A Performance Improvement Method for the Global Live Migration of Virtual Machine with IP Mobility

Hidenobu Watanabe[†], Toshihiro Ohigashi[‡], Tohru Kondo[‡], Kouji Nishimura[‡], and Reiji Aibara[‡]

[†]Graduate School of Integrated Arts and Sciences, Hiroshima University
1-7-1, Kagamiyama, Higashi-Hiroshima-Shi, Hiroshima, 739-8521, Japan, h-watanabe@hiroshima-u.ac.jp
[‡]Information Media Center, Hiroshima University, {ohigashi,tkondo,kouji,ray}@hiroshima-u.ac.jp

ABSTRACT

Virtual machine (VM) live migration, which is the ability to move a VM from one physical host to another under hypervisor control without suspending for a long time, is a capability being increasingly utilized. With IP mobility, a capability for migrating VMs among distributed sites is provided. We call it the global live migration. However, it affects an application program on a VM compared with ordinary live migration limited a local link. This problem is caused by IP mobility process is executed after migrating. In the proposed method, the source host executes the network configuration before migrating, which sets destination network information previously to the secondary interface of a VM. The method can reduce the number of steps for IP mobility process after migrating. We have implemented the prototype, and have evaluated our method quantitatively. By the evaluation experiment, we have confirmed that a VM's application downtime during the global live migration is less than 1 second.

Keywords: virtualization, IP mobility, live migration, fast migration

1 INTRODUCTION

A virtualization technology such as VMware [1] or Xen [2] is widely used for electric power saving or usability improvement. It has the special architecture, which a software (called Virtual Machine Monitor or Hypervisor) is interposed between hardware and OS. The technology enables multiple Virtual Machines (VMs) to activate simultaneously on a physical host. In addition, a virtualization supports live migration [3] as sophisticated VM migration. VM migration is the ability to move a VM from one physical host (source host) to another (destination host) on the local link. Live migration enables an application program on a VM to keep running without significantly impacting the application. In this paper, application downtime is denoted by T_{DW}.

E. Harney et al. [4] have been proposed the advanced function for Xen's live migration. The function enables a VM to migrate among distributed sites. We call it *the global live migration*. They have implemented the Mobility Support in IPv6 (MIP6) [5] to the VM itself. Thus, an application on the VM does not notice the conflicted connection between the migrated VM and the destination host. In other words, this method can shift ordinary live migration to the global live migration. We call this method *the VM with IP mobility*.

However, when the VM with IP mobility is used, the global live migration affects an application program on a VM than ordinary live migration. Namely, a VM suspends more than a few seconds during the global live migration despite T_{DW} of ordinary live migration is less than 0.5 second. Therefore, IP mobility for the global live migration increases T_{DW} in exchange for migrating over the Internet. This problem is caused by IP mobility process is executed after migrating. Currently, no one proposes the solved method for this problem.

We aim at bringing T_{DW} of the global live migration close to 0.5 second. In our method, the source host executes the network configuration before migrating. Namely, it sets destination network information previously to one of the interfaces of a VM. After migrating, the destination host just switches to the interface with destination network information. As the result, the global live migration's T_{DW} is almost the same as time of 0.5 second.

The rest of this paper is structured as follows. Section 2 discusses superiority of the VM with IP mobility and an IP mobility architecture used in our method. In addition, we introduce existing techniques for improving the performance of live migration, and we show the purpose of our research. Section 3 illustrates the architecture and the mechanism of the proposed method. We consider efficacy of our method based on the experimental result in Section 4. Section 5 summarizes conclusion and introduce our future work.

2 RELATED WORK

We illustrate the mechanism and two assignments of live migration. VM migration, which is the ability to pretend as if a VM migrated, copies the memory image of a VM from the source host to the destination host after suspending the VM. Live migration, which is able to provide that the suspended time of a VM is nearly zero second, ensures a work memory for restoring state of a memory image. Then, it copies a memory image and memory pages before suspending. A memory page is different between an original memory image and a new memory image during the copy process. The source host keeps copying memory pages until one of two requirements are met. One is that when the VM's memory image was almost copied at the destination host. The other is that when copy frequency reached a limiting value. Finally, live migration copies a last memory page after suspending a VM. Thus, T_{DW} of ordinary live migration is the time taken for copying

a last memory page, and it is less than 0.5 second introduced in [2].

