IPSJ Transactions on Advanced Computing Systems Vol.5 No.2 86–98 (Mar. 2012)

Regular Paper

Reactive Cloud: Consolidating Virtual Machines with Postcopy Live Migration

Takahiro Hirofuchi[1,a] Hidemoto Nakada[1] Satoshi Itoh[1]
Satoshi Sekiguchi[1]

Received: July 19, 2011, Accepted: November 15, 2011

Abstract: Dynamic consolidation of virtual machines (VMs) through live migration is a promising technology for IaaS datacenters. VMs are dynamically packed onto fewer server nodes, thereby eliminating excessive power consumption. Existing studies on VM consolidation, however, are based on *precopy* live migration, which requires dozens of seconds to switch the execution hosts of VMs. It is difficult to optimize VM locations quickly on sudden load changes, resulting in serious violations of VM performance criteria. In this paper, we propose an advanced VM consolidation system exploiting *postcopy* live migration, which greatly alleviates performance degradation. VM locations are reactively optimized in response to ever-changing resource usage. Sudden overloading of server nodes are promptly resolved by quickly switching the execution hosts of VMs. We have developed a prototype of our consolidation system and evaluated its feasibility through experiments. We confirmed that our consolidation system achieved a higher degree of performance assurance than using precopy migration. Our micro benchmark program, designed for the metric of performance assurance, showed that performance degradation was only 12% or less, even for memory-intensive workloads, which was less than half the level of using precopy live migration. The SPECweb benchmark showed that performance degradation was approximately 10%, which was greatly alleviated from the case of using precopy live migration (21%).

Keywords: virtual machine, live migration, server consolidation

1. Introduction

Dynamic virtual machine (VM) consolidation is a promising technology for IaaS datacenters, hoping to improve *the density* of VM hosting. VMs are dynamically repacked onto the fewest possible server nodes, thereby reducing power consumption caused by use of excessive hardware resources. When VMs are idle, a consolidation system moves them onto a small number of server nodes, shutting down the rest of the server nodes. When VMs become active, the system quickly starts up sleeping server nodes and migrates some of the VMs to them, thereby assuring appropriate performance of all VMs.

Live migration is the key technology for dynamic consolidation, enabling relocation of VMs onto new server nodes without stopping guest operating systems. There are two types of live migration mechanisms, precopy and postcopy. In precopy live migration, all states of a VM are completely copied to a destination host before the execution host is switched to the destination. Updated memory pages during memory copy are iteratively copied to the destination. It takes a long time to switch the execution host of an actively-running VM, and it is hard to estimate when migration is completed. On the other hand, postcopy live migration executes memory page copies after the execution host is switched; it is possible to change the execution host in several hundred milliseconds, and the whole live migration process is completed in a determinable period.

To the best of our knowledge, all existing studies of dynamic VM consolidation are based on precopy live migration mechanisms, which are already available in widely-used virtual machine monitors (VMMs). However, we consider postcopy live migration to be more suitable for dynamic consolidation systems that can quickly migrate VMs to other server nodes when the loads of VMs are suddenly changed.

In this paper, we propose an advanced VM consolidation system exploiting postcopy live migration. It can quickly re-optimize the locations of VMs by changing the execution host of a VM in several hundred milliseconds. The system provides a great benefit for environments where the load changes of VMs are unpredictable; when the loads of VMs suddenly become high and server nodes are overloaded, postcopy live migration can remove the overloading of server nodes (i.e., the performance degradation of consolidated VMs) much more quickly than precopy.

Although postcopy migration techniques themselves have been discussed in research papers [5], [11], these implementations have not been seen in publicly-available VMMs. In our previous work [6], [7], therefore, we implemented postcopy live migration for KVM [10]. Next, in this paper, we focus on a VM consolidation system with postcopy live migration.

The contribution of this paper is that this is the first work that clarifies the effectiveness of postcopy live migration for dynamic VM consolidation. We have developed a prototype of our consol-

[1] National Institute of Advanced Industrial Science and Technology (AIST), Tsukuba, Ibaraki 305–8568, Japan
[a] t.hirofuchi@aist.go.jp

© 2012 Information Processing Society of Japan

idation system to evaluate the effectiveness through experiments on our server cluster.

Section 2 highlights existing problems in our target environments. Section 3 presents our reactive VM consolidation system. Section 4 discusses its evaluation. Section 5 describes related work. Finally, Section 6 concludes this paper [*1].

2. Background

The target environment of our consolidation system is a commercial IaaS datacenter where customers run various kinds of workloads on their VMs. The datacenter presents performance criteria for VMs. One example is Amazon EC2; it says the CPU processing speed of Small Instance is equivalent to a 1.0–1.2 GHz 2007 Opteron. Although these criteria are casual and not defined in service level agreements, service providers need to achieve them as much as possible.

