worth further discussions. First, compared with the current VM offerings for major cloud providers, the new notion of VM offering gives more flexibility in resource provisioning. Still, it is much simpler than supporting arbitrary VM resource provisioning. Second, current virtualization platforms such as Xen and OpenStack are ready to support this new notion. Third, it is an open problem on developing a pricing model for this new notion. This paper focuses on the PM provisioning problem, and leaves the pricing model as our future work.

Quality of service. We define an indicator for the availability of data center namely blocking probability, P_B, to measure whether a data center can serve a particular user request. Formally, P_B is defined as the ratio of number of blocked requests to the total number of requests. Cloud providers may aim to offer $P_B = 0$ by an accurate provisioning of n PMs. However, maintaining $P_B = 0$ in a cloud system makes resource utilization very low because the workloads are not always in peak periods. Thus, cloud providers may adjust P_B within a tolerable threshold and gain more saving from provisioning a much smaller number of PMs. Moreover, due to the unpredictable nature of workloads, instantaneous overloading is inevitable, and this will result in temporary service blockage to some requests. Thus, we define a threshold α as a minimum supporting blocking probability for the system.

PM provisioning. There are several basic approaches to allocate VMs based on their requirements, which are First Fit (FF), Best Fit (BF), Worst Fit (WF) and Second Fit (SF), etc. Experiments [7], [9] show that different basic VM placement algorithms are almost the same for resource usage of data center. Since the allocation of VMs is not the main concern of this paper, we simply choose FF as our VM placement algorithm. Our analytical model can be extended to other VM placement algorithms. In FF, the provisioned n PMs are

Algorithm 1 Provisioning algorithm in DMRP

1: **if** Current time is beginning of a time slot **then**
2: Predict the workload;
3: **for** $n = 1$ **to** N **do**
4: Compute P_B with model in Section III;
5: **if** $P_B \leq \alpha$ **then**
6: Provision n;
7: Break;

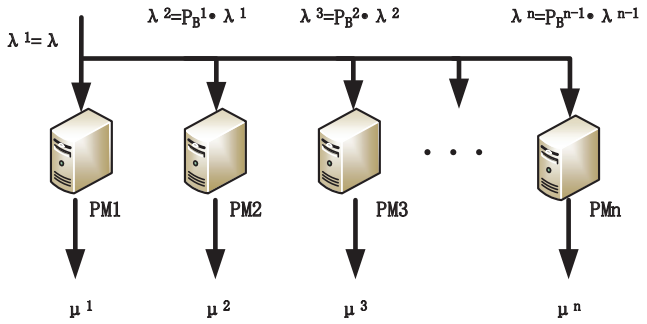

Fig. 1. Model method.

collected to form a list of active PMs, the order of PMs in the list is not critical. For each request, DMRP searches the list for available resources for the allocation. If the allocation is successful, the requested type of VM will be created. Otherwise, if there is no PM in the list that can offer adequate resources, this request will be blocked.

To determine the provisioning decision n, DMRP runs the workload prediction and then the Markov Chain based model at beginning of every time slot, as shown in Algorithm 1. Since P_B monotonously decreases as n increases, we can always find the smallest n that satisfies the condition where $P_B \leq \alpha$ if such a condition exists. By this mechanism, we can obtain the optimal solution for provisioning the number of active PMs in cloud systems while satisfying the predefined quality of service.

In the following, we shall introduce the Markov Chain model to predict the blocking probability given a number of PMs in DMRP.

III. System Modeling

To form an analytical relationship between operational configurations and performance outcomes, we develop a Continuous-Time Markov Chain model describing the evolution of resource usage. With the model, we can determine the optimal number of PMs for cost-effective provisioning while satisfying the predefined blocking probability.

While it is possible to devise a Markov Chain to describe the evolution of all resources on each PM in the data center, capturing all details requires huge state spaces, especially when a relatively large number N of PMs are involved. For instance, if all resources are detailed in the model, the computation and space complexity grows exponentially with the number of PMs and the number of resource types. Due to

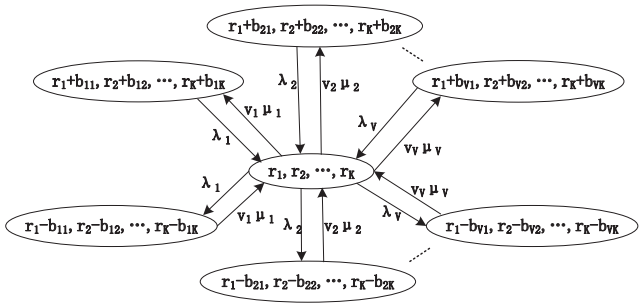

Fig. 2. State transitions in the model.

the online requirement of PM provisioning, this approach is infeasible and not scale in practice.

