https://doi.org/10.5392/IJoC.2017.13.2.029

An Efficient VM-Level Scaling Scheme in an IaaS Cloud Computing System: A Queueing Theory Approach

Doo Ho Lee
Department of Industrial & Management Engineering
Kangwon National University, Samcheok 25913, Kangwon, Republic of Korea

ABSTRACT

Cloud computing is becoming an effective and efficient way of computing resources and computing service integration. Through centralized management of resources and services, cloud computing delivers hosted services over the internet, such that access to shared hardware, software, applications, information, and all resources is elastically provided to the consumer on-demand. The main enabling technology for cloud computing is virtualization. Virtualization software creates a temporarily simulated or extended version of computing and network resources. The objectives of virtualization are as follows: first, to fully utilize the shared resources by applying partitioning and time-sharing; second, to centralize resource management; third, to enhance cloud data center agility and provide the required scalability and elasticity for on-demand capabilities; fourth, to improve testing and running software diagnostics on different operating platforms; and fifth, to improve the portability of applications and workload migration capabilities. One of the key features of cloud computing is elasticity. It enables users to create and remove virtual computing resources dynamically according to the changing demand, but it is not easy to make a decision regarding the right amount of resources. Indeed, proper provisioning of the resources to applications is an important issue in IaaS cloud computing. Most web applications encounter large and fluctuating task requests. In predictable situations, the resources can be provisioned in advance through capacity planning techniques. But in case of unplanned and spike requests, it would be desirable to automatically scale the resources, called auto-scaling, which adjusts the resources allocated to applications based on its need at any given time. This would free the user from the burden of deciding how many resources are necessary each time. In this work, we propose an analytical and efficient VM-level scaling scheme by modeling each VM in a data center as an M/M/1 processor sharing queue. Our proposed VM-level scaling scheme is validated via a numerical experiment.

***Key words**: VM-level Scaling, IaaS Cloud Computing System, M/M/1 Queue, Processor-sharing.*

1. INTRODUCTION

The attention to user on-demand cloud services has spurred the migration of increasing numbers of applications to the cloud [1]. One of the attractive features of cloud services to application providers is the ability to access computing resources such as shared hardware, software, application, and information elastically according to dynamic resources demands. A cloud computing system provides a dynamic pool of virtualized computing resources and offers an elastic scheme matching the user demands, so that allocated computing resources can be scaled-up or scaled-down on a pay-as-you-go basis [2].

In an Infrastructure-as-a-Service (IaaS) cloud, users can launch their virtual machines (VMs) with required computing resources, and an IaaS cloud is able to control and maintain several VMs on physical machines (PMs) with remarkable flexibility [3]-[5]. One of the main goals in IaaS cloud computing system is to allocate computing resources that are truly needed without violating service level agreements (SLAs). SLAs describe all aspects of cloud service usage and obligations between users and cloud service providers including the price for cloud services, Quality-of-Service levels required while the services are provisioned, and penalties with regard to SLA violations. To achieve such a goal, IaaS cloud providers and third party cloud services offer rule-based (or schedule-based) scaling policies to help users automatically scale-up and scale-down resources, called auto-scaling. Decisions of scaling-up or scaling-down are made according to the last values of monitored variables. Amazon AWS Auto-Scaling [6] and some cloud service brokers such as Rightscale [7] and Dell Cloud Manager [8] offer rule-based auto-scaling schemes to allow users to add and remove resource at a given time. Most of the schemes are based on resource utilization, such as "Create 5 VMs if the average CPU utilization rate exceeds 70% over the past 10 minutes." In this rule, by the way, users may wonder whether 5 VMs are enough or not. In other words, VMs under-provisioning will inevitably harm performance and cause SLAs violations, while VMs over-

* *Corresponding author, Email: enjdhlee@kangwon.ac.kr*
Manuscript received Mar. 09, 2017; revised Apr. 25, 2017; accepted Apr. 25, 2017

provisioning will lead to resources idle and cost waste. Therefore, the main objective of an auto-scaling is to decide the proper number of VMs provisioned in a PM without user's intervention while satisfying SLAs.