The challenge of our dynamic VM consolidation is to achieve power saving and performance guarantees. If dynamic VM consolidation is introduced into an IaaS datacenter like Amazon EC2, customers' VMs, which have performance criteria like "1.0 GHz Opteron," are dynamically packed onto the fewest possible server nodes, depending on actual resource usage. When the VMs become idle, they are relocated onto a small number of server nodes. By putting unused server nodes into the hardware sleep mode (ACPI S3), excessive power use of the datacenter is eliminated. On the other hand, when the VMs become active, they are relocated onto resumed server nodes, thereby guaranteeing promised VM performance criteria.

2.1 Limitations of Precopy Live Migration

Prior studies regarding VM consolidation are based on precopy live migration. We feel, however, that precopy live migration is not suitable for our consolidation system, which puts primary importance on performance guarantees.

Precopy live migration is employed in widely-used VMMs (e.g., Xen, KVM, and VMware). It reconstructs a VM's memory image at a destination host **before** switching its execution node [3], [12], [13]. After live migration is initiated, this basically works as follows.

1; Start dirty page logging at a source host. This mechanism detects updates of memory pages during the following memory copy steps. **2;** Copy all memory pages to the destination. Since the VM is running at the source host, memory pages are being updated during this period. **3;** Copy dirtied memory pages to the destination again. Repeat this step until the number of remaining memory pages is small enough. **4;** Stop the VM at the source. Copy the content of virtual CPU registers, the states of devices, and the rest of the memory pages. **5;** Resume the VM at the destination host.

At the second step, all memory pages are transferred to the destination, which means that migration time basically increases in proportion to the memory size of the VM. Moreover, at the third step, dirtied pages must be iteratively copied to the destination. If the VM is intensively accessing large amounts of memory, numerous dirty pages are created and transferred continuously. In the worst case, live migration is never completed; i.e., a workload dirties VM memory faster than network bandwidth can accommodate.

Prior studies, therefore, employed a load prediction technique to alleviate this migration overhead; before a target system becomes fully saturated, a consolidation system rebalances VMs to remove the potential risk of overloading proactively.

However, we feel this approach is not feasible for the target environments that our consolidation system is focusing on, where the prediction of VM loads is not effective.

First, it is difficult to predict the future demand for processing power precisely without workload-specific information [16]. In the real world, an IaaS datacenter allows users to run any workload in their VMs, and it is difficult for the consolidation system to support all types of workloads with load prediction. In addition, from the viewpoint of security and privacy concerns, there should exist isolated administrative domains between the inside and outside of a VM; the consolidation system should be independent of unreliable information that can be maliciously altered on the inside of a VM.

Second, to get the maximum power saving with dynamic consolidation, the system should optimize VM locations frequently as much as possible. In our research project, we aim to establish more fine-grained, aggressive optimization at the level of every second, not in daily and weekly cycles. Existing load prediction studies focusing on longer time spans (e.g., the load will increase during business hours) are not applicable to this short time span. The system is required to optimize VM locations reactively for load changes.

3. Reactive Consolidation System with Postcopy Live Migration

First, postcopy live migration is briefly summarized here to bring readers up to speed. Next, the design and implementation of our consolidation system is presented.

3.1 Postcopy Live Migration

In our previous work [6], we developed a postcopy live migration mechanism for KVM. In contrast with precopy migration, memory pages are transferred **after** a VM is resumed at a destination host. The key to postcopy migration is an on-demand memory transfer mechanism, which traps the first access to a memory page at the destination and copies its content from a source host. Postcopy migration basically works as follows:

1; Stop the VM at the source host. Copy the content of virtual CPU registers and the states of devices to the destination. **2;** Resume the VM at the destination without any memory content. **3;** If the VM touches a not-yet-transferred memory page, stop the VM temporarily. Copy the content of the memory page from the

[*1] This work is based on an earlier work: Reactive Consolidation of Virtual Machines Enabled by Postcopy Live Migration, in Proceedings of the 5th International Workshop on Virtualization Technologies in Distributed Computing, (C) ACM, 2011. http://doi.acm.org/10.1145/1996121.1996125. In this paper, we extend the earlier work to include more detailed discussions; especially the section of evaluation is extended with detailed descriptions and new experiments. We updated overall the paper to improve readability and presentation. This paper is a substantially-developed, new derivative work, which adheres to the submission rules.

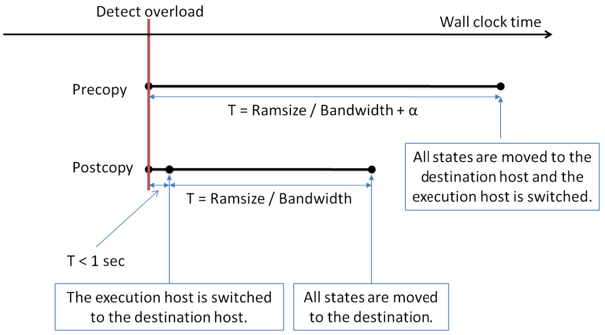
Detect overload
Wall clock time
Precopy
T = Ramsize / Bandwidth + α
All states are moved to the destination host and the execution host is switched.
Postcopy
T = Ramsize / Bandwidth
T < 1 sec
The execution host is switched to the destination host.
All states are moved to the destination.