Instead, our rationale is to obtain an analytical model that is sufficiently accurate and has reasonable runtime overhead. Thus, we seek modeling simplification to make our proposed model tractable and practical. As shown in Fig. 1, we design a Markov Chain that models each PM individually, where the input of each PM is the blocked requests from previous one. λ^a, μ^a and P_B^a $(a = 1, 2, ..., n)$ denotes the arrival rate, service rate and blocking probability of the a^{th} PM. λ is the total arrival to the data center. This design is based on the premise that FF algorithm searches the PM list sequentially to find a PM that can host the requested VM. In other words, a request will search for resources starting from the first PM on the list. If the requested resources are available on the PM, it will be served by that PM. Otherwise, it will continue to search on the second PM for resources. This search process continues until all PMs on the list are exhausted. Then the request is *blocked* by the cloud provider.

Let us introduce the details of our model. Our modeling approach is that we model each PM individually into a common Markov Chain. This approach reduces state space demand significantly while capturing the detailed resource evolution in each PM. Let $\vec{\lambda} = [\lambda_1, \lambda_2, ..., \lambda_V]$ and $\vec{\mu} = [\mu_1, \mu_2, ..., \mu_V]$ denote the average arrival rate and service rate, respectively. λ_j and μ_j $(j = 1, 2, ..., V)$ represent the arrival and service rate of type-j request with resource demand $\vec{b_j}$. Let $\vec{R} = [R_1, R_2, ..., R_K]$ be the resource capacities for K types of resource. Then we develop a K-dimensional Markov Chain for each PM where the state $\{r_1, r_2, ..., r_K\}$ represents the available multiple resources in a PM and $\{r_i \leq R_i, i = 1, 2, ..., K\}$. Thus, the total utilized resources in the PM is $\vec{R} - \vec{r}$. Since the utilized resources may be shared by several VMs, we describe the expected number of active VMs in type-j by v_j.

In Markov Chain, a state transition is triggered by an allocation or release of resources. While an allocation of resources is a result of a request arrival with successful allocation, a release of resources is a result of termination of a VM. The overall state transition map is given in Fig. 2 where the evolution of the system state is governed by job arrivals and departures illustrated by arrows where the weights are arrival or departure rates. The node in the center represents the current state and

others are state nodes that are directly associated with current state. By such transitions, all the states exist in a PM are able to reached. Next, we introduce the details of the model.

We first denote $R\{\vec{s}|\vec{r}\}$ to be the rate of transition from state $\{\vec{r}\}$ to state $\{\vec{s}\}$. Then, we define an indicator function for our convenience to represent, where

$$I(\vec{r}) = \begin{cases} 1, & 0 \leq r_i \leq R_i, i = 1, 2, ..., K \\ 0, & \text{otherwise.} \end{cases} \quad (1)$$

Upon an arrival of a type-j request, the request is accepted if there is enough resource to initialize the requested VM. The transition condition is $r_i \geq b_{ji}, i = 1, 2, ..., K$. The transition occurs in this case from state $\vec{r}$ to state $\vec{r} - \vec{b_j} = \{r_1 - b_{j1}, r_2 - b_{j2}, ..., r_K - b_{jK}\}$ with rate

$$R\{\vec{r} - \vec{b_j}|\vec{r}\} = \lambda_j \cdot I(\vec{r} - \vec{b_j}). \quad (2)$$

The release of resources occurs when a VM is terminated. The rate that resource release occurs depends on the number of VMs currently active in a PM. At a particular state $\{\vec{r}\}$, there is $\vec{R} - \vec{r}$ amount of resources being utilized, and the utilization of a type-j VM is λ_j/μ_j. Therefore, we can determine the expected number of active type-j VMs by

$$v_j = \sum_{i=1}^{K} [\frac{\frac{\lambda_j}{\mu_j}}{\sum_{j=0}^{V} \frac{\lambda_j}{\mu_j} \cdot b_{ji}} \cdot (R_i - r_i)]/K. \quad (3)$$