On the other hand, live migration has two assignments. One is the migration range. IP address of a VM does not change after migrating. When a VM migrated to the destination host, the virtual link between the migrated VM and the destination host becomes conflicted state. Thus, ordinary live migration recommends that the migration range is limited to a local link. We think that the global live migration can realize by solving this assignment. The other is fast migration. Live migration increases the total time than VM migration. The total time is the amount of time from the command execution of live migration at the source host to the VM reactivates on the destination host. In this paper, the total time is denoted by T_{ALL}. If virtual memory state keeps changing, copy frequency of memory pages keeps increasing until one of two requirements are met. As the result, live migration expands T_{ALL}, and it increases network traffic too. Thus, the copy process of memory pages becomes of particular importance in order to realize fast migration. We introduce some related works to these assignments.

2.1 System Model for the Global Live Migration

We introduce three system models for the global live migration, and we discuss superiority of the adopted model based on two situations. One of situation is that many VMs are used. The other is that the global live migration is executed frequently.

The first, F. Travostino et al. [6] have proposed the model that some network nodes provide IP mobility. These nodes have a task given respectively. For example, there are the migration support server and the agent server. Former provides the optimum the lightpath between hosts for migrating. The latter provisions some network resources and re-provisions an IP tunnel. This model is able to control live migration dynamic by network nodes without extending hosts and VMs. However, it has to establish IP tunnels every time when VMs migrated. This approach consumes many resources of hosts. In addition, it inflicts a burden on the host itself. Thus, judging from situations to be considered, we think that this model does not suit from the standpoint of utility and efficiency.

The second, L. Qin et al. [7] have proposed the model that hypervisor provides IP mobility, called HyperMIP. They implemented the IP Mobility Support for IPv4 [8] (MIP4) to hypervisor. Thus, HyperMIP can provide the function like the Home Agent (HA) for all VMs on the host. This architecture is similar to the Proxy Mobile IPv6 [9] (PMIP). This model needs not to extend VMs, and it enables the global live migration performance to improve by operating hypervisor directly. However, HyperMIP must relay all packets for each VM, and it must allocate IPv4 address for all migrated VMs. These functions inflict a huge load on the host itself. In addition, we consider that a VM is hard to migrate between hypervisors belonging to different HA since MN can only establish IP tunnel with set HA in MIP4. This point is sufficiently discussed in [7]. Therefore, judging from situations to be considered, we think that this model is not the best model from view points of fault tolerance and an overhead of hypervisor.

The third, E. Harney et al. [4] have proposed the model that the VM provides IP mobility. They have implemented MIP6 [5] to a VM itself. In this model, a VM can communicate as an end host through the optimum route. Then, the destination host needs not to execute some IP mobility processes such as the IP packet relay. In fact, the model can decrease the possibility that the host chokes up. Also, the host can manage multiple migrated VMs at the same time since each VM executes network management. On the other hand, this model request to implement IP mobility in the communicating node. If the client machine does not introduce IP mobility, users can not get a service provided by the VM. However, when two situations are considered, we consider that this restriction does not become a huge issue than problems of above two models. Therefore, we think the model is better than two models as the system model of the global live migration.

2.2 IP Mobility Architecture

We describe the problem of MIP6 in the global live migration, and we introduce the Mobility Support Architecture and Technologies (MAT) [10]. We use MAT as an IP mobility architecture.

MIP6 has the ability to provide the proposed method by extension to use multiple interfaces. In MIP6, HA is set as the fixed node, and the node with MIP6 can communicate with the communicating node without MIP6 through HA. In MAT, all communication nodes must include MAT. MIP6 has the advantage over MAT on this point. However, the communication route is not best when HA is used. MIP6 can use the route optimization method without going through HA, but the method requests IP tunnel. IP tunnel causes increment for IP header length. In some applications such as IP phone or teleconference system, over 32 bytes IP header has possible to become fatal overhead. Therefore, we adopted MAT as IP mobility architecture.

