VM Allocation in Data Center Subject to CPU Percentile Constraints

Yue Han
College of Information and Communication
National University of Defense Technology
Xi'an, China
yueh2000_8@163.com

Harry Perros
Computer Science Department
NC State University
Raleigh, USA
hp@csc.ncsu.edu

Mingwu Yao
ISN State Key Lab
Xidian University
Xi'an, China
mwyao@IEEE.org

***Abstract*—At existing cloud computing environment, virtual machine allocation mechanisms have been actively applied to resources management for processing dynamic workload in large-scale data center. In this paper, we address the problem of how many virtual machines should be allocated to a server, so that a given percentile of the execution time of a job is bounded by a predefined value. We first consider the case of "dedicated CPU", whereby we calculate the CPU allocated to a single VM. Subsequently, we extend the analysis to the case where the CPU is shared equally by different groups of VMs. In this case, we calculate how many VMs need to be allocated in each group.**

***Index Terms*—Virtual machines allocation, Data Center, Percentiles, Phase-type distribution**

I. Introduction

Allocating VMs in a data center is a multi-objective problem that includes factors such as operation costs, revenues and performance metrics such as resource utilization, availability and response time. It is necessary for various vendors to guarantee high VMs utilization in long-term job execution period. In recent years, as the the advent of NFV and SDN, it is getting important to study VMs allocation solutions for job execution management in large scale data center network. For example, the works in [1] can be designed as a building block of a SDN orchestrator. VMs allocation can dynamically adjust the number of VMs during execution process and the notable examples are auto-scaling methods [2].

Although the auto-scaling methods are practicable and suitable for VMs allocation, it is difficult for them to know about resource requirements in advance from variable user workloads. To address energy consumption issue in cloud data centers, most of existing solutions save cost at the expense of performance degradation, In [3], authors present VM allocation algorithms, which leverages the performance-to-power ratios and in [4], two semi-Markov decision process (SMDP)-based VMs allocation methods are proposed to balance the tradeoff between the services cost and the computing capacity.

In this paper, we consider the problem of VM allocation to a single core server so that a given percentile of job execution time on a VM is statistically bounded for the optimal balance between resource utilization and work performance. The job execution time on a VM depends on the resources allocated to the VM by hypervisor tools, mainly, CPU and memory, but also disk storage, LAN and WAN bandwidth. The works in [5] solves the multi objective VMs allocation problem . Specifically, we study the VM allocation problem only under the CPU constraint. We assume that a VM executes jobs as the result of user requests. This is a typical scenario in a SaaS environment, whereby the service in a VM is accessed by users over the internet.

The paper is organized as follows. An overview of related works is introduced in Section 2. In Section 3 we use real workload traces to design a Markov model, and then model the job execution time in a VM as a phase-type distribution. In Section 4, we make the 95^{th} percentile of job execution time in a VM as a function of the CPU (we can obtain any percentile, but for presentation purpose we use the 95^{th} percentile). In Section 5, we calculate the number of VMs that should be allocated to a single core, so that a given 95^{th} percentile is satisfied. Finally, Section 6 gives the conclusions.

II. Related work

Cloud data centers allow dynamic and flexible resource provisioning to accommodate time varying multi-user demands. Jing V. Wang et al [6], propose a VM allocation mechanism based on stable matching. It may reduce the overall energy consumption of a cloud data center while maintaining high QoS. Xinqian Zhang et al [7], propose a novel and effective evolutionary approach for VM allocation that can maximize the energy efficiency of a cloud data center. Ali Pahlevan et al [8], propose a heuristicanda machine learning (ML)-based VM allocation method. This approach improve the flexibility and applicability of VM allocation in large-scale DC scenarios.

VM allocation has been extensively studied in the area of horizontal and vertical auto-scaling under a variety of different assumptions, whereby the VMs resources can be scaled up or down (vertical scaling), or the VMs number can be increased or decreased (horizontal scaling). In this area, predicting VMs demand is a specific topic in the case of a delay in making requested resources available[9] [10]. A reasonable assumption is that a VM is assumed to have a fixed required CPU known in advance. Under this assumption, optimal VMs number can be calculated to improve CPU resource utilization and thus reduce the operating data center consumption.