In this work, we propose a novel VM-level scaling scheme by modeling a VM as a processor-sharing queue. Note that in practical cloud computing system, not only task request arrivals but also task executions are random, which means cloud computing system should adapt to uncertainties such as variations of both task arrivals and task executions. It is obvious that these uncertainties should have a great impact on resource performance. This motivates us to investigate stochastic cloud computing system in order to develop an efficient auto-scaling scheme. The proposed scheme decides the proper number of VM instances in a PM while satisfying the SLA related to the response time and VM utilization thresholds.

2. QUEUEING MODEL

In this section, we model a VM as a processor sharing (PS) queue. Tasks arrive at a PM following the Poisson process at a rate λ. While the number of potential clients is high, each client typically submits a task at a time with low probability. Therefore, the task arrival can be adequately modeled as a Poisson process. We assume that the number c of identical VMs are currently provisioned in a PM and an arriving task is assigned to one of the VMs in a round-robin manner. Not only Amazon AWS [6] but also other IaaS cloud services practically adopt a round-robin task assignment policy for easy load balancing of PMs. Due to the decomposition property of a Poisson process, the task arrival process in a VM also follows a Poisson process at a rate λ / c (see Fig. 1). The task service time is assumed to have the exponential distribution with a rate of μ. Unfortunately, the assumption of exponential service time is not as general as the task arrival time distribution. But it could still be a reasonable assumption when no other data is available about service times. The task process duration define the service time for utilization of a VM. Let us see if the task process duration can be assumed to be exponentially distributed. A single VM takes a very small fraction of the resources of the IaaS cloud data center. Furthermore, decisions on how long to use a VM are independently made by each task. From these phenomena, it appears that exponential service times are a good fit. Intuitively, the probability of a task making a very long process duration is very small. There is a high probability that a task process duration will be short. This matches with the observation that service times of a VM can follows exponential distributions. Let ρ denote the utilization rate (or offered load) of a VM, i.e. $\rho = (\lambda / c)\mu^{-1}$. Throughout this work, we assume $\rho < 1$ for each VM to be stable.

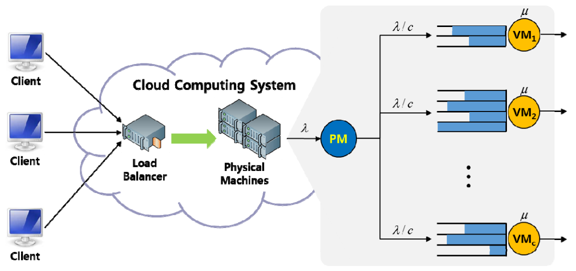

Fig. 1. The architecture of IaaS cloud computing system

Consider an M/M/1 queue where the queue discipline is processor-sharing (time-sharing). This discipline implies if there are already $n-1$ tasks in a VM, then an arriving task as well as the other (waiting) tasks in a VM all start receiving service immediately at the average rate of μ / n. There is no queue (waiting space) as such, and the service rate at which tasks receive service changes each time a new task joins a VM and each time a task whose service requirements is fully met departs from a VM. It is highly reasonable to model a VM as an M/M/1 processor sharing queue since VM instances are generally equipped with a multiple-core virtual central processing unit (CPU) and the hyper-threads technology. It should be noted that the probability distribution of the number of tasks in the M/M/1 processor sharing queue, namely M/M/1-PS queue, is stochastically equivalent to that of the ordinary M/M/1 queue due to the memoryless property of the exponential task service time distribution. Let p_n denote the probability that there exist n tasks in a VM. Then, p_n is given by

$$p_n = (1-\rho)\rho^n, \quad n = 0, 1, \cdots . \tag{1}$$

Next, we investigate the response time distribution. The response time is defined as the time length between a task arrival epoch and at task departure epoch. Among lots of previous works finding the response time distribution of various PS queues, we introduce Masuyama and Takine [9]'s result, which is relatively simple and gives the explicit form of the response time distribution. A randomly chosen task who finds n tasks in a VM upon arrival is called T_n. Let W_n denote a random variable representing the response time of task T_n. We define $w_n(x) = \Pr\{W_n > x\}$, for $x \ge 0$, and $n = 0, 1, \cdots$. It is easy to see that $w_n(0) = 1$ and $w_n(x)$ satisfies the following differential-difference equations:

$$\frac{d}{dx}\mathbf{w}(x) = \mathbf{P} \cdot \mathbf{w}(x) , \tag{2}$$

where $\mathbf{w}(x) = \left(w_0(x), w_1(x), w_2(x), \cdots\right)^{\bullet}$ and $\mathbf{P}$ is defined as

$$\mathbf{P} = \begin{bmatrix} -\lambda/c-\mu & \lambda/c & 0 & 0 & \cdots \\ \mu/2 & -\lambda/c-\mu & \lambda/c & 0 & \cdots \\ 0 & 2\mu/3 & -\lambda/c-\mu & \lambda/c & \cdots \\ 0 & 0 & 3\mu/4 & -\lambda/c-\mu & \cdots \\ \vdots & \vdots & \vdots & \vdots & \ddots \end{bmatrix}.$$

The solution of (2) is generally given by $\mathbf{w}(x) = \exp(\mathbf{P}x)\mathbf{e}$, where $\mathbf{e} = (1,\ 1,\ 1,\ \cdots)^{\bullet}$. By making use of the uniformization technique [10], we obtain

$$w_n(x) = \sum_{k=0}^{\infty} \frac{(\lambda/c+\mu)^k x^k e^{-(\lambda/c+\mu)x}}{k!} h_{n,k}. \qquad (3)$$

The quantities $h_{n,k}$ in (3) satisfy the following recurrence relations:

$$h_{n,k+1} = \frac{n}{n+1}\frac{1}{1+\rho} h_{n-1,k} + \frac{\rho}{1+\rho} h_{n+1,k}, \qquad (4)$$

where the boundary conditions are given by $h_{n,0} = 1$ and $h_{-1,k} = 0$. Let $W(x)$ denote the probability that the response time is longer than *x*. By the total probability theorem, $W(x)$ is expressed as

$$\begin{aligned} W(x) &= \Pr\{\text{response time} > x\} = \sum_{n=0}^{\infty} p_n w_n(x) \\ &= \sum_{n=0}^{\infty} (1-\rho)\rho^n \sum_{k=0}^{\infty} \frac{(\lambda/c+\mu)^k x^k e^{-(\lambda/c+\mu)x}}{k!} h_{n,k}. \end{aligned} \qquad (5)$$

It should be noted that recurrence equations in (4) can be used to figure out the characteristics of the function $W(x)$ in (5). Since these expressions are, however, numerically unstable, various approximation techniques for calculating the response time probability have been developed. Among them, we introduce Guillemin and Boyer [11]'s approximation technique. By utilizing the spectral theory, Guillemin and Boyer [11] presents the following asymptotic distribution function of $W(x)$ for $x >> 0$:

$$W(x) \approx G(x)\exp\left(\frac{B}{D} + \frac{K}{C-\left(1-\sqrt{\rho}\right)^2} - \frac{1}{D}\left(\sqrt{xC_2}+\sqrt{C_1}\right)^2\right), \qquad (6)$$

where

$$A = \left(1-\sqrt{\rho}\right)\left(0.28-2.28\sqrt{\rho}\right),$$

$$B = \frac{1-\sqrt{\rho}}{\sqrt{\rho}}\left(1.42\sqrt{\rho}-0.28\rho-0.14\right),$$

$$C = 0.28\left(1-\sqrt{\rho}\right)^2, \quad D = 0.72\sqrt{\rho}+0.28,$$

$$K = A - \frac{BC}{D}, \quad G(x) = \sqrt{\frac{2\pi(1-\rho)^2}{\rho\left|H(x)\right|}},$$

$$H(x) = -\frac{2C_2^3 x}{D^3}\left(1+\sqrt{\frac{C_1}{C_2 x}}\right)^3 - \frac{2C_1C_2^2}{D^3}\left(1+\sqrt{\frac{C_2 x}{C_1}}\right)^3,$$

$$C_1 = \frac{AD-BC}{C-\left(1-\sqrt{\rho}\right)^2} \text{ and } C_2 = 0.72\left(1-\sqrt{\rho}\right)^2.$$

Numerical simulations conducted in Guillemin and Boyer [11] reveal that the approximation in (6) is very accurate for high values of ρ and is less accurate when ρ decreases, but even for ρ in range of [0.55, 0.7], it still yields a not so bad estimate of $W(x)$. It is reasonable to use the above approximation when computing the response time probability.