MAT provides communication between end hosts without HA. In MAT, a moving node including the MAT kernel is called as Mobile Node (MN). A node communicating with MN is called as Correspondent Node (CN). MN uses two IP addresses called the Home Address (HoA) and the Mobile Address (MoA). The HoA is an IPv6 address used permanently in an application. The MoA is the temporary IPv6 address allocated every time when MN migrated. After allocating the MoA, MN updates to the IP Address Mapping Server (IMS)[1]. This process is called the IMS update, and it is executed for ensuring the consistency between a HoA and a MoA. Also, the kernel translates a source IP address or a destination IP address in IP packets appropriately. Thus, the mapping information of two IP addresses is managed by not HA but IMS, and reachability of IP packets is provided by MN itself. As well, in MIP6, the process for establishing an IP tunnel is added further after finishing the updating process.

[1]IMS differs from the IP Multimedia Subsystem in this paper.

Thus, MAT has the advantage in that the process work requested for IP mobility is fewer than MIP6 though it has the disadvantage in that CN must include MAT function.

2.3 Fast Migration

We discuss fast migration of live migration and the global live migration. T. Hirofuchi et al. [11] and M. R. Hines et al. [12] have proposed methods independently for migrating fast. These methods have a future that memory pages are posted by the destination host. Namely, the source host suspends a VM, and it immediately begins live migration. Then, the copy process of memory pages is executed at the destination host. We call this method *the post-copy*. This approach enables T_{ALL} to shorten in exchange for inflicting a load of the migrated VM. They aim at T_{ALL} of about 1 second. Thus, we define the requirement of fast migration as that T_{ALL} is about 1 second.

On the other hand, the global live migration by [4], [7] cannot meet this requirement. E. Harney et al. [4] have introduced that the time required for IP mobility after migrating is 8 seconds when the storage is not shared. In fact, T_{ALL} becomes more than 8 seconds in this case. They have not referred to the problem. L. Qin et al. [7] have cared about T_{DW}. They have realized stable T_{DW} by reducing a load of the host itself. In Xen, hypervisor confirms whether the MAC address of a VM overlaps. At this time, it broadcasts ARP packets to all network nodes on a local link. In HyperMIP, ARP packets are transferred to only some network nodes needed to update the ARP table. As the result, network restoration latency can reduce. Although, T_{DW} is 3.2 seconds at a minimum. Thus, T_{ALL} becomes more than 3 seconds in this case.

The proposed method is the method to provide performance nearly ordinary live migration for the global live migration. Namely, it realizes the global live migration that T_{DW} closes to 0.5 second. Also, we think a streaming service as an application of the global live migration. A VM has a special function such as the Multi point Control Unit (MCU), and data stream is transferred to all users by the VM. The global live migration will be used for relocating the VM dynamic to the suitable position based on network condition among users. Therefore, it is essential to realize T_{DW} of less than 1 second in order to improve performance of the global live migration for providing the application like a streaming.

3 PROPOSED METHOD

A future of our method is that the source host executes the network configuration before migrating. The problem, which is T_{DW} increases further by introducing IP mobility, is caused by IP mobility process is executed after migrating. In fact, the proposed method aims at reducing the process work of IP mobility after migrating. The network configuration includes two configurations. One is the MoA configuration. In MAT, the process for allocating a MoA follows four procedures. First, a MAT daemon monitors MN's interface state. Next, a MAT kernel issues the Router Solicitation (RS) when

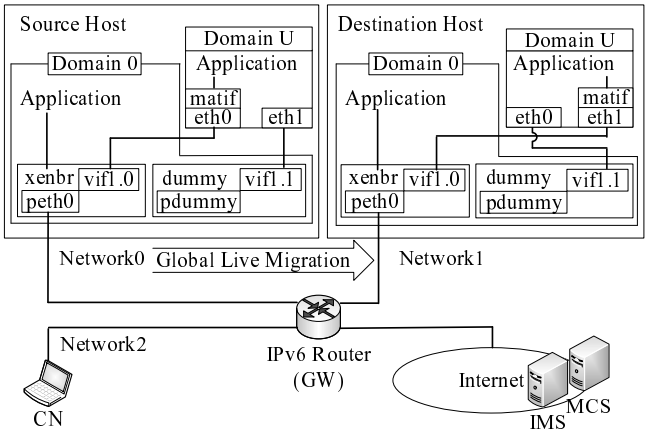

Figure 1: Virtual link composition of the global live migration.

interface's state changed by handoffing MN. Then, the daemon allocates a new MoA after getting the Router Advertisement (RA). Finally, it executes the IMS update for ensuring consistency between a new MoA and the using HoA. When an acknowledgment of the IMS update received, the MoA configuration finishes. The other is the default gateway (GW) configuration. This configuration is one of key points of our method, and it enables RAs from the GW same as the destination host to provide to a VM at the source host. In the proposed method, IP mobility process after migrating is that the destination host just switches to the interface with MoA. As the result, our method can provide the global live migration that T_{DW} closes to 0.5 second.

