To pre-copy or To post-copy, That is the Question

Asraa Abdulrazak Ali Mardan[1,a)] Akane Koto[1,b)] Kenji Kono[1,c)]

Abstract: Live migration of virtual machines (VMs) is a promising technique for cloud management. It allows cloud administrators to control and manage the computing resources flexibly in the cloud. Live migration offers two policies: 1) pre-copy and 2) post-copy. The differences in the policies result in different performance characteristics. The trade-off stemming from the differences in the policies must be taken into consideration when a virtual machine is to be moved from one physical machine to another. In this paper, we summarize our previous results of the performance comparison of Xen pre-copy, KVM pre-copy and KVM post-copy, and show a simple model that estimates the migration time of Xen pre-copy and KVM post-copy.

1. Introduction

Live migration of virtual machines (VMs) is a promising technique for cloud management. Live migration moves VMs across different physical machines without disrupting the services offered by VMs. This feature enables cloud administrators to manage computing resources in the cloud with great flexibility. For example, the administrators can mitigate heavy load of a physical machine by moving some VMs running on the machine to another. If the administrators move all VMs from the machine, it can be shutdown for maintenance or power saving. Since a management policy of the computing resources differs in clouds, the administrators can use live migration for various purposes. Live migration can be used in the virtualization technique that recent cloud platforms (e.g., Amazon Web Services [1], Google Compute Engine [2], and Microsoft Azure [3]) commonly use as their technical base.

However, using live migration effectively in clouds is sometimes difficult because cloud can be customized variously depending on the principle of the cloud that defines workloads, a management policy, and implementations of the cloud. Since the behavior of live migration differs in the migrating VM's workload, cloud administrators serving various kinds of services may be bothered with unpredictable effects of live migration. Specifically, Clark *et al.* [4] say that live migration adjusts downtime and migration time of live migration in response to the memory access pattern of the migrating VM. In addition to the workloads, the administrators can select a migration policy depending on their purpose of live migration, which may cause unpredictable negative impact on cloud services. Our experimental results reveal that one of the famous migration policy named pre-copy [4] may increase downtime and migration time of live migration, and another famous migration policy named post-copy [5] may cause terrible performance interference on the migrating VM. In cloud environment, cloud administrators can customize implementations of the cloud, which makes the effect of migration policy more complicated. For example, our experimental results show that migration time, downtime, and performance interference on VMs differ in Xen and KVM. Since various things have an effect on the performance of live migration and clouds, the administrators should take care of many things for effective manage of the cloud.

For considering the causes of performance instability of live migration, understanding the effects of live migration in each cloud environment is practical way. To understand the effects of live migration, downtime and migration time are sometimes used as metrics for evaluating the performance of live migration. These metrics show the length of time in the part of the whole of live migration, and are used in many previous studies for determining the replacement of VMs among physical machines. Since live migration may cause performance interference on VMs, performance degradation of each VM is also an important metrics for evaluating live migration. In this study, we define the performance interference by live migration as *migration noise*. Especially, we observed CPU, network, and memory usage of each VM and use the degradation of each as migration noise adding as metrics of live migration in our experiments.

In our previous study [6], we compare the performance of three migration methods. Specifically, we use two migration policies: pre-copy and post-copy and two VMMs: Xen and KVM in our experiments. Since Xen post-copy is not available to the public, we compare three migration methods: Xen pre-copy, KVM pre-copy, and KVM post-copy. Our major finding is that Xen pre-copy, KVM pre-copy, and KVM post-copy all show different trends in migration time. Thus, it is important to choose an appropriate policy and/or implementation of live migration in the cloud. In this paper, we show a simple model to estimate the migration time of Xen pre-copy and KVM post-copy.

[1] Department of Information and Computer Science, Keio University
[a)] asraaiteng@sslab.ics.keio.ac.jp
[b)] koto@sslab.ics.keio.ac.jp
[c)] kono@sslab.ics.keio.ac.jp

© 2015 Information Processing Society of Japan

2. Migration policies and Implementations

The performance of live migration differs in clouds since cloud administrators can customize clouds and live migration following their principles. Live migration has been widely studied and many policies and implementations have been proposed. This diversity gives cloud administrators several choices for customizing the cloud. In this section, we explain the difference of pre-copy and post-copy, and difference of Xen and KVM focusing on the performance stability of live migration.