Upon a depart of a type-j request, the system state will transit from state $\{\vec{r}\}$ to state $\{\vec{r} + \vec{b_j}\}$ and the possible transitions triggered by VM departures are given by (4) with $j = 1, 2, ..., V$,

$$R\{\vec{r} + \vec{b_j}|\vec{r}\} = v_j \cdot \mu_j. \quad (4)$$

The above state transitions construct a Markov Chain with a total number of states of T, which is expressed by

$$T = \prod_{i=1}^{K} ((R_i/b_{zi}) + 1) \quad (5)$$

where $\vec{b_z}$ is the resource amount for VM type with the minimum resource demand among V types of VMs.

With the above formulas, a T^2 infinitesimal generator matrix for the Markov Chain model (Q) can be constructed. The steady-state probability of each state, $p(\vec{r})$, can be solved numerically by following balance equations,

$$p(\vec{r}) \cdot [\sum_{j=1}^{V} (v_j \cdot \mu_j \cdot I(\vec{r} + \vec{b_j}) + \lambda_j \cdot I(\vec{r} - \vec{b_j}))] =$$
$$\sum_{j=1}^{V} [p(\vec{r} + \vec{b_j}) \cdot \lambda_j \cdot I(\vec{r} + \vec{b_j}) + p(\vec{r} - \vec{b_j}) \cdot v_j \cdot \mu_j \cdot I(\vec{r} - \vec{b_j})]. \quad (6)$$

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

$$\vec{U} = \sum_{r_1=0}^{R_1} \sum_{x_r=0}^{R_2} ... \sum_{r_K=0}^{R_K} p(\vec{r}) \cdot (\vec{R} - \vec{r}) \quad (7)$$

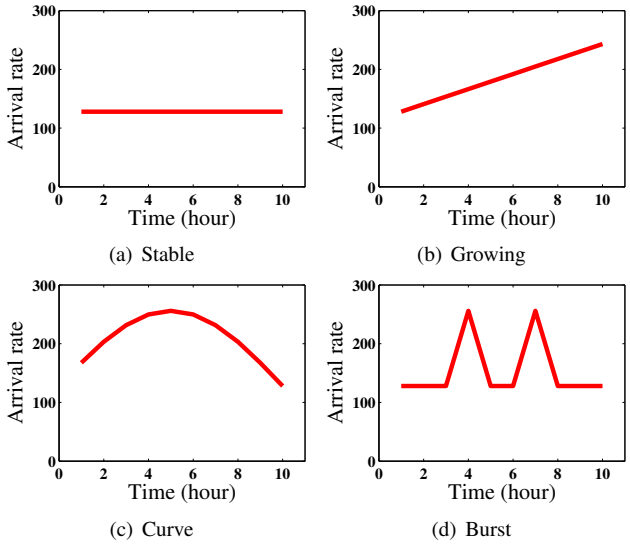

Fig. 3. The four synthetic workloads against time slot (hour).

where U_i is the i^{th} resource type.

We now analyze the blocking probability. Precisely, we determine the blocking probability of each type of requests arriving to the PM. It measures the probability that a request of a particular type is rejected for service due to unavailable resources. Let $P_{B,j}$ be the blocking probability of a type-j requests, and P_B is that of any type of requests which measures the overall blocking probability of the system. The quantity of $P_{B,j}$ can be computed by

$$P_{B,j} = \sum_{r_1=0}^{R_1} \sum_{r_2=0}^{R_2} ... \sum_{r_K=0}^{R_K} p(\vec{r}) \cdot I(\vec{b_j} - \vec{r}) \quad (8)$$

and the overall blocking probability of the system is

$$P_B = \frac{\sum_{j=1}^{V} P_{B,j} \lambda_j}{\sum_{j=1}^{V} \lambda_j}. \quad (9)$$

The computation and space complexity of model is $O(N \cdot T^3)$ and $O(T^2)$, respectively. We will experimentally evaluate the runtime overhead.

