Just-in-time server procurement to private cloud for mobile thin-client service

Fumio Machida[1], Shunsuke Kohno[1], Kosuke Maebara[2], and Masayuki Nakagawa[3]
[1]Knowledge Discovery Research Laboratories, [2]Carrier Services Division, NEC Corporation
[3]Network Software Development Operations Unit, NEC Communication Systems, Ltd.
{f-machida@ab, s-kohno@ct, k-maebara@ak}.jp.nec.com, and nakagawa.msy@ncos.nec.co.jp

***Abstract*— Mobile thin-client services are gaining lots of attention from companies who concern about the security yet recognize the benefits of mobile computing in their businesses. The service is based on a private cloud system hosting virtual machines that can execute mobile OS instances. The owner of the private cloud needs to prepare sufficient server resources for hosting those virtual machines. In this paper, we propose a framework to guide server procurement decisions in a private cloud for mobile thin-client service, which aims to minimize the cost of unused servers while avoiding service level violation due to the lack of resources. To make a timely server procurement decision, the framework combines the techniques for workload estimation to individual VMs, demand estimation of newly created VMs and repetitive simulation of VM replacement algorithm. Through a simulation study, we show that the proposed framework can reduce the cost of unused servers by 25% while satisfying a service level, compared with a time-based heuristic decision method.**

Keywords*— *cloud computing, optimization, server procurement, smartphone, virtual machine

I. Introduction

Smartphones and tablets are widely accepted for business use, not only for communication tools but also for terminal devices for remote working. Cloud service providers like NEC provide secure smartphone solutions called Virtualized Smartphone (VSP) [1], where mobile OS instances are executed on virtual machines hosted in a private datacenter and users can access the instances from their mobile devices through encrypted communication. Since the proprietary data is stored in the secure datacenter and is accessible via secure communication, the risk of information leak is reduced even when the mobile devices are lost or theft. Customers of VSP solution need to establish private cloud environments in secure sites. The server resources for the private cloud are necessary for accommodating mobile OS instances, but they impose the cost as well according to the amount of server resources required. On the other hand, once a customer decides to add a new server, the additional resource becomes an idle server which is regarded as wasted cost until the server is really in operation. Hence a timely server procurement decision for optimizing the resource cost is a key challenge of cloud owners.

In this paper, we propose a framework to assist the decision of timely server procurement by workload characterization of VSP instances and demand prediction for virtual machines. Based on a practice, we assume that VMs for mobile OS instances are reallocated among host servers on the cloud at a certain time interval (e.g., everyday) by a VM replacement algorithm. A difficulty in the decision of server procurement is partly caused by such a behavior of replacement algorithm, which typically relies on heuristics and results in an ad-hoc placement. Depending on the algorithm or policy for VM reallocation, the time when a new server is required significantly changes. Such problem has not been addressed in the research of mobile cloud computing [2]. To address this issue, we exploit a simulation of VM replacement algorithm in the framework so as to predict the time to resource contention resulting in additional server requirement. The effectiveness of the proposed framework is presented through the experiments in comparison with decision methods relying on heuristics. Our experimental results show that the proposed decision framework outperforms the heuristic approaches in terms of server unused period (UNP) while not violating a given service level.

The rest of the paper is organized as follows. Section II introduces the configuration of the target service, VSP. Section III formulates the problem of server procurement decision. In Section IV, we propose a procurement planning framework that includes the simulation of VM replacement algorithm. Section V shows the evaluation results and Section VI gives our conclusion.

II. Virtual Smartphone service

VSP is a thin-client service that offers computing resources in a secure datacenter to mobile devices through encrypted network communication. Operating systems for smart devices are installed in virtual machines and their images and data are saved in the datacenter. A security policy prohibits the smart device users from saving any confidential data in the local storage and hence the company can minimize the risk of information leak due to accidental lost of the mobile device. All the processes associated with company activities, such as mailer, scheduler and document viewers, are run in the datacenter environment instead of the mobile device. User inputs on the smart device are transferred to the corresponding VM instance in the datacenter and, on the other side, the only screen data is transferred to the mobile device (see Fig. 1). The content of user inputs and screen data is protected by encrypted communication.