2.1 Migration policy: Pre-copy vs Post-copy

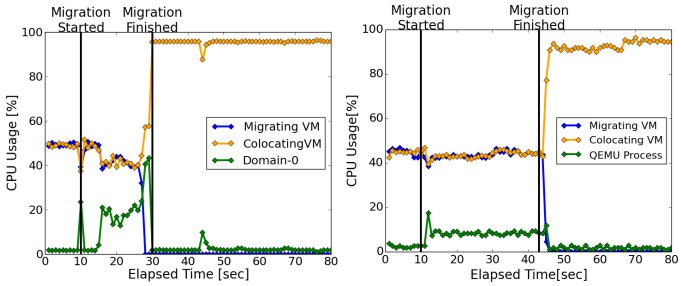
Pre-copy and post-copy aim to reduce downtime of live migration with their own memory transfer mechanism, which may show different trends under the same workloads. Downtime is the duration of stop-and-copy, one of the migration phase in which migrating VM should be suspended for converting its workspace from the source to the destination. Pre-copy reduces # of memory pages transferred during stop-and-copy by transferring most of the VM's memory pages in iterative pre-copy phase. In iterative pre-copy phase, pre-copy transfers most of the memory pages at first, and re-transfers only dirty pages (i.e., memory pages updated in the previous memory transfer) iteratively without disturbing the VM. If the # of remaining dirty pages becomes small enough, pre-copy stops iterative pre-copy and transfers only frequent-updated dirty-pages and the VM's device states in stop-and-copy. Since the # of remaining dirty pages differs in the VM's dirty-rate and network capability, pre-copy may increase downtime and migration time. On the other hand, post-copy defined the types of memory pages transferred in stop-and-copy, which maintains downtime independently from the VM's dirty-rate. The flow of post-copy is divided in two phases: stop-and-copy and demand paging. In post-copy, all VM's memory pages are transferred only once; first it transfers only the VM's device states in stop-and-copy and transfers remaining memory pages in response to the VM's workloads. Since this feature, post-copy also prevents migration time becoming longer in response to the VM's dirty-rate.

In post-copy migration, cloud administrators should consider the time of fetching each memory page appearing during demand-paging phase. After the stop-and-copy phase, post-copy fetches each of the remaining memory pages from the source host via the network. Since each of the memory fetching is triggered by the VM's memory access on the destination, the VM may be waited for a long time when the VM accesses many memory pages in a short time after it resumes on the destination. This waiting time may cause terrible performance degradation of the VM. By contrast, pre-copy migrates most of the VM's memory pages without using the requests of the VM and does not causes such stopping time.

2.2 Implementation: Xen vs KVM

2.2.1 Difference on VMM

Xen and KVM can provide para-virtualized VM with different CPU management architecture, which may cause different trends



(a) Xen pre-copy (b) KVM pre-copy

Fig. 1 CPU utilization of each VM on the source host (workload of migrating VM set working set size (WSS): 64 MB, Dirty-rate: 15360 pages/sec). Two VMs (migrating VM and collocating VM) running on the source cause CPU resource contention, and thus live migration starts moving migrating VM at 10 seconds to mitigate the contention.

in CPU resource allocation for migrating VM, collocating VM, and migration process. In Xen, all VMs are served as *domain*: one administer VM (domain-0) manages other VMs (domain-Us), and CPU times for each VM is fairly allocated with credit scheduling algorithm. All instructions for resource management of VM including live migration are executed in domain-0. Unlike Xen, KVM manages all VMs as QEMU process, a device emulator running with KVM, and allocate CPU times of each VM as CPU threads of the process. In KVM, all CPU times are managed with completely fair scheduler (CFS) implemented in Linux. Since KVM entrust the execution of live migration to QEMU, live migration is executed as a thread of the QEMU process. In our experiment, we reveal that these differences on management architecture and scheduling algorithm vary the impact of live migration. Figure 1(a) and (b) are examples of our experimental results. The green line in each graph is the CPU utilization of the migration process. These figures show that each migration process shows different trends on CPU utilization per unit time and the duration that the migration process consumes CPU. These differences also vary the impacts on migrating VM and collocating VMs. In fact, these figures shows that Xen pre-copy causes much performance interference on VMs than KVM pre-copy, but the interference continues shorter than KVM pre-copy.