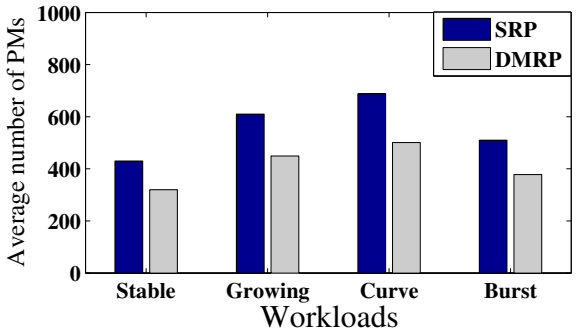

Fig. 4. The overall results of SRP and DMRP with four synthetic workloads.

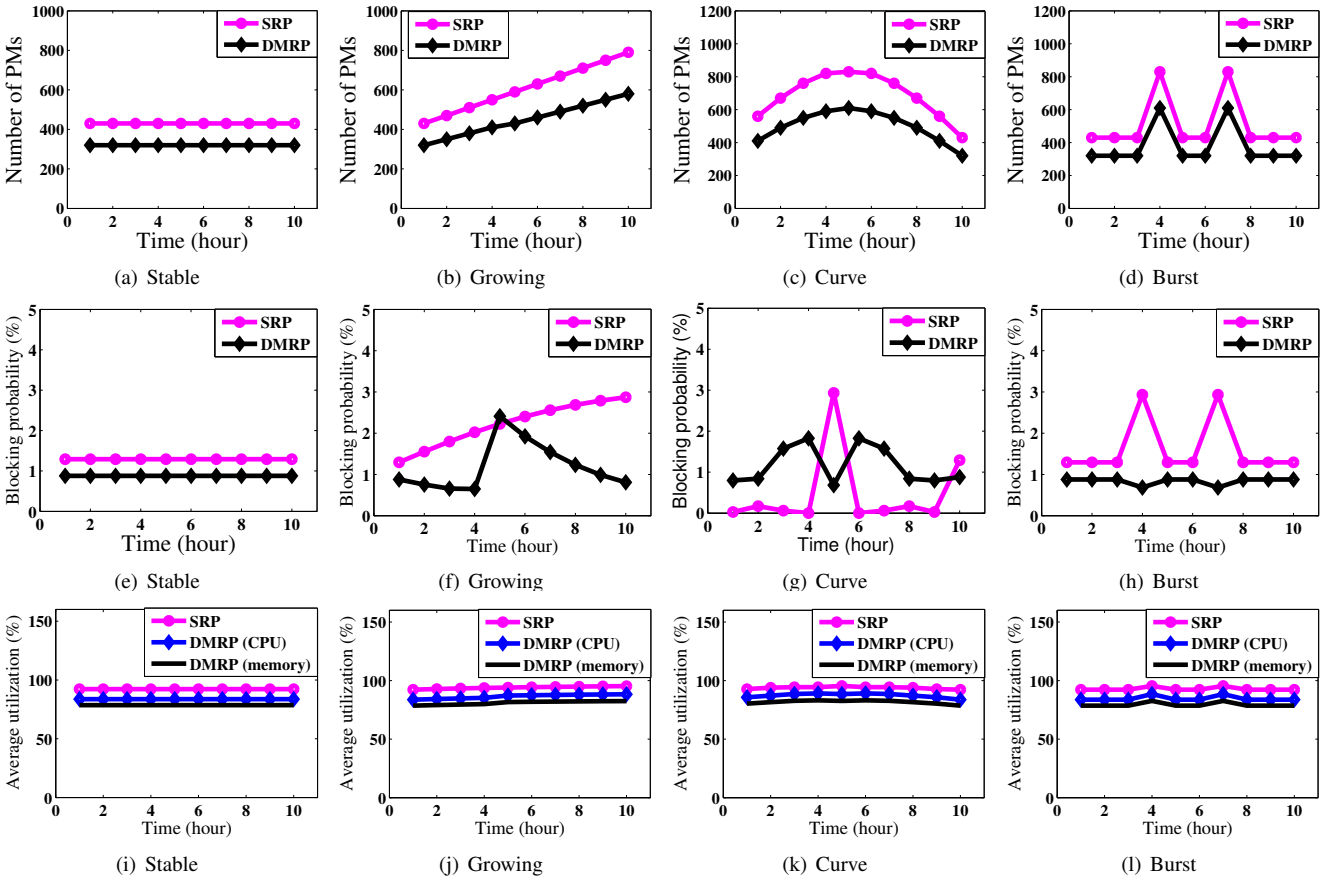

Fig. 5. The detailed results of three metrics with four workload patterns.