Fig. 1 Comparison between precopy and postcopy live migration for VM consolidation.

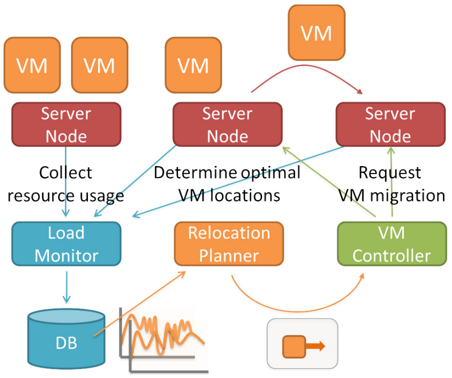

Fig. 2 System components.

source. Then, resume the VM.

The third step is repeated until all memory pages are transferred to the destination. In addition, in parallel with the on-demand page retrievals, a background copy mechanism works to make bulk copies of not-yet-transferred pages. Because on-demand page copy may not cover all ranges of VM memory in a short period of time, the background copy mechanism gets rid of dependency on a source host as soon as possible. The background copy mechanism analyzes important memory areas with page fault statistics, and starts to deal with hot-spot memory pages for current VM workloads. On-demand memory page retrievals over a network are reduced by this mechanism.

Figure 1 summarizes the comparison between precopy and postcopy live migration. For dynamic VM consolidation, the advantage of postcopy migration is that the system can react to sudden load changes of VMs more quickly than precopy; especially, some of the VMs on a server node can be quickly moved to other hosts, when their loads suddenly become high and the server node gets overloaded. Strictly speaking, the execution hosts of the VMs are quickly switched to other hosts (in less than one second), and then the rest of the memory pages are transferred in a fixed period of time.

As in Fig. 1, with precopy live migration, the performance criteria of the VMs on a saturated node are not satisfied during $Ramsize/Bandwidth + \alpha$ seconds; α depends on the memory update speed of the guest operating system. On the other hand, with postcopy live migration, the performance of VMs is recovered in $Ramsize/Bandwidth$ seconds, even in the worst case. In normal cases, performance degradation ends in a shorter period time, because the working set of memory pages for running workloads are limited and transferred in a high priority manner. In the best case, such as pure CPU-intensive workloads, all the VMs achieve their performance criteria without visible degradation, even though some of the VMs are migrated to other server nodes.

3.2 System Components

The overall design of our consolidation system is illustrated in **Fig. 2**, which is composed of Load Monitor, Relocation Planner, and VM Controller modules. Load Monitor collects resource usage data every one second and put it into a database. Relocation Planner periodically calculates optimal locations for VMs from the latest resource usage histories in the database. VM Controller requests live migration to server nodes according to the results from Relocation Planner.

Load Monitor collects resource usage data from each server node, such as CPU usage, network I/O, and disk I/O of both the server node and the VMs running on it. This information is retrieved from `/proc/` of the host Linux operating system and the monitor interface of QEMU/KVM. At one second intervals, a daemon program on each server node scans resource usage information and sends data to Load Monitor. Load Monitor stores the received data to an SQLite database as soon as possible. The consolidation system requires the latest resource usage to quickly respond sudden load changes. Considering the Linux kernel basically updates the contents of `/proc/` at every second, we decided to collect resource usage at 1-second intervals. The CPU overhead of the resource monitor daemon on each host is negligible (i.e., lower than 1–2%) in our experimental environments.

Relocation Planner retrieves resource usage histories from the database, determines whether a server node is overloaded or not, and calculates a relocation plan. We carefully designed this component to be independent from the others, so that it is possible to implement various consolidation algorithms. The current strategy of consolidation planning is explained in Section 3.3.

VM Controller executes live migration according to the relocation plan. We use XML-RPC to control VMs on server nodes remotely; three request messages (e.g., `CREATE_VM`, `MIGRATE_VM`, and `DESTROY_VM`) are defined to create, migrate, and destroy the requested VM. On each server node, there is a server daemon handling these XML-RPC requests.

VM Controller also executes the suspend/resume of server nodes. When there is no VM on a server node, VM Controller puts the server node into the sleep mode (ACPI S3 state). When a relocation plan indicates the suspended node needs to be woken up, VM Controller requests the resume of the node via the Intel AMT (Active Management Technology) web service interface. AMT is an out-of-band hardware management system working in the BIOS level, which is more reliable and powerful than Wake-On-LAN, and less expensive than other out-of-band management systems.