Lots of VMs allocation mechanism in cloud data centers have been proposed. In the literatures, it is typically assumed that the CPU is single core, or it is treated as a number of single cores if it is multi-core. Bo Yang et al [11], proposed a combinatorial auction-based mechanism to address VM allocation in clouds in the presence of multiple types of VMs. Nejad et al [12], proposed the auction-based greedy mechanisms for VM provisioning and allocation. Finally, a comprehensive survey on the problem of allocating VMs to a data center can be found in [13].

III. A PHASE-TYPE MODEL OF EXECUTION TIME

Our work is based on real workload traces of 644 VMs that were collected from a production environment by an IaaS provider, which is not aware of the application configuration inside a virtual machine, and the data was collected at the VM level using the VMware ESXi hypervisor. The traces give the average CPU usage in MHz on a per 5-minute basis of each of the 644 VMs that ran for a month. Each virtual machine in the trace ran on a single 2 GHz CPU. Other statistics were also collected, but they are not used in this paper.

A. Cluster Analysis

We run a cluster analysis on the data points of all VMs by *k*-means algorithm in Matlab in order to identify the number of CPU clusters. It is noted that the set of observations in each cluster does not belong to the same VMs. That is, we do not clustered the VMs themselves, but rather their CPU usage. Thus, the same VMs may appear in all clusters. After experimenting with different values, we determine *k=3*, its cluster means of CPU are 180, 935 and 607MHz, respectively.

TABLE I
TRANSITION PROBABILITIES p_{ij}

Cluster	1	2	3
1	0.999	0.0006	0.0005
2	1	0	0
3	1	0	0

B. Markov Process Model

A virtual machine during its life typically switches between different clusters. Going through all the data we calculated the number of times a VM in cluster *i* switches to cluster *j* in the next minute. After normalization, we obtain the transition probabilities p_{ij} shown in table 1. We model the VM evolution by a continuous-time Markov process with three states, one per cluster, with the transition probabilities as shown in table 1 and services rates: μ_1 =1/180, μ_2=1/935, and μ_3=1/607. The stationary probability π_i shows that the VM is in steady state *i*, where *i*=1, 2, 3, can be obtained numerically by solving the Markov process with a rate matrix *R*:

$$R = \begin{pmatrix} -u_1 & u_1p_{12} & u_1p_{13} \\ u_2p_{21} & -u_2 & u_2p_{23} \\ u_3p_{31} & u_3p_{32} & u_3 \end{pmatrix} \quad (1)$$

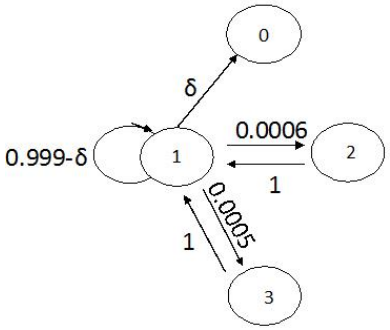

Fig. 1. The phase-type distribution of the execution time

C. Phase-type Distribution

The execution time can be modelled by a phase-type distribution (α, T) with four states, namely the above transient states 1, 2, 3 and an absorbing state, state 0. The initial probability vector α gives the probability that the job execution starts in state *i*, *i*=0,1,2,3, and *T* is the 3×3 transition matrix which is part of the generator matrix *Q*, where

$$Q = \begin{pmatrix} T & T^0 \\ 0 & 0 \end{pmatrix} \quad (2)$$

or

$$Q = \left(\begin{array}{ccc|c} -u_1 & u_1p_{12} & u_1p_{13} & u_1p_{10} \\ u_2p_{21} & -u_2 & u_2p_{23} & u_2p_{20} \\ u_3p_{31} & u_3p_{32} & -u_3 & u_3p_{30} \\ \hline 0 & 0 & 0 & 0 \end{array}\right) \quad (3)$$

with *T* and T^0 being the left and right upper blocks respectively. We assume that the distribution of a job execution time is the distribution of absorption time. That is, a job starts from one of three states with probability α and eventually it is terminated when the process gets absorbed. We may generate different distributions by varying the absorption probabilities. In this paper, we assume that the absorption state is only reachable from transient state 1 as shown in figure 1. The higher the absorption rate δ, the shorter the execution time.

IV. THE 95^{th} PERCENTILE OF THE EXECUTION TIME UNDER A CPU CONSTRAINT

In this section, we will obtain the 95^{th} percentile of the a job execution time on a VM, assuming that the VM has been allocated *B* MHz. Let *X* be a random variable that has a phase-type distribution (α, T). The cumulative distribution *F(x)* is given by the expression: $F(x) = 1 - \alpha e^{Tx} e$, where $e = (1, 1...1)^T$ and *F(x)* = 0.95. We will use Matlab to obtain *x* numerically by successive approximations.