The memory management architecture implemented in Xen and KVM differently interfere VMs during and after live migration. Xen cannot free VM's unused memory automatically since it does not support memory overcommit. If cloud administrators want to rearrange each VM's memory in Xen, they should resize another VM's memory to small before they increase the size one VM with insufficient memory. Even Xen uses self-ballooning mechanism that rearranges the size of VM's memory automatically depending on the workload, the administrators cannot use the mechanism with live migration because the mechanism does not support live migration. In order to use live migration for memory management of the cloud, they should resize VM's memory after the migrating VM is migrated and freed on the source host. By contrast, KVM can rearrange VM's memory automatically even when it is executing live migration. Since KVM uses memory management mechanism implemented in Linux, KVM can use memory overcommit, which means that it does not need to resize the VM's memory explicitly. However, KVM can-

ⓒ 2015 Information Processing Society of Japan

not free VM's memory completely even after it migrates the VM because of QEMU's architecture. In KVM+QEMU architecture, QEMU keeps holding the VM's memory in preparation for the VM's return from another hosts.

2.2.2 Difference on pre-copy

In addition to the difference of VMM, Xen and KVM provide pre-copy live migration with different implementations. One of the difference between Xen pre-copy and KVM pre-copy is the threshold for stopping iterative pre-copy phase. As described in Section 2.1, pre-copy migration execute iterative pre-copy until the # of remaining dirty pages becomes small enough. To make pre-copy suitable for various workloads, Xen pre-copy use three values; # of remaining dirty-pages, the total # of transferred pages, and # of pre-copy iterations. Using these values, Xen defines three stop conditions of iterative pre-copy: (1) the memory transfer rate required in stop-and-copy may become the same or higher than the maximum network throughput taken for migration. (2) The total # of iterations becomes 30. (3) The # of remaining dirty pages in the iteration becomes smaller than 50. (4) The total # of pages transferred during iterative pre-copy becomes three times larger than the size of the VM's memory. If one or more of the stop conditions are satisfied, pre-copy stops iterative pre-copy and move to stop-and-copy. In case of KVM, pre-copy uses # of remaining pages only for the threshold. Specifically, KVM pre-copy continuously calculates the time taken for transferring the remaining pages in stop-and-copy during iterative pre-copy, and compare the calculated value to a pre-defined threshold. The pre-defined threshold indicates the maximum downtime allowed in the migration. If the calculated value does not become smaller than the pre-defined threshold, pre-copy cannot move to stop-and-copy and may time out at the worst. Although the threshold can be adjusted before each migration, it is difficult for cloud administrators to set it appropriately depending on the workload. We reveal the fact in our experiments and show the results in Section **??**.

2.2.3 Difference on post-copy

Like the difference on pre-copy, post-copy is also implemented in Xen and KVM with different technique for reducing the waiting time on each memory fetching. Although Xen post-copy is not available to the public, the paper of studying Xen pre-copy [5] shows that Xen post-copy uses dynamic self-ballooning (DSB) mechanism for reducing the total # of the fetched pages. DSB mechanism is an extension of memory ballooning implemented in Xen, which can free the migrating VM's free pages dynamically before the pages are transferred for the fetching. These released free pages can be recreated at the destination by memory ballooning mechanism, DSB mechanism may reduce the # of memory pages transferred in demand paging. By contrast, KVM post-copy uses memory compression mechanism to reduce the size of each memory page. However, since this memory compression mechanism does not reduce the total # of memory fetching over the network, it may not reduce the waiting time under the slow network.

3. Quantitative Comparison

In this section, we briefly summarize the results shown in our previous paper [6].

Table 1 Machine configurations

Host Machine	Dell PowerEdge T610 with Xeon 2.8 GHz CPU, 32 GB RAM
VMM	Xen 4.1.0 / KVM (Linux 3.0.4) + QEMU 1.1-rc1
VM	Fedora 14 (Linux 3.0.4 kernel) with 1 VCPU and 1.5 GB RAM
Network switch	Cisco Catalyst 3750G, Gigabit Ethernet

Table 2 workloads

Migrating VM	change working set size and dirty-rate	
Collocating VM	CPU intensive	simple C program (empty infinity loop)
	Network I/O intensive	Netperf with 2 client machines
	Memory I/O intensive	the same workload as migVM with Full MB WSS, 51,200 pages/sec dirty-rate

3.1 Experimental Setup

We prepare three machines and three para-virtualized VMs as noted in Table 1. The three machines are connected each other with Gigabit Ethernet via the network switch. One of the machines is used as NFS and all VM images locating on the host are accessed from other two machines via the network.