3.3 Consolidation Strategy

Our consolidation system tries its best to achieve guaranteed performance for VMs. In this paper, the system assigns **one physical CPU core and 1.7 GB memory** to each VM, which is the performance criterion presented to users. We defined this

© 2012 Information Processing Society of Japan

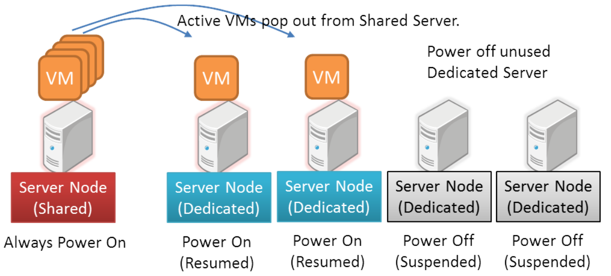

Fig. 3 Server node types (Shared Server and Dedicated Server).

criterion considering an IaaS service in the real world; a Small Instance of Amazon EC2 has one virtual CPU core and 1.7 GB memory. The service claims the virtual CPU core is equivalent to a 1.0–1.2 GHz 2007 Opteron, which supposedly corresponds to one physical CPU core of a physical machine. Compared to EC2, in our dynamic consolidation system, when a VM is consuming a small part of the assigned CPU resource, the VM may share a physical CPU core with other VMs through dynamic relocation. As far as we know, Amazon EC2 and other providers do not employ this kind of dynamic consolidation.

3.3.1 Server Nodes

As illustrated in **Fig. 3**, our consolidation system introduces two types of server node to a target datacenter, Shared Servers and Dedicated Servers. VMs are migrated between Shared Servers and Dedicated Servers, depending on their resource usage. Both types of servers have the same hardware, except that a Shared Server has a large amount of physical memory. The memory size is sufficient to host many idle VMs; however, the number of physical CPU cores is not sufficient to exclusively assign one physical CPU core to each VM on the server. Basically, the system tries to put as many VMs as possible on Shared Servers, as far as performance criteria of VMs are being satisfied; unused Dedicated Servers are suspended. When more physical resources are required to meet performance criteria, the system resumes a Dedicated Server and uses it to host VMs. When a VM is running on a Dedicated Server, it uses one physical CPU core exclusively; the number of VMs hosted on a Dedicated Server is limited by the number of CPU cores on it. A Dedicated Server does not need a large amount of memory like a Shared Server.

One of the reasons behind this design choice is that our consolidation system does not exploit dynamic memory ballooning. It allocates a fixed memory size to each VM, never resizing it. Although dynamic memory ballooning is supported by VMMs, it requires coordination of guest operating systems, which does not match isolated administrative domains between the outside and inside of a VM, as noted in Section 2. Therefore, the system requires special server nodes with large memory to consolidate VMs. The other reason is based on our assumption that most workloads on VMs run in a ‘race-to-halt’ manner. A new job is processed with all the available resources to finish it as soon as possible. Allocating dedicated resources to an active VM is a realistic design choice.

3.3.2 Packing Algorithm

The packing algorithm of our consolidation system is required to calculate optimal locations in the shortest time. The system must reactively optimize VM locations as soon as possible when the loads of VMs suddenly change. We therefore employ the following heuristic algorithm that roughly optimizes VM locations without explicitly solving bin-packing problems.

It should be noted that the algorithm is currently based on only CPU usage statistics, not including disk and network I/O data. At the time this paper is being written, KVM does not support live migration for paravirtualized devices, such as VirtIO Block Device and VirtIO Network Device. All the VMs on our consolidation system must use fully-virtualized devices incurring relatively high CPU overheads.

Basically, VM relocation is performed as soon as possible when one of the following events happens:

- 1) The system detects that a Shared Server is overload, and finds a destination Dedicated Server.
- 2) The system detects that a VM on a Dedicated Server is idle (the CPU load of the VM is under the below Return threshold), and find a destination Shared Server.

It should be noted that these events asynchronously happen. For each Shared Server, Relocation Planner launches an overloading detector thread that performs the events of the type 1). For each VM on a Dedicated Server, Relocation Planner launches an idling detector thread that performs the events of the type 2). All overloading/idling detector threads asynchronously check the latest resource usage by querying the database, and trigger VM relocation if needed.

First, all the VMs are launched at one of the Shared Servers, and then the following steps are iterated by overloading/idling detector threads.

Overloading Detection: When the latest CPU load average of the Shared Server reaches 90% (i.e., is regarded as overloaded), the most CPU-consuming VM is migrated to a Dedicated Server. Because usage statistics are measured in outside of the VM, it is difficult to determine what amount of a CPU resource is actually required. Therefore, simply, we pick up the VM that is probably in a ‘race-to-halt’ state. The target Dedicated Server is chosen from power-on Dedicated Servers, in which the Dedicated Servers with the fewest empty cores (but not zero) are the first in priority to be chosen. If not available, one is selected from the sleeping Dedicated Servers. In this case, the Dedicated Server is resumed.

Idling Detection: The system does not move the migrated VM for at least 20 seconds after the last migration ends, in order to avoid overreaction. After that, the idling detector thread of the VM is started to periodically check whether the latest CPU load average of the VM is under the Return threshold value (50%). If the load average is under the threshold, the monitoring daemon tries to move the VM back to one of the Shared Servers; it tries to find the Shared Server that has sufficient CPU and memory resources for the VM. An *admission ticket* to a Shared Server is given to the VM on a ‘first come, first served’ basis, in order to serialize migrations to the Shared Server. If a Shared Server with sufficient resources is found, the VM is migrated to it. Otherwise, the VM remains at the Dedicated Server; the daemon pauses at one second intervals and tries the above steps again.