With the phase-type model, *B* has no impact on the execution time as long as it is larger than the service rates μ_i, *i*=1, 2, 3. If μ_i is greater than *B*, then the execution time will get longer. We set *B* is the largest service rate μ_1 and calculate the execution time. Figure 2 gives the plot of the 95^{th} percentile of the execution time as a function of *B* with three absorption rates, δ= 0.1, 0.5, 0.9.

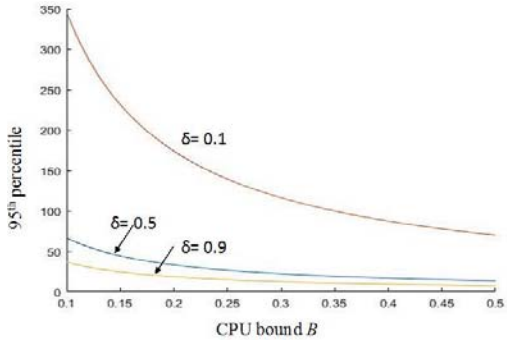

Fig. 2. The 95^{th} percentile of the execution time with the CPU bound *B*

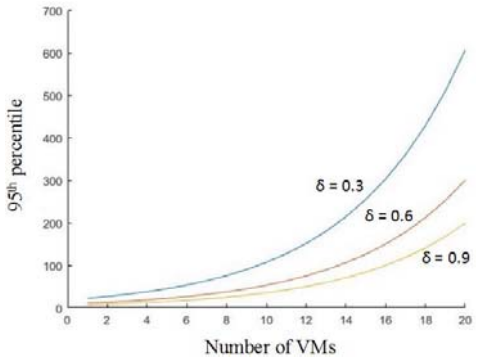

Fig. 3. 95^{th} percentile of the execution time with the number of VMs

VMs allocation to a single CPU can be seen as a one dimensional bin-packing problem. The total CPU requirement of all the VMs allocated to the same server should be less or equal to the server's CPU. The solution can be extended to the case where a VM executes a mixture of loads, each represented by a different phase-type distribution. The VM requirements can be determined by constructing an aggregation of the execution time as described in the following section.

V. VM ALLOCATION SUBJECT TO A CPU PERCENTILE CONSTRAINT

In previous section, the bound *B* is used exclusively by the VM, and it is not shared with other VMs. A more efficient CPU usage is to allow a number of VMs to share the same CPU. In this section, we determine the number of VMs to allocate on the same CPU. If the total required CPU by all the VMs is less than the CPU capacity *B*, then all the VMs can run without any CPU restrictions. However, if it exceeds *B*, we assume that the CPU allocated to each VM is reduced proportionally, so that the total required CPU is equal to *B*.

For example, let us consider two VMs, referred to as VM1 and VM2, each represented by a three-state Markov process. Let μ_1, μ_2, and μ_3 be the service rates in states 1, 2, and 3. To simplify operations, we assume that the two VMs are identical, but this method can be applied to heterogeneous VMs. Let π_i be the stationary probability that a VM is in state *i*. We note that the two VMs are independent of each other. Therefore, the stationary probability π_{ij} that VM1 is in state *i* and VM2 in state *j* is $\pi_i\pi_j$. Then, we combine the two Markov process into a single one with nine states, as shown in table 2.

TABLE II
STATES AND SERVICE RATES OF THE COMBINED MARKOV PROCESS

States	state of each VM	Service rate	Adjusted Service rate
1	(1, 1)	$2\mu_1$	$2\mu_1$
2	(1, 2)	$\mu_1+\mu_2$	$\mu_1+\mu_2$
3	(1, 3)	$\mu_1+\mu_3$	*B*
4	(2, 1)	$\mu_1+\mu_2$	$\mu_1+\mu_2$
5	(2, 2)	$2\mu_2$	$2\mu_2$
6	(2, 3)	$\mu_2+\mu_3$	$\mu_2+\mu_3$
7	(3, 1)	$\mu_1+\mu_3$	*B*
8	(3, 2)	$\mu_2+\mu_3$	$\mu_2+\mu_3$
9	(3, 3)	$2\mu_3$	$2\mu_3$

We still identify the states of the combined Markov process which exceed *B*, and replace their rates by *B*. For instance, let us assume that states 3 and 7 exceed *B*. Then, as can be seen in table 2, their corresponding service rates are changed to *B*. The job execution time is modelled by a phase-type distribution with 10 states, 9 of which are the 9 states of the combined Markov process, and the 10^{th} state being the absorbing state. The 95^{th} percentile is obtained as before, i.e., calculating *x* by successive approximations so that *F(x)* = 0.95, where $F(x) = 1 - \alpha e^{Tx} e$, and *T* is the upper left-hand matrix of the rate matrix *Q* representing the new phase-type distribution.