To analyze the worst performance of each migration method, we prepare several biased scenarios with simple benchmark. On migrating VM, we run simple C program requiring two parameters: *working set size (WSS)* and *dirty-rate*. WSS is the range of memory where the VM accesses and dirty-rate is the rate of accessing. These metrics are inspired by the study of each policy based on [4] and [5]. These studies explain that various memory access patterns on migrating VM change the behavior of live migration. During executing live migration with the migrating VM's workload, we run one of the single-resource contention workloads as shown in Table 2. In each experiments, we adjust machine configurations to analyze migration noise caused in each resource clearly. We configured the number of CPU cores attached to each host from 4 to 1 under CPU intensive workload and limited the RAM size of each machines from 32 GB to 2 GB under memory I/O intensive workloads. We execute each experiment 5 times, but all results taken from the each time show similar feature. To explain our results clearly, we use one set of the results selected randomly.

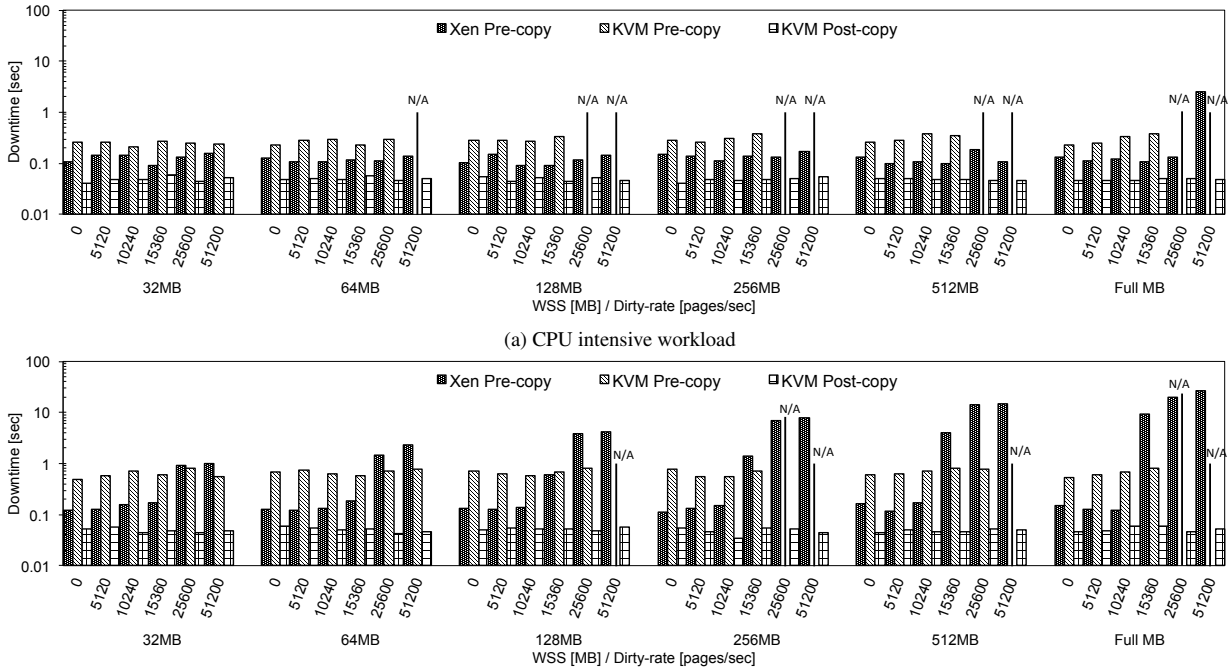
3.2 Experimental results

3.2.1 Downtime

Our experimental results shows that each migration method reduces downtime in various workloads. Figure 2 (a) shows to keep the almost all downtime within 0.4 seconds under CPU intensive workload. Especially, Figure 2 (a) and (b) show that KVM post-copy keeps less downtime than others (34.4-59.5 milliseconds) because post-copy approach keeps the duration of stop-and-copy minimally. Xen pre-copy and KVM pre-copy may also keep less downtime, but their downtime becomes more than 1 second where the network I/O contention occurs. This is because the degradation of memory transfer rate of the migration increases the number of pages transferred during stop-and-copy.

These figures also reveal that different implementation causes different effect on downtime. According to the figures, Xen pre-copy shows less downtime than KVM pre-copy under the light workloads but the situation reverses under the heavy workloads.