3. VM-LEVEL AUTO-SCALING

Our VM-level scaling scheme runs continuously to ensure that scaling goals are met at all times. We establish the following design goals for our VM-level scaling approach:

- Automation: All decisions related to scaling VMs should be made automatically without human intervention.
- Adaptation: The IaaS cloud computing system should adapt to uncertainties such as a variation in task arrivals and task processes.
- SLA compliance: The cloud computing system should comply with SLAs at any given time.
- VM utilization assurance: Utilizations of all VMs in a PM should be within the control range, i.e. between the lower utilization threshold (LUT) and the upper utilization threshold (UUT).

The proposed VM-level scaling scheme assumes all VMs have the same hardware and software configuration, thus they deliver the same performance. It can be achieved in practice with proper configuration and management of VMs either via a hypervisor such as Xen [12] or via high-level virtual environment managers such as OpenNebula [13] and OpenStack [14]. VMs with heterogeneous specifications may also be instantiated in the same PM. In such a case, we have to decide when to scale VMs with different specifications, and this topic is subject of future research.

Most operation control analyses in the cloud computing system should be based on the task arrival pattern. From this point of view, the scaling analysis should also begin with identifying (updating) the task arrival rate. Due to the Poisson task arrivals, the inter-arrival time of between two consecutive tasks follows the exponential distribution with a mean λ^{-1}. Let τ_i denote the i^{th} observation of the task inter-arrival time. The log-likelihood function is given by

$$l(\lambda;\tau_1,\cdots,\tau_n) = n\ln\lambda - \lambda\sum_{i=1}^{n}\tau_i . \tag{7}$$

From (7), the maximum likelihood estimator (MLE) of λ is given by $n\Big/\sum_{i=1}^{n}\tau_i$. The strong law of large number guarantees $n\Big/\sum_{i=1}^{n}\tau_i$ converges to λ as the value of n increases. When we finish updating the estimation of the task arrival rate, we check whether the current pool of VMs complies with SLAs or not. Only SLA considered in this work is the response time, which means "$W(d) < \alpha\%$" is stated in the SLA, for example. To make this SLA verification, we then identify the current task service rate. The MLE of μ is also calculated as $m\Big/\sum_{j=1}^{m}\sigma_j$, where σ_j is the j th observation of the task service time. The estimated task service rate is used to predict the response time of a task, the VM utilization rate, and other performance measures.

We now introduce the VM-level auto scaling scheme based on the M/M/1-PS queueing system in Algorithm 1. Briefly summarizing Algorithm 1, the number of required VMs c is updated relying on the SLA related to response time, VM utilization, the MLEs of task arrival rate and task service rate, and maximum and minimum values of c . If the SLA is violated or VM utilization rate exceeds UUT, c is recalculated according to lines 11-16. In addition, we keep track of both maximum and minimum values of c in order to get the number of VMs that either been tested before or whose value is known to be insufficient depending on previous tested values. It prevents loops in the process. If the VM utilization rate is predicted to be below LUT, the number of VMs is updating according to lines 17-23. The initial maximum number of VMs possible in a PM is contingent on either the policy made by users and their negotiation with IaaS providers. On the other hand, the minimum number of VMs is initially set as 1 and it is updated while Algorithm 1 is executed.

The VM utilization rate in line 9 is predicted as the MLE of a task arrival rate divided by MLE of a task service rate, which interpreted as the average percent of time or the probability that a VM undertakes a task. In the literature, time-series techniques have been applied to VM or resource usage prediction. For example, Huang et al. [15] present a good resource prediction model (for CPU and memory utilization). One may replace our VM utilization prediction method in line 9 with the result in Huang et al. [15] or others; thus, it can enhance the accuracy of the proposed scaling scheme. Finally, Algorithm 1 ends if the calculated value of c does not violate the SLA and the VM utilization rate exists between UUT and LUT. Computation time of Algorithm 1 is governed by the repeat loop between lines 6-24. The number of iterations in the loop depends on finding the proper value of c which satisfies the SLA compliance and VM utilization rate assurance.