For *n* ($n \geq 2$) identical VMs, we calculate their joint state probability using the multinomial distribution. For instance, for the three-state VMs, the probability that k_i VMs are in state *i*, *i*=1, 2, 3, where $n = k_1 + k_2 + k_3$, is:

$$\binom{n}{k_1}\binom{n-k_1}{k_2}\binom{n-k_1-k_2}{k_3}\pi_{1k_1}\pi_{2k_2}\pi_{3k_3} \quad (4)$$

Subsequently, we can obtain the 95^{th} percentile of the execution time as above, where *Q* is the rate matrix based on the combined *n* Markov processes. Figure 3 shows the 95^{th} percentile of the execution time, with δ = 0.3, 0.6, 0.9 as a function of *n*. As expected, for a given δ, it increases as *n* increases. Furthermore, for a given number of VMs, the 95^{th} percentile increases as the service time gets longer (i.e., δ gets smaller). This is because longer service times have higher 95^{th} percentiles than that of shorter service times.

Now we consider the heterogeneous VMs case. Assume that there exist *G* VM groups, each associated with a different Markov process. The execution time of group *g*, *g* =1, 2... *G* is modelled by a phase-type distribution. We use the same approach to obtain the 95^{th} percentile of the execution time. First the aggregated Markov process is constructed. Let s_g be the total number of the aggregated states and π_{ig} be its stationary probability, g=1,2,...,s_g. The probability in a state is the product of all the individual probabilities. Finally, we can obtain the 95^{th} percentile of the execution time based on the aggregated Markov processes. Using this method, we can search for different combinations of VMs in each group for which the percentile of the execution time is as close as possible to a pre-determined value.

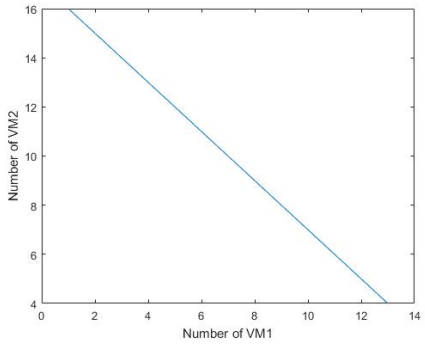

Fig. 4. Optimum combinations of VMs from two groups(2 GHz CPU)

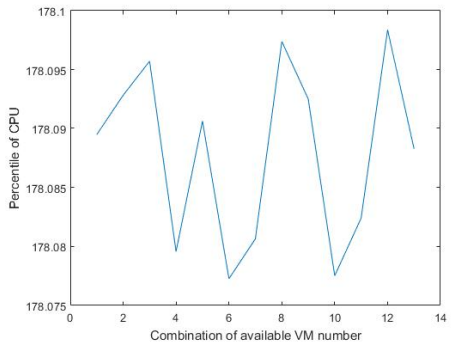

Fig. 5. 95^{th} percentile of the optimum combinations in Fig.5

Figures 4 give results of two groups (groups 1 and 2) of VMs for a 2 GHz CPU assuming no RAM bottlenecks. We first calculate the maximum number of group 1 VMs that can be allocated without violating the percentile constraint, likewise, for group 2. Seen in figure 4, a maximum of 13 group 1 VMs and 16 group 2 VMs can be allocated. The straight line gives the set of the optimum combinations (n_1, n_2) and implies that a group 1 VM is interchangeable with a group 2 VM as far as achieving the target percentile. For instance, (1, 16) and (2, 15) are both optimum combinations.

Figure 5 gives the actual percentile value for each of the optimum combinations. For instance, we have a percentile value of 179.1 for the case of (1, 16). Seen in figure 5, none of these combinations have a 95^{th} percentile close to the percentile constraint, and so any of these combinations can be used to be the optimum combinations, though the combination (12,5) seems to be the best by an extremely small margin. It is worth noting that the set of optimum combinations may not always lie on the same line as the input parameters change.