3.1 System Architecture

In order to prepare network environment of the migrated VM in advance, we add a special interface to a VM. Namely, the VM has dual interfaces. Figure 1 shows the virtual link composition between a VM and Hosts by the global live migration when our method is used. One is an ordinary interface, which is linked to the MAT virtual interface (matif). We call this interface the primary interface. In MAT, the matif is switched to another physical interfaces when MN handoffed. In fact, the matif enables an application on a VM to constantly use the HoA. Also, the primary interface is attached to a virtual bridge (xenbr) connected to the physical interface (peth). Therefore, to provide reachability for a VM requests to attach to the xenbr. This reachability means that a VM is able to communicate with CN through the Internet. On the other hand, the other interface is a special interface allocated a MoA by the source host. We call the interface the secondary interface. Normally, the secondary interface is attached to a virtual bridge without reachability (dummy). The dummy is only used for pretending the GW of the destination host. In our method, it is necessary to prepare pair of two bridges for a VM. As well, these interfaces are recognized as the virtual interface (vif) in a host.

We have used Xen 3.2 and MAT 2.0 as a virtualization tech-

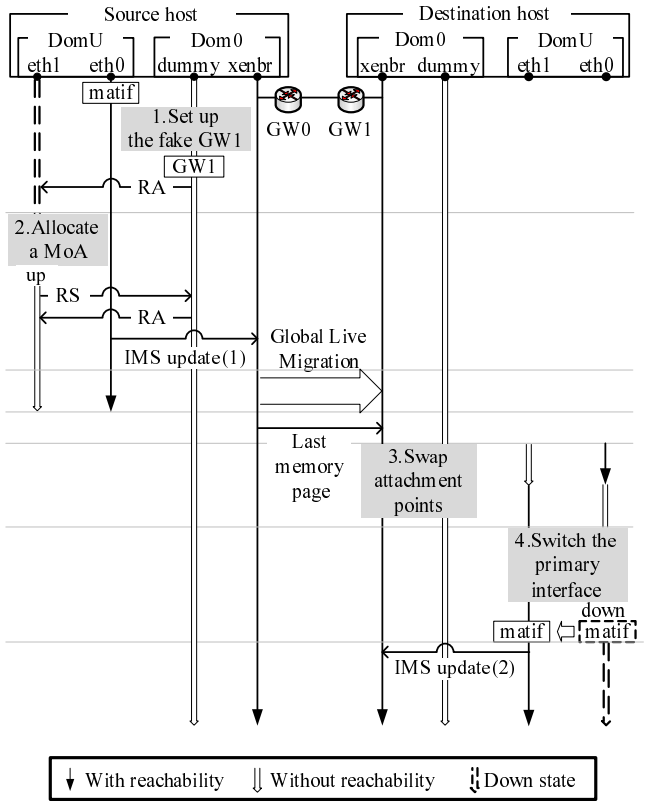

Figure 2: Time chart of the proposed mechanism.

nology and an IP mobility architecture. In Xen, the host's OS controlling a VM is called the Domain 0 (Dom0), and the VM's OS is called the Domain U (DomU). Also, we have prepared the Migration Control Server (MCS) for controlling the global live migration. The MCS manages related information needed for the network configuration. The information includes the host's IP address, the VM's domain name and the GW information connected each host. The MCS is located on the Internet, and it is recognized as the one server from hosts. In addition, it is synchronized with IMS for providing VM's IP address information.

3.2 Detailed Mechanism

There are four steps in the proposed method. The first step is the destination GW setting, the second step is the MoA allocation, the third step is the attachment points swapping and the fourth step is the primary interface switching. Figure 2 shows a time chart of the proposed mechanism. The first step is executed at the source host, and the second step is executed at a VM. The third step is executed at the destination host, and the fourth step is executed at the migrated VM. Initially, we suppose that the VM's secondary interface is down at the source host.