This packing algorithm is intended to be sufficient and lightweight, allowing rapid re-optimization in response to sudden load

© 2012 Information Processing Society of Japan

changes. If the loads of multiple VMs increase simultaneously in a shared server, the system tries to pop out these VMs from the shared server in a one-by-one manner. Obviously, the second popping-out VM is adversely affected. The packing algorithm does not invoke simultaneous migrations from/to a shared server. If multiple live migrations share one network path, there is a higher possibility that precopy live migration does not finish. Postcopy live migration may involve serious performance degradation. Although both the one-by-one and simultaneous approaches have such trade-off, we chose the former approach for the first prototype, which allowed the system design to be much simpler.

4. Evaluation

First, experiments using micro benchmarks are performed to identify basic characteristics of the proposed system. The consolidation system using postcopy live migration is compared with the one using precopy live migration. The effectiveness of the proposed system under various memory update intensities is discussed. Next, experiments with compound load change scenarios are conducted to evaluate the proposed system's feasibility for long-term datacenter operations. We also use the SPECweb benchmark to evaluate our consolidation system for more realistic applications.

Figure 4 shows our experimental environment. Both the public and private network segments are connected to each server node. The network interface of a VM is bridged to the public network, so that a web server in the VM is accessible through the Internet. The private network, which is isolated from the public network, is used to manage VMs on the datacenter side. The management node and storage nodes are connected to the private network segment. In addition, a dedicated network segment for live migration is added, in order to isolate bursty migration traffic from other management traffic. If the private network is used for live migration, the disk I/O traffic of the VMs will be adversely affected, directly resulting in performance degradation of all VMs.

In the following experiments, all server nodes have a 2-core processor (Intel Core2 Duo E6305) and 20 GB memory; one server node is used as a Server Node, and the other five nodes are used as Dedicated Servers. All VMs are composed of one virtual CPU core and 1.7 GB of memory. The first physical CPU core of each server node is dedicated for VMs; all VMs are pinned at the first core by using the `taskset` command. If there are several VMs on a Shared Server, they share the first CPU core. The second CPU core is used to run control and monitoring daemons. The control of VMs (e.g., create, migrate, and destroy operations) and the monitoring of resource usage should be performed promptly anytime, not affected by the activity of VMs.

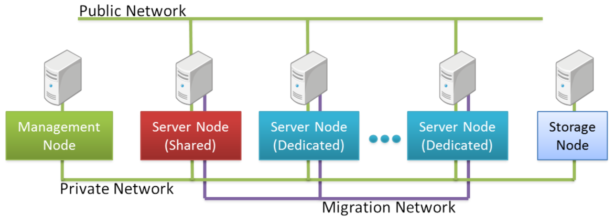

Fig. 4 Experimental environment.

Due to page limitations, this paper focuses on performance assurance, not power savings. Our work-in-progress paper [8] summarizes the preliminary results of power savings in another cluster: in the ACPI S3 state, a server node consumes only 8 W (10% of the power-on mode); a server node was resumed in approximately 5 seconds from the S3 state and became acceptable for migrating VMs. In the following experiments, Dedicated Servers were not actually suspended/resumed, although the number of in-use Dedicated Servers was tracked.

4.1 Micro Benchmarks

Because the motivation for our consolidation system is to assure the best VM performance possible, we have developed a new benchmark program that shows performance degradation from specified criteria. The metric of performance assurance is the number of achieved operations per second against that of target operations per second.

The benchmark program, executed in a guest operating system, can create any specified CPU load. A target CPU load is generated by interlacing short busy loops and sleeps. In addition, the benchmark program allows specifying any memory update intensity; this will significantly affect performance of migrating VMs.

4.1.1 Benchmark Detail

Figure 5 illustrates how the benchmark program works. The benchmark program iterates a short calculation operation; the time of one operation is composed of T_{cpu} (the time for busy CPU loops), T_{mem} (the time for memory touches), and T_{sleep} (the time for sleep). During T_{cpu}, the benchmark program iterates a busy loop C times. During T_{mem}, the benchmark program touches $C\alpha$ memory pages (i.e., write 1 bytes data at the first byte of each memory page); the α is the parameter of memory update intensity. It should be noted that these memory touches consume CPU resources. During T_{cpu} and T_{mem}, the CPU of the VM is busy. We assume memory update speeds of workloads are proportional to their CPU load.