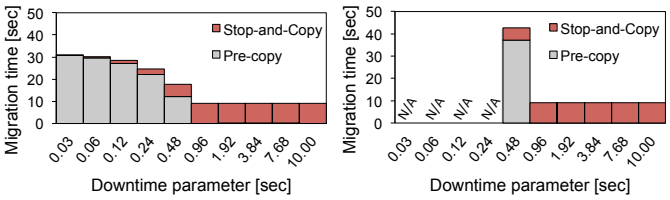
ⓒ 2015 Information Processing Society of Japan



(a) CPU intensive workload

(b) Network I/O intensive workload

Fig. 2 Downtime caused by each migration method under various workloads. Change the parameter of migrating VM's workload when the collocating VMs run CPU intensive workload (a) and network I/O intensive workload (b).



(a) (b)

Fig. 3 Migration time of KVM pre-copy under various user-defined downtime parameter (a) WSS: 0 MB, Dirty-rate: 51,200 pages/sec (b) WSS: Full MB, Dirty-rate: 51,200 pages/sec

Since Xen pre-copy consumes as much computational resource as possible, its downtime changes variously depending on the resource utilization of the host. Actually, Xen pre-copy holds about 800-1000 Mbps during CPU intensive workload but it holds only about 300 Mbps under network I/O intensive workloads, which results in terrible downtime. Comparing with Xen pre-copy, KVM pre-copy does not causes long downtime even under heavy workloads since it keeps resource consumption regardless of the workload.

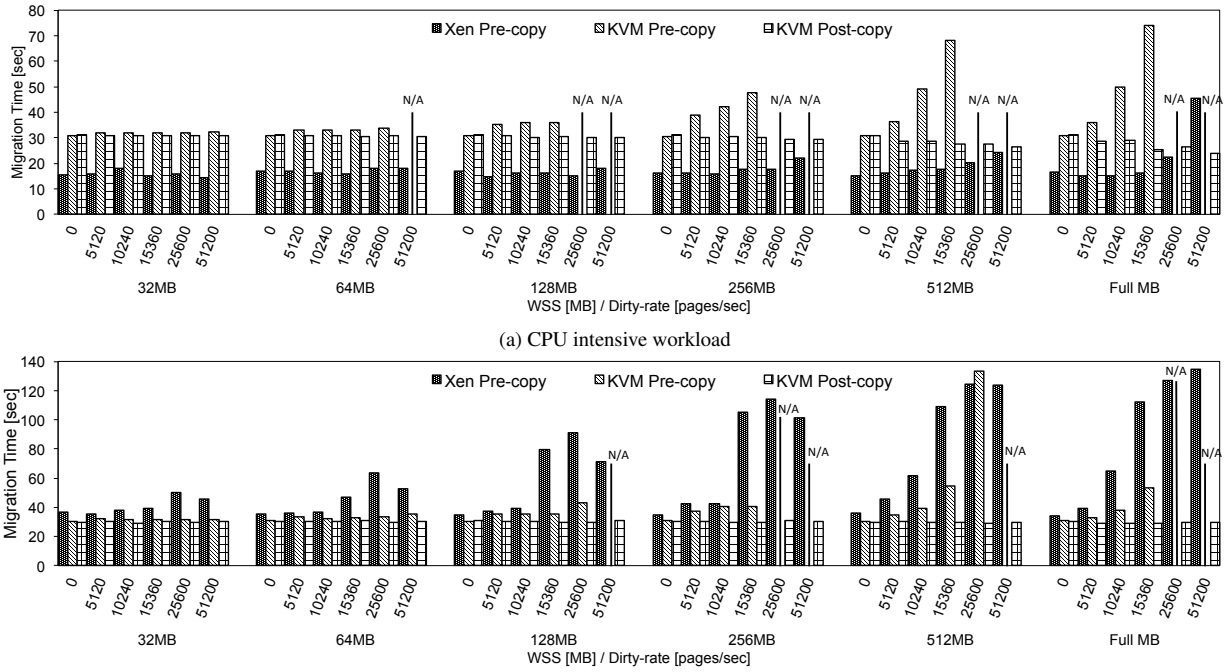
One point to be noted is that KVM pre-copy fails when the workload is heavy. This is because its user-defined parameter prevents the migration when the estimated downtime is heavier than the parameter. Figure 3 shows that KVM pre-copy changes its execution depending on the parameter even under the same workload. Figure 3 (a) presents that KVM pre-copy shows short migration time with long downtime when the parameter is larger than 0.96. Since the estimated downtime instantly becomes lower than the parameter when the value is large, KVM pre-copy moves to stop-and-copy phase suddenly after it starts migration. The parameter is so sensitive that the KVM pre-copy changes its execution violently only when the parameter changes slightly (Figure 3(b)).