978-3-901882-77-7 © 2015 IFIP

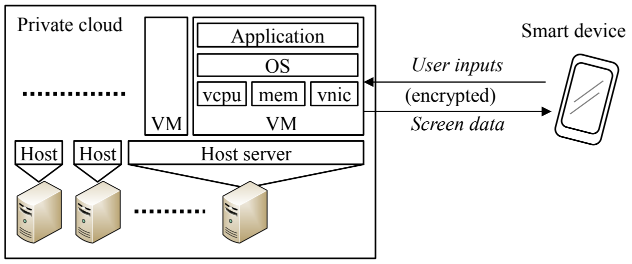

Fig. 1. A service architecture of virtual smartphone service

Applications and an operating system run on a virtual machine consume shared computing resources in the datacenter. Increased number of VMs might cause resource contention which results in performance degradation of user applications. User-perceived performance is affected not only by resource contention in the datacenter (e.g., CPU, memory and disk I/O) but also by end-to-end network bandwidth between the datacenter and the mobile device.

We focus on the case that CPU resource in the datacenter is the bottleneck of the service performance. To keep a good performance of smartphone service, it is important to avoid CPU resource contention proactively by adding server resources for hosting VMs. Meanwhile unused server incurs unnecessary cost. To reduce the idling periods of server resources, a virtualized datacenter employs a heuristic approach to optimize VM placement such that all the workloads of the VMs are packed in the given server resources.

III. Server Procurement Decision

Server procurement decision problem can be regarded as a variation of classical inventory management problem with respect to the following two performance measures.

- Server unused period (UNP)
  UNP represents the time period between the time procured server is ready for use and the time when the server actually starts being used in the system. To reduce the unused resources and maximize the efficiency, the server unused period should be minimized.
- CPU overload
  CPU overload event occurs when the average CPU utilization of a server in a time window exceeds a predefined threshold value. CPU overload is caused by the lack of computation resources and it degrades the quality of the services hosted on the server.

Let $N(t)$ denote the number of CPU overload events observed in the system for time interval t from when the last server is added to the system. Since $N(t)$ is a counter of events from t=0, it increases monotonically over time. We assume the system has sufficient computing resources at the beginning and thus there is no overload as represented by $N(0) = 0$. The probability that the $N(t)$ becomes positive (i.e., $\Pr[N(t) > 0]$) increases according to workload fluctuations as well as increased demands for new VM instances. In order to avoid a CPU overload event, additional server resource should be procured before $N(t)$ becomes positive. Let t_{proc} be the time when a server procurement decision is made. The ordered server will be delivered with a certain lead time d. A server procurement should be scheduled under the constraint $N(t_{\mathrm{proc}} + d) = 0$ so as to avoid CPU overload event. At the time $t_{\mathrm{proc}} + d$, the procured server becomes an unused server resource and UNP starts. To reduce UNP, t_{proc} should be postponed as much as possible into the future. When we regard t_{proc} as a decision variable, the problem can be defined as below.

Problem 1: *Server procurement decision problem*
Determine the maximum server procurement time t_{proc} under the condition $N(t_{\mathrm{proc}} + d) = 0$.

Note that $N(t_{\mathrm{proc}} + d)$ is not given a priori since the number of overload events in the future depends on the dynamics of a system under uncertainty. However, it is a monotonic function over time and thereby $\Pr[N(t_{\mathrm{proc}} + d) > 0]$ increases as t_{proc} becomes larger. The condition $N(t_{\mathrm{proc}} + d) = 0$ constrains the value of t_{proc} from above. We capture the system dynamics by stochastic processes and present a framework to estimate the maximum t_{proc} based on a simulation model.

IV. Procurement Planning Framework

We propose a framework to determine the optimum server procurement time for VSP by combining the techniques for VM demand estimation and workload prediction.

A. Architecture

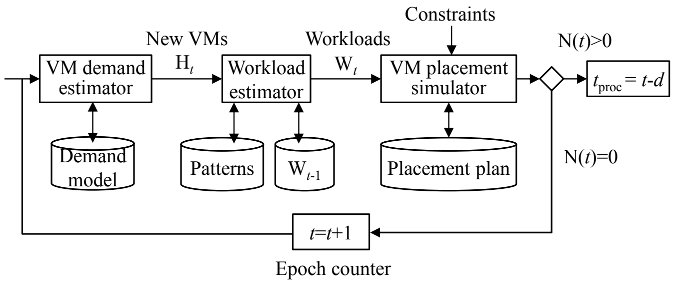

Fig. 2. Block diagram of the server procurement decision framework

The overview architecture of the proposed framework is shown in Fig. 2. The framework includes two estimators; VM demands estimator and workload estimator. The VM demand estimator predicts the number of new requests for VMs, H_t, that arrive in the next time period. The estimation is based on the demand model which is assumed to derive from the history of new VM requests. While, the workload estimator is used for predicting the CPU workloads of individual VMs, W_t, for the next time period based on the patterns derived from the history data of CPU utilization. The estimated workloads are then fed into the VM placement simulator in which a VM placement algorithm determines the optimum VM placement in the server infrastructure under some allocation constraints. An important constraint is the CPU resource allocation constraint which ensures that the CPU overloads do not occur in any hosts. When the CPU resource allocation constraint is not able to be satisfied with the existing host servers, a new host server is decided to be added. If a procured server is not ready until then, the CPU overload counter, N(t), increases in the next period.