In the first step, the Dom0 of the source host executes the GW configuration. Firstly, the Dom0 gets the GW information of the destination host from the MCS. The information includes the IPv6 address and the link local address of the GW. Then, it sets their addresses to the dummy and it requests two demands by presenting the link local address to the DomU through the xenbr. One of two demands is the IP table updating. The other is the secondary interface up. In addition, the Dom0 keeps issuing RA to the suspending secondary interface through the dummy. We use the radvd daemon for issuing RA.

In the second step, the DomU executes the MoA configuration. Firstly, the DomU confirms the primary interface managed by the DomU itself. Next, it updates a dummy's link local address to the IP table as the GW of the secondary interface. Then, the DomU ups the secondary interface. In MAT, the MAT daemon recognizes to handoff by changing interface state. Thus, we introduced the process which ups the secondary interface forcibly to a VM for getting the MoA configuration. Finally, the DomU replays results that whether the IP table was updated and whether a MoA was allocated to the Dom0. At this point, the network configuration concludes. On the other hand, the Dom0 executes live migration if it does not receive the error message from the DomU.

In the third step, the Dom0 of the destination host ensures reachability dynamically for an application on a VM. The Dom0 swaps the attachment point of the virtual interface. Namely, it attaches the secondary interface to the xenbr after migrating. The primary interface is attached to the dummy. We extended a part of the live migration program. A VM is restored at the destination host after copying the last memory page. The restore process includes some procedures that the creation of the virtual interface and the attachment to the same bridge before migrating. We modified the latter part for swapping the attachment point of the secondary interface.

In the fourth step, the Dom0 switches the secondary interface dynamically to the primary interface. We extended the front-end device driver on a VM. The driver is used in controlling a virtual device by notifying the I/O operation to the Dom0. The extended driver enables the primary interface to down after migrating. By the result, the secondary interface switches as the primary interface since the matif handoffs to the secondary interface.

The first and the second steps do not seriously affect T_{DW}. But, these steps are possible to affect T_{ALL}. If these steps are not executed in parallel with other programs at the background, T_{ALL} will surely not reduce. Therefore, we recommend that the host ends these steps previously before executing the xm migrate command.

4 EVALUATION

We aim at reducing T_{DW} increased by introducing IP mobility. In order to evaluate the efficacy of our method quantitatively, we have measured T_{DW} and T_{ALL} by three subjects. The first subject is the global live migration with the proposed method. The second subject is the global live migration without the proposed method. The third subject is ordinary live migration. Network environments are 100 Mbps and 1000 Mbps. We have prepared five VMs, which set different virtual memory sizes: 32 MB, 64 MB, 128 MB, 256 MB and

Table 1: Specification of used machines.

	Source / Destination host	VM	CN
CPU	Core 2 Quad 2.6GHz	virtual CPU 1 core	Pentium Ⅲ 1.0GHz
RAM	4096 MB	32, 64, 128, 256, 512 MB	512 MB
OS	Debian Linux 5.0	Debian Linux 4.0	Debian Linux 4.0
kernel	2.6.26-2-xen-686	2.6.16.29 + MAT 2.0	2.6.16-5 + MAT 2.0
Hypervisor	Xen 3.2-1	-	-

Table 2: Experimental results.

RAM size (MB)	Time (second)											
	The global live migration (the VM with MAT)								Live migration			
	with the proposed method				without the proposed method							
	100 Mbps		1000 Mbps		100 Mbps		1000 Mbps		100 Mbps		1000 Mbps	
	T_{DW}	T_{ALL}	T_{DW}	T_{ALL}	T_{DW}	T_{ALL}	T_{DW}	T_{ALL}	T_{DW}	T_{ALL}	T_{DW}	T_{ALL}
32	0.63	3.68	0.57	1.38	3.26	6.71	2.67	3.59	0.37	3.36	0.45	1.02
64	0.72	6.70	0.61	1.68	2.76	9.09	2.76	3.88	0.49	6.04	0.39	1.60
128	0.70	12.45	0.62	2.20	3.56	15.68	2.93	4.80	0.47	11.73	0.37	2.35
256	0.66	23.96	0.60	3.42	2.71	26.67	2.82	5.68	0.53	23.30	0.43	3.61
512	0.74	46.97	0.66	6.20	4.87	52.72	3.05	8.43	0.40	46.07	0.53	5.90