The benchmark program automatically determines the appropriate value of C that makes a specified target CPU load with a specified parameter of memory update intensity. Given S_{cpu} (the number of busy loops per second that the VM can achieve) and S_{mem} (the number of memory page touches per second that the VM can achieve), which are measured before the execution of benchmarks, we get the following equations regarding T_{cpu} and T_{mem}:

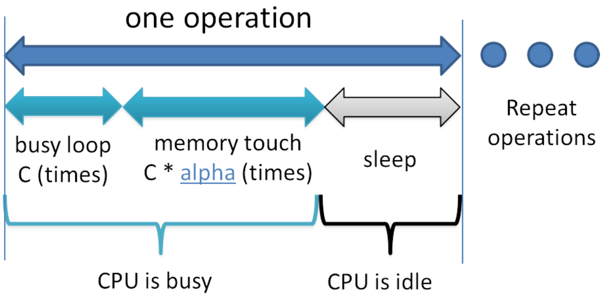

Fig. 5 The overview of the micro benchmark program.

$$T_{cpu} = \frac{C}{S_{cpu}} \quad (1)$$

$$T_{mem} = \frac{C\alpha}{S_{mem}} \quad (2)$$

In experiments, we use $T_{sleep} = 10$ (ms), considering the clock accuracy in guest operating systems. The target CPU load L, specified by users of the benchmark, is given as follows:

$$L = \frac{T_{cpu} + T_{mem}}{T_{cpu} + T_{mem} + T_{sleep}} \quad (3)$$

Through the Eqs. (1), (2), (3), the benchmark program determines the value of C for the target CPU load and memory update intensity.

The benchmark program outputs the results of performance assurance every second; the number of achieved operations per second is measured by the benchmark program, and that of target operations per second is calculated from $1/(T_{cpu} + T_{mem} + T_{sleep})$ in advance. The difference between these values, the number of failed operations per second, shows the degree of performance degradation incurred by dynamic consolidation. In the following experiments, we use this value as the metric of performance assurance.

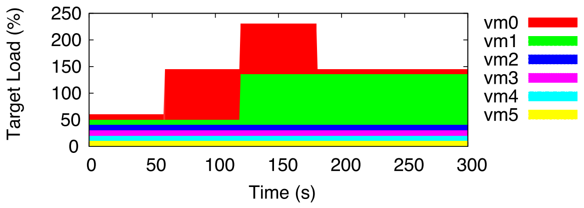

Fig. 6 The target CPU load change scenario of the micro benchmark program.

4.1.2 Benchmark Settings

The benchmark program allocated 1 GB of memory and then accessed it at a specified speed. The results of the benchmark are time-series data, which show actual achieved operations per second against the target operation per second that should be achieved.

In the following experiments in Section 4.1.3 and Section 4.1.4, the target CPU loads that the benchmark program tried to generate are shown in **Fig. 6**. First, all 6 VMs are loaded with 10% of a CPU core capacity, respectively. All VMs are running at the Shared Server. At 60 seconds, the target CPU load of VM0 is increased to 95%, which means the Shared Server is saturated, and VM0 will be removed from it. At 120 seconds, the target CPU load of VM1 is set to 95%. At 180 seconds, the target CPU load of VM0 is set back to 10%, again. The consolidation system automatically re-optimizes VM locations in response to the CPU usage of the VMs.

4.1.3 Pure CPU-intensive Workload

For the first case, we ran the benchmark program with no arbitrary memory update. The generated workload was nearly pure CPU-intensive.

As shown in **Fig. 7**, the consolidation system using postcopy live migration worked successfully. At 65 seconds (i.e., 5 seconds after the CPU load of VM0 became high), the consolidation system detected the overload of the Shared Server, and switched the execution host of VM0 to Dedicated Server 4. In the same manner, VM1 was switched to Dedicated Server 5 at 125 seconds. By using postcopy migration, the execution hosts of VM0 and VM1 were changed 5 seconds after the overloads were detected.