To determine the optimum server procurement decision time, the overall framework works as a simulator. The simulator has a virtual timer which starts at the beginning of the simulation with the current state of the target VSP system. We assume that VM replacement takes place in a periodic maintenance period (e.g. every midnight) and define *epoch* as the time period bounded by the latest decision of VM placement. The simulation timer contains an epoch counter which counts the number of epochs from the beginning of the simulation. When the epoch counter increments, first the number of new requests for VMs during the new epoch, H_t, is estimated by VM demand estimator. Next, for all the target VMs in the system, the workloads during the new epoch, W_t, are estimated by workload estimators. The results are then used in the VM placement simulator that determines whether the existing host servers are tolerant to the workload changes and increased demands. If the host servers can accommodate all the VMs by replacement, the simulation proceeds to the next epoch. If not, it means CPU overload is inevitable without additional server resource. We can regard the next epoch as the time when the condition $N(t) = 0$ is violated. In this case, subtracting the lead time d from the virtual time, we can get the maximum server procurement time t_{proc}. Note that the optimum server procurement time is 0 when the lead time d is larger than the time to reach $N(t) > 0$ (i.e., $N(d) > 0$). VM demands estimator and workload estimator are detailed in the subsections below.

B. *VM demand and workload estimators*

In response to the increasing demands to a VSP service, we assume that the number of VMs monotonically increases over time. The requirements of VM instantiations can be considered as a random request arrival process, thus we model the VM demands by a Poisson Arrival Process. Once a VM is instantiated in the cloud, the workload of the VM changes over time. With the assumption that VM workloads in an epoch are categorized into a certain small-set of patterns, we employ a pattern estimation approach for VM workload estimation. Let us define workload pattern as a vector of average CPU utilizations in k segmented time intervals in an epoch, $\boldsymbol{u} = (u_1, u_2, \dots, u_k), u_1, u_2, \dots, u_k \in [0,100]$. Workload changes in the consecutive epochs can be represented by the transitions among workload patterns. Let S be the set of patterns, which is constructed from history data, assume that any observation of workload change in an epoch is mapped onto a pattern $\boldsymbol{u}^{(i)} = \left(u_1^{(i)}, u_2^{(i)}, \dots, u_k^{(i)}\right), 1 \le i \le l$, where l represents the number of patterns identified. By analyzing the statistics about the number of transitions among patterns $\boldsymbol{u}^{(i)}$ to $\boldsymbol{u}^{(j)}$ ($\boldsymbol{u}^{(i)}, \boldsymbol{u}^{(j)} \in S$), we can estimate the transition probabilities among individual patterns. The state space S and the estimated transition probabilities over S define a Discrete Time Markov chain (DTMC) [2] which capture the probabilistic behavior of pattern transitions over time epochs. Let Ξ be the transition probability matrix of the DTMC whose (i, j)-element represents the transition probability from pattern $\boldsymbol{u}^{(i)}$ to $\boldsymbol{u}^{(j)}, 1 \le i, j \le l$. The DTMC characterized by Ξ is used to predict the VM workload pattern in the next time epoch by the following steps.

1) For a target VM, the latest workload pattern is observed as $\tilde{\boldsymbol{u}} = (\tilde{u}_1, \tilde{u}_2, \dots, \tilde{u}_k)$. Compute the Euclid distances between $\tilde{\boldsymbol{u}}$ and any $\boldsymbol{u} \in S$ and determine the pattern $\boldsymbol{u}^{(i)}$ which has the shortest distance from $\tilde{\boldsymbol{u}}$.
2) Determine a next workload pattern $\boldsymbol{u}^{(j)}, 1 \le j \le l$, which has a transition from $\boldsymbol{u}^{(i)}$ in the DTMC, sampled by the corresponding probability. In other words, $\boldsymbol{u}^{(j)}$ is chosen as the next workload pattern by probability Ξ_{ij}.