3.2.2 Migration Time

Figure 4 (a) and (b) shows that Xen pre-copy takes less migration time than KVM post-copy although post-copy eliminates the redundant memory transfers adopted in pre-copy. As noted in Section 3.2.1, Xen pre-copy consumes much computational resources than KVM pre-copy and KVM post-copy, which shortens the duration of Xen pre-copy. Instead of the less resource consumption, KVM post-copy keeps migration time even under the heavy workload, while Xen pre-copy increases to more than four times as long as KVM post-copy.

These figures also reveal the optimization to reduce the migration time implemented in each method as noted in Section 2. Figure 4 (a) indicates that KVM post-copy shortens the migration time when the WSS is full MB and dirty-rate is higher than 5,120 pages/sec. When the workload becomes heavy, KVM post-copy increases the memory transfer rate. Actually, it increases the network bandwidth from 300 Mbps to 400 Mbps when the workload becomes heavy. Figure 4 (a) and (b) also shows that Xen pre-copy presents a slow curve increase depends on the dirty-rate while KVM pre-copy prolongs the migration time linearly. This is because Xen pre-copy decrease the duration of the 1st pre-copy depending on the workload (Figure **??**). Xen pre-copy regard the pages updated many times during the 1st iteration as the pages that will be updated soon after the subsequent iterations thus it does not need to transfer during the 1st iteration. Instead of these optimizations, KVM pre-copy adopts user-defined parameter to shorten the migration time as shown in Section 3.2.1.

ⓒ 2015 Information Processing Society of Japan



(a) CPU intensive workload

(b) Network I/O intensive workload

Fig. 4 Migration caused by each migration method under various workloads. Change the parameter of migrating VM's workload when the collocating VMs run CPU intensive workload (a) and network I/O intensive workload (b).

4. Estimating Migration Time

4.1 Pre-copy in Xen

To estimate the migration time in the experiments in the previous section, we assume M, W, d represent the total memory size of a migrating VM, the working set size, and the dirty rate, respectively. Here, the page size p is 4Kbyte. Suppose that t_i is i-th iteration time. In the i-th iteration in Xen pre-copy, all the memory pages modified in $(i-1)$-th iteration are transferred to the destination. Therefore, we have

$$t_1 = M/B \tag{1}$$

$$t_i = \min(W,\ p \cdot d \cdot t_{i-1})/B, \tag{2}$$

where B is the available bandwidth. The iteration phase stops if one of the following conditions is satisfied (N is the number of iterations).

(1) V exceeds $3 \cdot M$, where V is the total amount of memory transferred in the iteration phase;

$$\sum_{i=1}^{N} B \cdot t_i > 3 \cdot M.$$

(2) The number of pages to be transferred is less than 50;

$$\min(W,\ p \cdot d \cdot t_N) < 50 \cdot p.$$

(3) The number of iterations reaches 30;

$$N = 30.$$

Therefore, the migration time T_{pre} is estimated to

$$T_{pre} = \sum_{i=1}^{N} t_i.$$

Figure 5 shows the comparison of the estimated and measured migration time of Xen pre-copy.

4.2 Post-copy in KVM

For the post-copy, all the memory pages are transferred to the destination without retransmissions. Therefore, the migration time T_{pst} can be estimated to

$$T_{pst} = M/B.$$

Figure 6 shows the comparison of the estimated and measured migration time of KVM post-copy.

5. Conclusion

Live migration of virtual machines (VMs) is a promising technique for cloud management. It allows cloud administrators to control and manage the computing resources flexibly in the cloud. Live migration offers two policies: 1) pre-copy and 2) post-copy. To give a clue in the selection of the migration policies, this paper summarizes our previous results of the performance comparison of Xen pre-copy, KVM pre-copy and KVM post-copy, and shows a simple model that estimates the migration time of Xen pre-copy and KVM post-copy.