Although the consolidation system using precopy live migra-

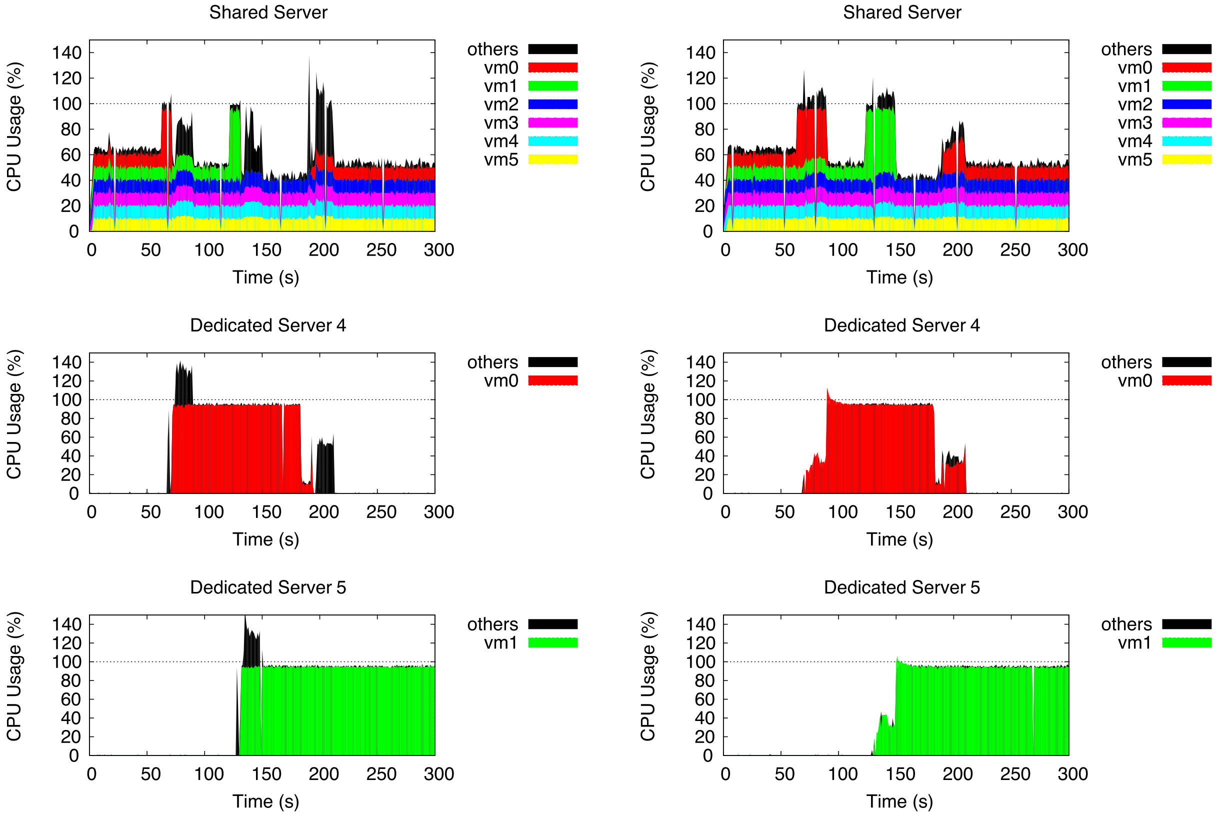

Fig. 7 The CPU usage of server nodes and VMs (left: using *postcopy* live migration, right: *precopy*).

© 2012 Information Processing Society of Japan

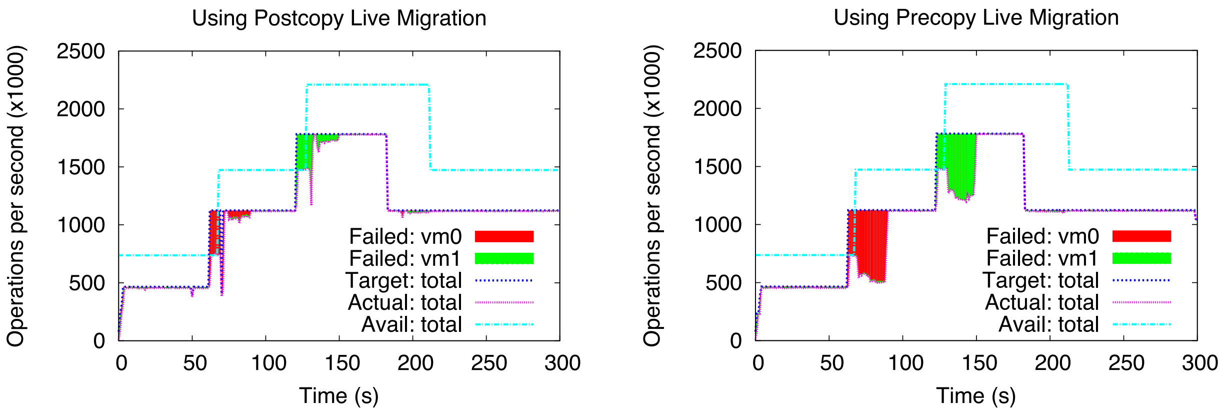

Fig. 8 The comparison of the benchmark results between the cases of using Postcopy and Precopy migrations, including operations achieved per second (Actual:total), target operations per second (Target:total), failed operations per second of each VM (Failed:VM0 and Failed:VM1, colored area), and maximum achievable operations per second estimated from the number of running server nodes (Avail:total). (No memory update).

tion basically worked, the execution host of VM0 was switched at approximately 85 seconds and that of VM1 was done at approximately 145 seconds, both of which were 20 seconds later than using postcopy. During these 20 seconds, VM0 and VM1 were stuck at the Shared Server, resulting in performance degradation.

In order to discuss performance assurance of VMs in more detail, **Fig. 8** shows the total performance degradation of VMs, compared with the target performance that should be achieved.

Actual:total is the total of the operations achieved per second, summing up those of all VMs. Since we increased the target CPU loads of VM0 and VM1 at different times, the total target operations per second (**Target:total**) stepped up twice, correspondingly.

As shown in the colored areas of Fig. 8 (**Failed:VM0** and **Failed:VM1**), however, there are failed operations between Target:total and Actual:total, resulting in violation of the performance guarantee. The first reason for these failed operations is that the consolidation system took 5 seconds to detect overloading of the Shared Server. **Avail:total** shows the maximum achievable operations per second, estimated from the number of running server nodes. This value means how much physical resource is available to perform operations. In this experimental environment, when the system uses one server node, the benchmark program can achieve up to 750 operations per second with the node. In Fig. 8, when the consolidation system started using a new server node, Avail:total stepped up.