Algorithm 1. VM-level scaling

```
Input: task arrival time $\tau_i$, $i = 0, \cdots, n$
Input: task service time $\sigma_j$, $j = 0, \cdots, m$
Input: SLA constraint $d$ and $\alpha$
Input: VM utilization thresholds UUT and LUT
Output: Number of VMs $c$
1:  $\hat{\lambda} \leftarrow n\Big/\sum_{i=1}^{n}\tau_i$;
2:  $\hat{\mu} \leftarrow m\Big/\sum_{j=1}^{m}\sigma_j$;
3:  $c \leftarrow$ number of current VMs;
4:  MAX_VM $\leftarrow$ maximum number of VMs possible in a PM;
5:  MIN_VM $\leftarrow 1$;
6:  repeat
7:      $newc \leftarrow c$;
8:      $\hat{\lambda}_{\text{VM}} \leftarrow \hat{\lambda}/c$;
9:      VM_utilization_rate $\leftarrow \hat{\lambda}_{\text{VM}}/\hat{\mu}$;
10:     SLA_violation_prob $\leftarrow W(d)$; // $W(x)$ in (6)
11:     if (SLA_violation_prob $>$ $\alpha$ || VM_utilization_rate $>$ UUT) then
12:         $c \leftarrow c + c \times 0.05$;
13:         MIN_VM $\leftarrow c \times 0.1$;
14:         if ($c$ $>$ MAX_VM) then
15:             $c \leftarrow$ MAX_VM;
16:         end if
17:     else if (VM_utilization_rate $<$ LUT) then
18:         MAX_VM $\leftarrow c$;
19:         $c \leftarrow$ MIN_VM + (MAX_VM – MIN_VM) $\times 0.5$;
20:         if ($c$ $\leq$ MIN_VM) then
21:             $c \leftarrow$ MIN_VM;
22:         end if
23:     end if
24: until ($c == newc$)
25: return ($\lceil c \rceil$) // $\lceil \cdot \rceil$ denotes a ceiling function
```

4. EXPERIMETAL RESULTS

In this section, we present numerical experiments to validate the proposed VM-level auto-scaling algorithm. We now consider following experiment assumptions: i) 50 VMs are currently provisioned in a PM; ii) A PM can provision at most 200 VMs due to the limited physical resources; iii) The task response time should be less than 20 seconds with probability 99.5% (i.e., the SLA violation probability should be lower than 0.5%); iv) A VM utilization rate should exist between 50% (LUT) and 60% (UUT). With the simulated task inter-arrival time data whose size is 950,237 and the service time data whose size is 886,471, a task arrival rate and a service rate are respectively estimated as $\hat{\lambda} = 47.616$ tasks/sec and $\hat{\mu} = 1.032$ tasks/sec. Under these input values, the SLA violation probability and the VM utilization rate are respectively calculated as 16.74% and 92.29% (see the first row in Table 1).

Applying Algorithm 1 to the scale-up case, it gives the desired value 77.57 of c in 9th iteration; therefore, we are recommended to provision more 28 VM instances in a PM. In case of 78 VMs being provisioned in a PM, the SLA violation probability is expected to be 0.37% which complies with the response time SLA and the VM utilization rate is expected to be 59.16%, which assures the VM utilization rate constraint.

Table 1. Experiment result: Scaling-up case

Iteration	SLA violation probability (%)	VM Utilization rate (%)	Number of VMs
0	16.74357	92.29201	50
1	9.71879	87.89715	52.5
2	6.04034	83.71157	55.125
⋮	⋮	⋮	⋮
8	0.56372	62.46686	73.87277
9	0.38966	59.49225	77.56641

Next, in the scaling-down case, we use the same assumptions as the scaling-up case except for the first assumption: 120 VMs are currently being provisioned in a PM. In our scaling-down case, although the SLA violation probability is lower than 0.05%, the VM utilization rate is lower than LUT (see the first row in Table 2). Applying Algorithm 1, we finally obtain the proper number 77.26 of c; thus, if we remove 42 VM instances, the SLA violation probability is expected to be 0.37% which complies with the response time SLA and the VM utilization rate is expected to be 59.16%, which assures the VM utilization rate constraint. Note that both in the scaling-up case and in the scaling-down case, it is recommended to provision 78 VMs in a PM under the same values of $\hat{\lambda}$ and $\hat{\mu}$.