IV. Preliminary Evaluation

In this section, we present our preliminary results on demonstrating the advantage of DMRP.

A. Experiment settings

We evaluate the effectiveness of our approach with synthetic workloads. Computer simulations are also used in our experiments for comparison with model. To perform an in-depth study of the advantages of DMRP over SRP scheme, we focus two most important resource types, CPU cores and memory (in GB), in DMRP ($K = 2$). For comparison, we also implement a SRP scheme with one dimensional ($K = 1$) Markov Chain.

In our simulations, we consider a data center with N PMs each of which has 32 cores and 64 GB of memory (by default, $N = 1,000$). SRP scheme uses fixed VM types of $[< 1, 1 >, < 1, 2 >, < 2, 4 >, < 4, 8 >, < 8, 16 >]$ (the first number is the number of CPU cores, and the second number represents the amount of main memory in GB). In DMRP, the system uses flexible combinations from CPU core options $[1, 2, 4, 8]$ and memory amount options $[1, 2, 4, 8, 16]$. Thus, there are 20 VM types in DMRP.

We consider four basic synthetic workload patterns: Stable, Growing, Curve and Burst, as shown in Fig. 3. The peak loads are capped at 600 PMs and the lowest loads are 300 PMs. In each pattern, the workloads are uniformly distributed into 20 kinds of requests, each of which matching one distinct VM type of DMRP. The inter-arrival time of each request type follows Poisson distribution.

The default time slot and P_B threshold are one hour and 3%, respectively. The sensitivities of these two parameters are investigated in the experiments. We study three metrics for performance of our proposed provisioning approach: Mean number of PMs per time slot, utilization and blocking probability.

B. Overall Comparisons

We implement a simulation of DMRP and run the same workloads. By comparison, the number of machines provisioned by DMRP in each time slot is with average error rate less than 3%. Fig. 4 shows the overall results on the average number of PMs provisioned for each workload. The results reveal that DMRP outperforms SRP, with 18-35% reduction on the number of PMs. Thus, DMRP can efficiently reduce the cost of cloud providers.

After presenting the overall comparison, we conduct detailed studies to investigate the temporal behavior. Fig. 5 shows the detailed comparison on the three metrics for resource provisioning between SRP and DMRP.

Through the detailed studies, we find that DMRP always

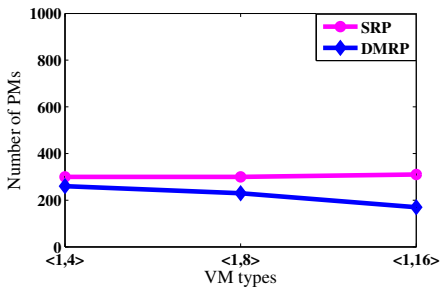

Fig. 6. Provisioned number of PMs under different request distributions.

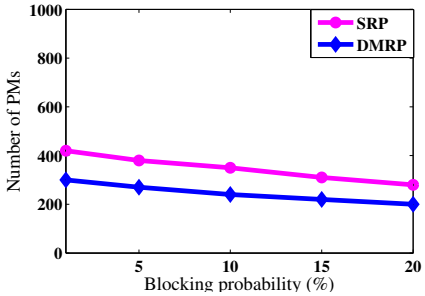

Fig. 7. Provisioned number of PMs under different blocking probability thresholds α.

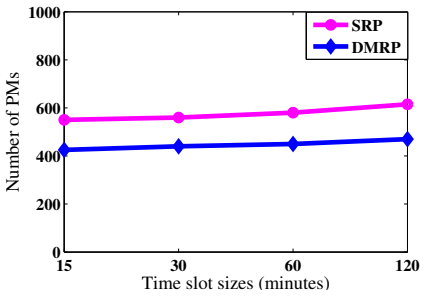

Fig. 8. Provisioned number of PMs under different time slot sizes (minute).

outperforms SRP, with a smaller number of PMs and with similar blocking probabilities and utilizations. Similar P_B values mean that DMRP reduces the number of provisioned PMs with no significant sacrifice on quality of service. Although there are some abnormal points in Fig. 5(f) and Fig. 5(g) caused by different workload variations, they are still in our P_B threshold 3%. The optimizations of provisioned PMs are mainly due to the high resource utilization by eliminating unnecessary resource waste. Moreover, the improvement of DMRP over SRP is consistent under diverse workloads.