We also need to estimate the workload of new VMs which is expected to start in the next epoch and has no recorded workload in the previous epoch. We can incorporate this boundary case into the pattern transition DTMC by adding the pattern for start-up workload. In the DTMC, the transition from start-up workload to any workload patterns is defined and the associated transition probabilities are estimated from the workload history of start-up VMs. In summary, the workload estimator predicts both the workload for existing VMs and new VMs in the next epoch based on underlying DTMC constructed from the history of VM workload change.

C. *VM replacement algorithm*

VM replacement algorithm, which reallocates VMs to host servers during a maintenance period, has a significant impact on the time to resource contention event (e.g., CPU overload event) resulting in server addition. Depending on heuristic algorithms or optimization methods used in the system, the total number of VMs that can be hosted by the given host servers changes [4]. There are several VM placement approaches which take into consideration the number of VM migrations required for replacement. Here we assume that the system employs a heuristic algorithm for VM replacement. In particular, we use Ajiro algorithm [5], which is a heuristic algorithm to determine a VM replacement plan with small number of VM migrations from the current state of VM placement. The algorithm is based on an iterative deepening search method with respect to the number of VMs extracted from each host for replacement. Note that the proposed framework does not rely on any specific VM replacement technique, we can employ different methods in the framework.

V. EVALUATION

To show the feasibility and the effectiveness of our framework, we conduct a simulation experiment. We capture the workload characteristics by the patterns, as described before. In our preliminary experiments, we use the pattern matrix $\boldsymbol{U}$ and the pattern transition matrix Ξ defined below. The values of $\boldsymbol{U}$ are determined by the real observed data from a trial system.

$$\boldsymbol{U} = \begin{bmatrix} \boldsymbol{u}^{(1)} \\ \boldsymbol{u}^{(2)} \\ \boldsymbol{u}^{(3)} \\ \boldsymbol{u}^{(4)} \\ \boldsymbol{u}^{(5)} \end{bmatrix} = \begin{bmatrix} 0 & 0 & 0 & 0 & 24 & 8 & 6 & 6 & 5 & 5 & 5 & 5 & 4 & 4 & 4 & 4 \\ 4 & 4 & 4 & 4 & 4 & 4 & 4 & 4 & 4 & 4 & 4 & 4 & 4 & 4 & 4 & 4 \\ 5 & 5 & 8 & 19 & 28 & 17 & 14 & 14 & 12 & 9 & 8 & 7 & 6 & 6 & 9 & 7 \\ 4 & 4 & 10 & 8 & 5 & 5 & 4 & 4 & 4 & 4 & 4 & 4 & 4 & 4 & 4 & 4 \\ 4 & 4 & 4 & 4 & 4 & 4 & 4 & 4 & 4 & 34 & 19 & 11 & 8 & 6 & 6 & 5 \end{bmatrix}$$

$$\Xi = \begin{bmatrix} 0 & 0.3 & 0.3 & 0.2 & 0.2 \\ 0 & 0.6 & 0.1 & 0.1 & 0.1 \\ 0 & 0.3 & 0.5 & 0.1 & 0.1 \\ 0 & 0.2 & 0.1 & 0.6 & 0.1 \\ 0 & 0.3 & 0.2 & 0 & 0.5 \end{bmatrix}$$

We use a Poisson arrival process with rate 1/12 [1/hours] to create the VM demands. VM replacement takes place once a day in the maintenance period and we do not consider dynamic VM replacement during the usage period. Each host can accommodate VMs as long as the sum of their average CPU utilizations in the same time interval does not exceed 600. Fig. 3 shows the results of the simulation based on the sample paths of VM demand arrivals by the Poisson process.

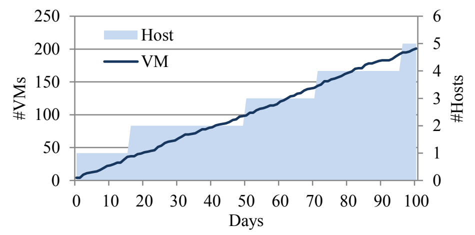

Fig. 3. A simulated sample path of the number of hosts and VMs

Using the data shown in Fig. 3, we evaluate the effectiveness of the framework compared with the two simple heuristic approaches. The first heuristic is an approach to order a new host after a lapse of predefined period t_D from the time the previous host is added to the system. The ordered server is delivered after the lead time that is assumed to be three days (d=3). The second heuristic considers the number of VMs created in the system after the latest server and decides the next server procurement when the number of created VMs reaches to a threshold n_D. When the number of new VMs exceeds n_D, an order of a new server is placed and the server is delivered after the lead time d.