Table 2. Experiment result: Scaling-down case

Iteration	SLA violation probability (%)	VM Utilization rate (%)	Number of VMs
0	0.01361	38.45500	120
1	2.68980	76.27439	60.5
2	1.81132	72.64227	63.525
⋮	⋮	⋮	⋮
5	0.583491	62.75113	73.87277
6	0.403263	59.76298	77.21503

5. CONCLUSION

Although the scaling operation of the cloud computing system has several benefits, there are still complexities deciding the proper number of VM instances being provisioned in a PM due to the fluctuation and of task request arrivals. To counter those complexities, this work presented an efficient and simple VM-level scaling scheme using an analytical performance of an M/M/1-PS queueing model. The goal of the proposed scaling scheme is to decide the proper number of VM instances satisfying the SLA target related to the response time and the utilization rate of available VMs. Experimental results show that our VM-level scaling algorithm works well to fine the proper number of VM instances complying with SLAs.

One of the future research topics is to model a VM as a finite buffer system. In other words, we should investigate the task dropping (or blocking) phenomena due to the limited capacity of a VM. It may be more realistic to model a VM as a finite buffer processor sharing queue.

ACKNOWLEDGEMENT

This study was supported by 2016 Research Grant from Kangwon National University.

REFERENCES

[1] M. K. Kim and J. Y. Choi, “An efficient two-phase heuristic policy for acceptance control in IaaS cloud service,” Journal of the Society of Korea Industrial and Systems Engineering, vol. 38, no. 2, 2015, pp. 91-100.

[2] T. W. Um, H. Lee, R. Woo, and J. K. Choi, “Dynamic resource allocation and scheduling for cloud-based virtual content delivery networks,” ETRI Journal, vol. 36, no. 2, 2014, pp. 197-205.

[3] Z. Zhuang and C. Guo, “Building cloud-ready video transcoding system for content delivery networks (CDNs),” Proc. IEEE GCC, 2012, pp. 2048-2053.

[4] J. He, Y. Wen, J. Huang, and D. Wu, “On the cost-QoE trade-off for cloud-based video streaming under Amazon EC2’s pricing models,” IEEE Transactions on Circuits and Systems for Video Technology, vol. 24, no. 4, 2013, pp. 669-680.

[5] S. P. Ponnusamy and E. Karthikeyan, “Cache optimization on hot-point proxy caching using weighted-Rank cache replacement policy,” ETRI Journal, vol. 35, no. 4, 2013, pp. 687-696.

[6] R. S. Huckman, G. P. Pisano, and L. Kind, *Amazon web service,* Harvard Business School Case (609-048), 2008.

[7] http://www.rightscale.com.

[8] http://software.dell.com/products/cloud-manager.

[9] H. Masuyama and T. Takine, “Sojourn time distribution in a MAP/M/1 processor-sharing queue,” Operations Research Letters, vol. 31, no. 5, 2003, pp. 406-412.

[10] H. C. Tijms, *Stochastic models: an algorithmic approach*, John Wiley & Sons, Inc, 1994.

[11] F. Guillemin and J. Boyer, “Analysis of the M/M/1 queue with processor sharing via spectral theory,” Queueing Systems, vol. 39, no. 4, 2001, pp. 377-397.

[12] P. Barham, B. Dragovic, K. Fraser, S. Hand, T. Harris, A. Ho, R. Neugebauer, I. Pratt, and A. Warfield, “Xen and the art of virtualization,” Proc. ACM SOSP, 2003, pp. 164-177.

[13] B. Sotomayor, R. S. Montero, I. M. Llorente, and I. Foster, “Virtual infrastructure management in private and hybrid clouds,” IEEE Internet Computing, vol. 34, no. 5, 2009, pp. 14-22.

[14] O. Litvinski and A. Gherbi, “Openstack scheduler evaluation using design of experiment,” Proc. IEEE ISORC, 2013, pp. 1-7.

[15] J. Huang, C. Li, and J. Yu, "Resource prediction based on double exponential smoothing in cloud computing," Proc. IEEE CECNet, 2012, pp. 2056-2060.

Doo Ho Lee
He is an assistant professor in the department of Industrial and management engineering at Kangwon National University in Republic of Korea. He obtained his Ph.D. in Industrial Engineering from Korea Advanced Institute of Science and Technology (KAIST), Republic of Korea. The focus of his research is on theory and application of queueing systems.