Runtime overhead. The execution time of running our model is 10 minutes for each time slot on the data center with 1000 PMs. The simulation runs on a workstation with a six-core CPU and 16 GB of DRAM. Thus, the runtime overhead is acceptable in practice.

C. Sensitivity studies

We now perform sensitivity studies on major parameters for our model, including the arrival rate distribution, P_B threshold and time slot size. For each experiment, we study the impact of varying one parameter while setting other parameters to their default settings.

Fig. 6 shows the results for different distributions of the VM requests in Stable workload. We design a workload with the VM requests which are require more memory than CPU cores. The difference between CPU cores and amount of memory are increasing. The results illustrate that DMRP can reduce the resource amount in the case where workload is diverse in different resources. Specifically, the effectiveness become more obvious when the resource requirements are more unbalanced among different resource types.

Fig. 7 shows the results for varying the threshold of P_B (α) under Stable workload. We can see from the figure that the number of active PMs in each time slot reduces as α increase. This is due to the fact that lower α blocks more requests and hurts the quality of service. In practice, cloud providers can set a proper α to achieve a good tradeoff on the quality of service and the number of active PMs.

Fig. 8 shows the results for varying slot size from 15 minutes to 120 minutes using Growing workload. Our model allows cloud providers to specify time slot according to their requirements. As shown in the figure, the number of active PMs can be further optimized with smaller time slots. This results suggest that we can obtain better optimization effect if we run our model and provision data center more frequently. However, the model computation overhead prohibits a time slot being too small.

V. Conclusion

This paper studied the multiple resource provisioning problem for IaaS cloud providers. To overcome the shortcomings brought by single resource provisioning, we proposed a Markov Chain model based provisioning approach. The goal was to provision a minimal number of PMs while satisfying the predefined threshold on blocking probability. Particularly, we designed an dynamic multiple resource provisioning approach with a multiple dimensional state Markov Chain model. We evaluated the effectiveness of the model with simulations under four synthetic workload patterns. Preliminary experimental studies showed that our model could reduce the number of PMs by 27% compared with the single resource provisioning scheme. We are interested in extending our resource management techniques for monetary cost optimizations for more complicated workloads [10].

References

[1] L. A. Barroso and U. Hölzle. The case for energy-proportional computing. *Computer*.
[2] T. J. Hacker and K. Mahadik. Flexible resource allocation for reliable virtual cluster computing systems. In *Proc. of SC'11*, 2011.
[3] J. Hamilton. http://perspectives.mvdirona.com/2008/11/28/CostOfPowerInLargeScaleDataCenters.aspx.
[4] D. Huang, B. He, and C. Miao. A survey of resource management in multi-tier web applications. *IEEE Communications Surveys Tutorials*, 2013.
[5] M. Lin, A. Wierman, L. L. H. Andrew, and E. Thereska. Dynamic right-sizing for power-proportional data centers. In *INFOCOM'11*, 2011.
[6] M. Mao and M. Humphrey. Auto-scaling to minimize cost and meet application deadlines in cloud workflows. In *Proc. of SC'11*, 2011.
[7] K. Mills, J. Filliben, and C. Dabrowski. Comparing vm-placement algorithms for on-demand clouds. In *Proc. of CLOUDCOM'11*, 2011.
[8] Z. Xiao, W. Song, and Q. Chen. Dynamic resource allocation using virtual machines for cloud computing environment. *IEEE Transactions on Parallel and Distributed Systems*, 2013.
[9] D. Xie, N. Ding, Y. C. Hu, and R. Kompella. The only constant is change: incorporating time-varying network reservations in data centers. *ACM SIGCOMM Computer Communication Review*, 42(4):199–210, 2012.
[10] A. C. Zhou and B. He. Transformation-based monetary cost optimizations for workflows in the cloud. *IEEE Transactions on Cloud Computing*, 2014.