Since the numerical results were obtained from the collected data traces, we have not chosen other related work to be compared with the proposed method. However, it is extensively applicable to any Markovian model based application. As for the time complexity, it depends on the number of aggregated VMs, i.e., the order O(i, j, k), where i, j and k is the number of group 1, 2, and 3 VMs respectively. As for the computation complexity, if the numbers of feasible VMs increase dramatically, larger Markovian representations may lead to unmanageable and thus the computation is infeasible. This problem can be resolved by collapsing such distributions to the mixture of exponential distributions with a small number of states using the moments method. In this case, an approximate search method may be usful in an online environment to reduce the computational complexity.

VI. CONCLUSION AND FUTURE WORK

We addressed VM allocation in a data center subject to a given execution time. We model it by a phase-type distribution, whose parameters were estimated from real traces. We calculate the dedicated CPU allocated to a VM so that the 95^{th} percentile of the execution time is satisfied. Subsequently, we extended this work to the case of allocating different VMs groups. This work can be extended to calculate the individual group percentile rather than that for all groups. Another extension is to use different CPU weights for the different VMs groups.

ACKNOWLEDGEMENT

This work was supported by the National Natural Science Foundation of China (NSFC) under Grant No. 61671353.

REFERENCES

[1] Giuseppe Portaluri, Davide Adami, Andrea Gabbrielli, etc., "Power Consumption-Aware Virtual Machine Allocation in Cloud Data Center," IEEE GLOBECOM, 2016.

[2] Michael Tighe, Michael Bauer, "Topology and Application Aware Dynamic VM Management in the Cloud," JOURNAL OF GRID COMPUTING ,vol. 15, no. 2, pp 273-294, 2017.

[3] Xiaojun Ruan, Haiquan Chen, Yun Tian, Shu Yin, "Virtual machine allocation and migration based on performance-to-power ratio in energy-efficient clouds," Future Generation Computer Systems-The International Journal of eScience, vol. 100, pp 380-394, 2019.

[4] Qizhen Li, Lianwen Zhao, Jie Gao, "SMDP-Based Coordinated Virtual Machine Allocations in Cloud-Fog Computing Systems," IEEE Internet of Things Journal , vol. 5, no. 3, pp 1977-1988, 2018.

[5] Neeraj Kumar Sharma and G. Ram Mohana Reddy, "Multi-Objective Energy Efficient Virtual Machines Allocation at the Cloud Data Center," IEEE Trans. on Services Computing, vol. 12, no. 1, pp 158-171, 2019.

[6] Jing V. Wang, Kai-Yin Fok, Chi-Tsun Cheng, and Chi K. Tse, "A Stable Matching-Based Virtual Machine Allocation Mechanism for Cloud Data Centers," IEEE World Congress on Services Computing, 2016.

[7] Xinqian Zhang, Tingming Wu, Mingsong Chen, etc.,"Energy-aware virtual machine allocation for cloud with resource reservation," Journal of Systems and Software, vol. 147, pp 147-161, 2019.

[8] Pahlevan, Ali, Qu, Xiaoyu, Zapater, Marina,"Integrating Heuristic and Machine-Learning Methods for Efficient Virtual Machine Allocation in Data Centers," IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, vol. 37, no. 8, pp 1667-1680, 2018.

[9] B. Bouterse and H. Perros, "Dynamic VM Allocation in a SaaS Environment," Annals of Telecommunications, vol. 73, no. 3-4, pp. 205-218, 2018.

[10] B. Bouterse and H. Perros, "Performance analysis of the reserve capacity policy for dynamic VM allocation in a SaaS environment," Simulation Modelling Practice and Theory, vol. 93, pp 293-304, 2019.

[11] Bo Yang, Q. Chen, Shilong Jiang and Keqin Li, "Envy-free auction mechanism for VM pricing and allocation in clouds," Future Generation Computer Systems, vol.86, pp 680-693, 2018.

[12] M.N. Nejad, L. Mashayekhy, and D. Grosu, "Truthful Greedy Mechanisms for Dynamic Virtual Machine Provisioning and Allocation in Clouds," IEEE Tran Parallel and Distributed Systems, vol. 26, no. 2, pp 594-603, 2015.

[13] Z. Mann, "Allocation of Virtual Machines in Cloud Data Centers-A Survey of Problem Models and Optimization Algorithms", ACM Computing Surveys, vol. 48, no. 1, 2015.