References

[1] : Amazon Elastic Compute Cloud (Amazon EC2).
[2] : Google Compute Engine.
[3] : Microsoft Azure.
[4] Clark, C., Fraser, K., Hand, S., Hansen, J. G., Jul, E., Limpach, C., Pratt, I. and Warfield, A.: Live Migration of Virtual Machines, *Proceedings of the 2nd Conference on Symposium on Networked Systems Design and Implementation (NSDI '05)*, pp. 273–286 (2005).
[5] Hines, M. R. and Gopalan, K.: Post-copy Based Live Virtual Machine Migration Using Adaptive Pre-paging and Dynamic Self-ballooning, *Proceedings of the 5th ACM SIGPLAN/SIGOPS International Conference on Virtual Execution Environments (VEE '09)*, pp. 51–60 (2009).

ⓒ 2015 Information Processing Society of Japan

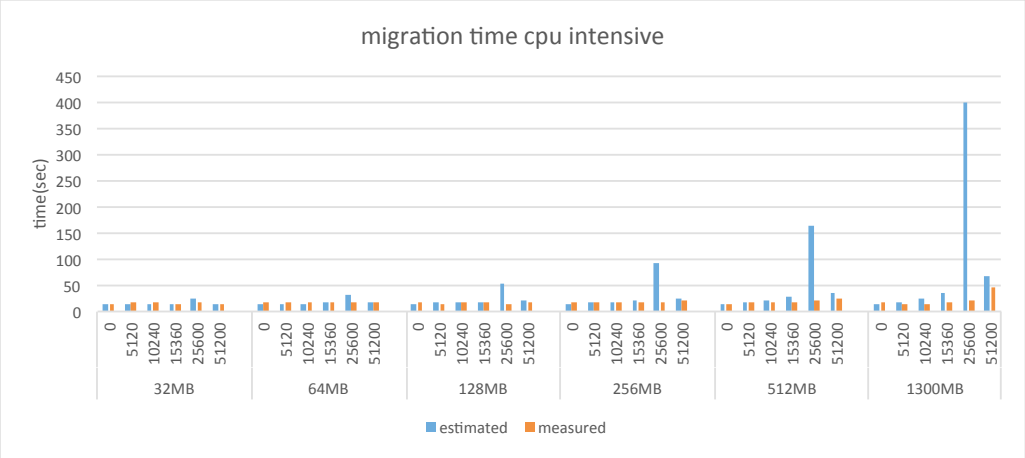


Fig. 5 Comparison of estimated and measured migration time of Xen pre-copy.

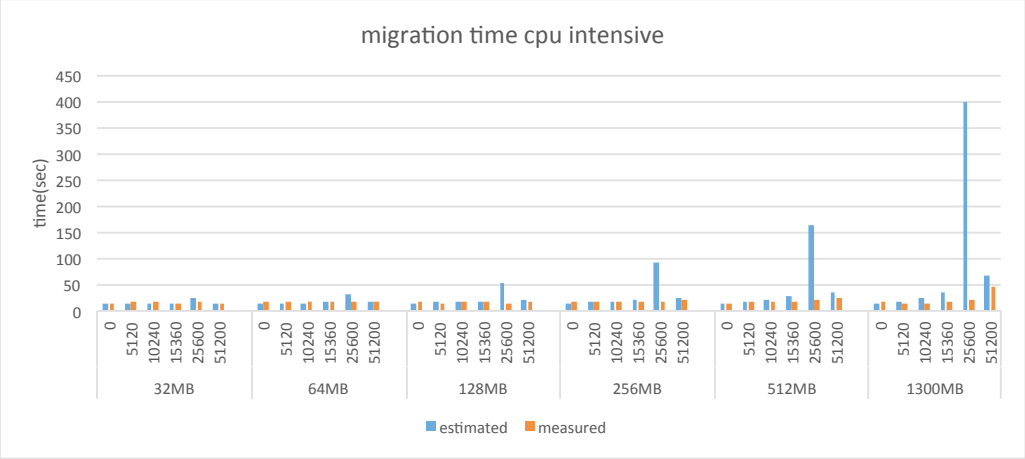


Fig. 6 Comparison of estimated and measured migration time of Xen pre-copy.

[6] Koto, A., Kono, K. and Yamada, H.: A Guideline for Selecting Live Migration Policies and Implementations in Clouds, *IEEE International Conference on Cloud Computing Technology and Science (CloudCom)* (2014).

ⓒ 2015 Information Processing Society of Japan