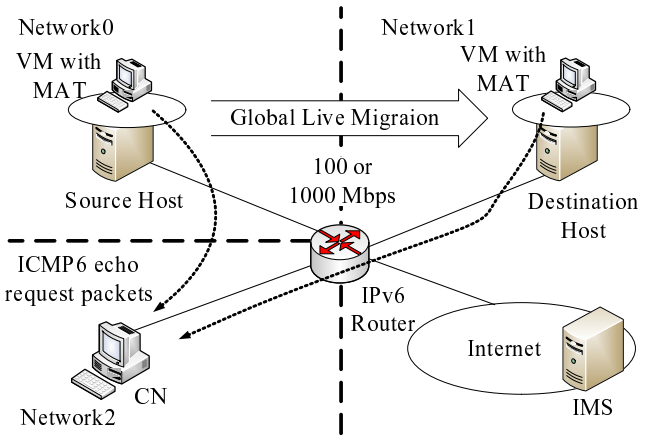

Figure 3: Experiment methodology.

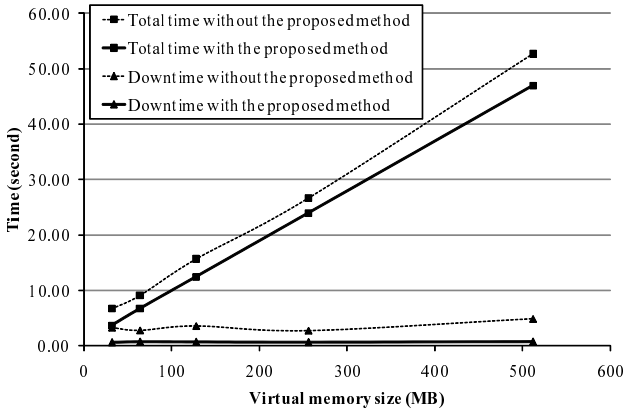

Figure 4: Experimental results in 100 Mbps environment.

512 MB. The VM's disk image is 2 GB, and each host has not been sharing the storage. Table 1 shows a specification of machines used at the experiment. We have used YAMAHA RTX1200 as the IPv6 router.

4.1 Experiment Method

We have used the ping6 command and the simple script for measuring each time. ICMP6 echo request packets, which are sent from the VM to CN at 0.2 second interval, is used to measure T_{DW}. We defined the time from ICMP6 echo request packets stopped to an ICMP6 echo request packet was recaptured on the CN as T_{DW}. The script, which is the ability to measure runtime of millisecond scale, is used to measure T_{ALL}. Figure 3 shows the experiment methodology.

4.2 Result and Consideration

Table 2 shows measured results of three subjects. Each value is average of three times. Figure 4 is the graph based on T_{DW} and T_{ALL} compared the first subject with the second subject in 100 Mbps environment.

The result of the first subject shows that T_{DW} is maintained constant in time of 0.65 second on average, and maximum T_{DW} is 0.74 second. All results of T_{DW} in Table 2 shows that T_{DW} does not affect from a virtual memory size, and variation of results includes an error of measurement.

In the second subject, the result shows that T_{DW} is between 2 seconds and 5 seconds. T_{DW} of the first subject is about 80% shorter than one in the second subject. Reduction of T_{DW} is noticed from Fig. 4. T_{ALL}, which increases in proportion to virtual memory sizes, also reduces as well as

T_{DW}. In addition, both T_{DW} and T_{ALL} closes to each result of the third subject. Therefore, these results indicate that the proposed method can provide the global live migration realizing T_{DW} of less than 1 second, and our method has efficiency enough.

The proposed method has possibility to coexist the post-copy such as [11], [12] since it needs not to extend hypervisor. If the combined method is realized, T_{ALL} of the global live migration shorten. This is a great advantage to fast migration of the global live migration. On the other hand, our method requests dual bridges for a VM. This idea affects host's resources. As the result, the proposed method is possible to worsen performance of the global live migration in some circumstances. We will verify host's limitation when our method is used.