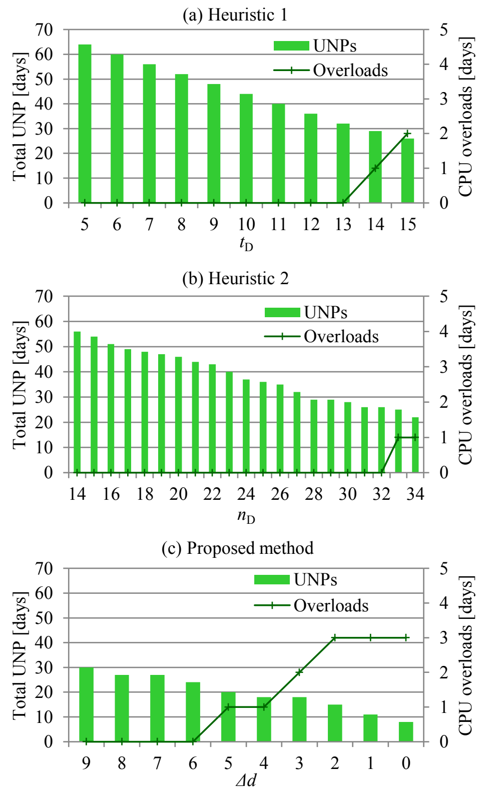

Fig. 4. The simulation results of total UNPs and CPU overload events

Unlike the two heuristic approaches, our framework can take into consideration the demand changes, workload changes and the effects of VM replacement algorithm. In the experiments, we conduct the simulation five times and decide a procurement when the next server addition is expected to occur after $d + \Delta d$ days on average. Δd is a simulation parameter which represents the margin of a delivery time.

As defined in the problem in Section III, the objective is to maximize the server procurement time without violating the condition $N(t_{\text{proc}} + d)$ to minimize UNP. We evaluate the approaches by the total UNP observed until fifth server is added to the system and the number of CPU overloads due to delayed server procurement.

Fig. 4(a) shows the UNPs and CPU violations result from the heuristic 1 with different value of t_D. As t_D increases the UNP decreases, while the number of CPU overloads starts increasing at certain time points. The UNP is minimized to 32 days by setting t_D = 13. Heuristic 2 is expected to yield better decisions because it can consider the number of VMs arrived to the system. Fig. 4(b) shows the UNPs by the heuristic 2 with different value of n_D. The higher n_D gives the better UNP, while the risk of CPU overloads increases. The UNP is minimized to 26 days by n_D = 32, which is 18.7% smaller than the result of the heuristic 1. Finally, Fig. 4(c) shows the results of our framework with different margin Δd for lead time d. When we set Δd =6, the UNP is minimized to 24 days, which is 25% smaller than the heuristic 1 and 7.6% smaller than the heuristic 2. Through the above preliminary experiments, we confirm that the proposed framework can reduce UNPs drastically while avoiding CPU violations.

VI. CONCLUSION

Making a timely decision of server procurement in a private cloud system for smart devices is an important and challenging issue. We formulated the problem of server procurement decision in light of the trade-off between the useless period caused by a premature decision and the increased risk of service level violation due to delayed decision. We proposed a framework to guide a better decision of server procurement by using workload estimation for individual VMs, demand estimation of new VMs and simulation of a VM replacement algorithm. Through the experiments, we have shown that the proposed framework can reduce the UNPs compared with the simple heuristic approaches.

REFERENCES

[1] NEC Virtualized Smartphone, http://www.nec.com/en/global/solutions/nsp/vsp/index.html

[2] Z. Sanaei, S. Abolfazli, A. Gani, and R. Buyya, Heterogeneity in mobile cloud computing: Taxonomy and open challenges, IEEE Communications Surveys & Tutorials, vol. 16, no. 1, pp. 369-392, 2014.

[3] K. S. Trivedi, Probability and Statistics with Reliability, Queuing, and Computer Science Applications, John Wiley & Sons, 2nd edition, 2001.

[4] F. Machida, M. Kawato, Y, Maeno, Redundant virtual machine placement for fault-tolerant consolidated server clusters, In Proc. of NOMS2010, pp. 32-39. 2010.

[5] Y. Ajiro, Recombining Virtual machines to autonomically adapt to load changes, In the Proc. of Annual Conf. of the Japanese Society for Artificial Intelligence, (In Japanese) 2008.