5 CONCLUSIONS

In this paper, we proposed a solution of the global live migration's problem, which the VM suspends certainly more than a few seconds. The problem is caused by IP mobility process is executed after VM migration. In our method, the source host executes the network configuration before migrating. Namely, it sets a MoA previously to the secondary interface of a VM. In addition, we have adopted MAT as an IP mobility architecture that requires no IP tunnels. Therefore, our method can reduce T_{DW} of the global live migration without inflicting a load on various network nodes. We have implemented the prototype and we have evaluated efficacys of our method quantitatively. By experiment result, we have confirmed that T_{DW} is less than 0.74 second when the proposed method was used. The result closes to T_{DW} of ordinary live migration. Therefore, our method has efficiency enough.

In future work, we will apply the post-copy to our method, and try to reduce T_{ALL} of the global live migration. In addition, we will evaluate two points. One is limitation of host's resources when the proposed method is used. In our method, the source host uses two bridges for a VM. Thus, we need to verify limitation of the number of VMs or bridges enabled to create. The other is a measurement of T_{DW} when the application on the VM provides a streaming service. We will verify whether our method is effective to the streaming application.

ACKNOWLEDGEMENTS

This work was supported in part by Grant-in-Aid for Scientific Research (KAKENHI 19300019, 20300029, 20700066) of JSPS and the Strategic Information and Communications R&D Promotion Program (SCOPE) of the Ministry of Internal Affairs and Communications (SCOPE-regional ICT 08230 8001) of Japan.

REFERENCES

[1] VMware, Virtualization: Architectural Considerations and Other Evaluation Criteria, `http://www.vmware.com/pdf/virtualization_considerations.pdf`

[2] P. Barham, B. Dragovie, K. Fraser, S. Hand, T. Harris, A. Ho, R. Neugebauer, I. Pratt, and A. Warfield, Xen and the Art of Virtulalization, Proceedings of the ACM Symposium on Operation Systems Principles (2003).

[3] C. Clark, K. Fraser, S. Hand, J. G. Hansen, E. Jul, C. Limpach, I. Pratt, and A. Warfield, live migration of virtual machines, Proceedings of the 2nd ACM/USENIX Symposium on Networked Systems Design and Implementation(NSDI), Boston, MA (2005).

[4] E. Harney, S. Goasguen, J. Martin, M. Murphy, and M. Westall, The Efficacy of Live Virtual Machine Migrations Over the Internet, Second International Workshop on Virtualization Technology in Distributed Computing(VTDC), Reno, NV, USA (2007).

[5] D. Johnson, C. Perkins, and J. Arkko, Mobility Support in IPv6, IETF RFC 3775 (2004).

[6] F. Travostino, P. Daspit, L. Gommans, C. Jog, C. de Laat, J. Mambretti, I. Monga, B. van Oudenaarde, S. Raaghunath, and P. Wang, Seamless live migration of Virtual Machines over MAN/WAN, Future Generation , Computer System, pp. 901-907 (2006).

[7] L. Qin, H.Jinpeng, L. Jianxin, W. Tianyu, and W. Minxiong, HyperMIP: Hypervisor controlled Mobile IP for Virtual Machine live migration across Network, IEEE High Assurance Systems Engineering Symposium 2008 (HASE2008), pp. 80-88 (2008).

[8] C. Perkins, IP Mobility Support for IPv4, IETF RFC 3344 (2002).

[9] S. Gundavelli, K. Leung, V. Devarapalli, K. Chowdhury, and B. Patil, Proxy Mobile IPv6, IETF RFC 5213 (2008).

[10] R. Inayat, R. Aibara, K. Nishimura, T. Fujita, Y. Nomura, and K. Maeda, MAT: An End-to-End Mobile Communication Architecture with Seamless IP Handoff Support for the Next Generation Internet, Proceedings of Second International Conference on Human Society@Internet, pp.465-475 (2003).

[11] T. Hirofuchi, H. Nakada, S. Itoh and S. Sekiguchi, Rapid Virtual Machine Relocation Based on Delayed Memory Transfer, IPSJ SIG Notes, system software and Operating System, Vol.2009-OS-112, pp. 1-8 (2009).

[12] M. R. Hines and K. Gopalan, Post-copy based live virtual machine migration using adaptive pre-paging and dynamic self-ballooning, Proceedings of the 5th International Conference on Virtual Execution Environments, ACM Press, pp. 51-60 (2009).