The higher parts of the colored areas, above those of Avail:total, correspond to this detection overhead. Until the system detected overloading and started using a new server node, there was not enough physical resource to achieve the target operations. Because our consolidation system works reactively, without any load prediction, it is impossible to avoid these failed operations. The number of failed operations in this part, which is out-of-scope for further discussion here, does not depend on live migration mechanisms.

The other part of the failed operations, focused on in this paper, can be alleviated by postcopy live migration. As shown in Fig. 8, the consolidation system using precopy live migration involved serious performance degradation (i.e., approximately 50% down over 20 seconds) before the execution hosts of VM0 and VM1 were switched. Around 75 seconds, for example, CPU resources that could achieve 1,200 operations per second were required. Although Dedicated Server 4 was already resumed, all VMs were still running on the Shared Server, due to the slow efficacy of precopy live migration; only the Shared Server could provide 750 operations per second at its maximum capacity. In addition, the dirty page tracking and iterative page transfer of precopy migration incurred a CPU overhead penalty of approximately 250 operations per second.

On the other hand, postcopy live migration greatly alleviated this degradation (i.e., approximately 10% down over 20 seconds). In these experiments with pure CPU-intensive workloads, the substantial memory footprint of the benchmark program was approximately 3 Mbytes, which was copied to the destination within one second. After that, on-demand page retrieval from the source did not occur. The small number of failed operations around 75 and 140 seconds were caused by the CPU overhead of the page access detection. The down spikes of failed operations at about 70 and 135 seconds were caused by the cache hit miss of the current working set of memory pages. At approximately 70 (or 135) seconds, the execution host of VM0 was switched to the destination host by using postcopy migration. Just after this moment, there were no transferred memory pages at the destination. The performance of the workload was momentarily degraded until its working set of memory pages was transferred to the destination.

As shown in Fig. 7, the postcopy case involved the higher CPU utilization of other processes than VMs. Our current implementation of postcopy live migration uses the special daemon process that transfers memory pages. After the execution host is switched, this process consumes CPU resource until all memory pages are transferred to the destination host. Because postcopy migration involves bursty network traffic in a short period of time, the temporal CPU utilization sometimes becomes higher than precopy migration.

A pure CPU-intensive workload is considered an ideal case for postcopy live migration. The next experiments focus on memory-updating workloads.

© 2012 Information Processing Society of Japan

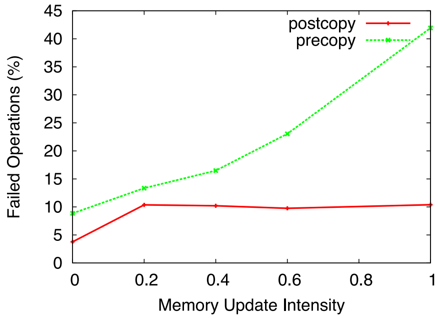

Fig. 9 Failed operations with different memory update intensity.

4.1.4 Workloads with Memory Updates

We repeated the previous experiments with different parameters of memory update. **Figure 9** summarizes the percentages of failed operations during these experiments. Memory update intensity means how many memory pages are updated during one calculation operation. See Section 4.1.1 for the detailed definition of memory update intensity. In the experimental environment, for example, when the memory update intensity α was set to 1.0, the memory update speed at a 95% CPU load reached approximately 1 GB/s. The memory update speed is over the bandwidth of Gb Ethernet.

Postcopy live migration enabled the consolidation system to mitigate failed operations, which were always under 10% for any memory update intensity. Note this 10% includes detection overhead (approximately 3%). In the precopy migration case, the percentages increased for the higher intensity parameters, resulting in a serious violation of performance criteria.

In precopy live migration, the elapsed time until migration is completed increases dramatically for memory update intensive workloads. In the experiments, the consolidation system using precopy live migration could not quickly move VM0 and VM1 to Dedicated Servers. In the case where the memory update intensity was 0.6, the migrations of VM0 and VM1 took approximately 40 and 50 seconds, respectively; however, with postcopy live migration, they took approximately 20 seconds at anytime.

In theory, the network bandwidth used for precopy live migration must be larger than the memory update speed of a migrating VM, thereby completing all state transfer in a finite time. Otherwise, live migration never finishes. In the real world, a migrating VM suffers high CPU overheads due to shadow paging and dirty page tracking. Since this overhead reduces the memory update speed of the VM, the VM is finally moved to the destination in most cases. As shown in the results, however, the consolidation system using precopy live migrations incurs serious performance penalties due to large migration time and high migration overheads, especially for memory intensive workloads. The performance penalty of postcopy-based consolidation is less serious and much more determinable for any kind of workload.

4.1.5 Compound Load Change Scenarios

We evaluated the consolidation system with other load change scenarios than the above. The following experiments are intended to show whether postcopy migration has advantages for more realistic scenarios, and also to focus the system characteristics from long-term viewpoint. Before the experiments, we randomly generated 5 one-hour scenarios with the following rules, considering *race-to-halt*-like workloads. An active VM consumes a random CPU load between 80% and 100%, and a non-active one consumes a random load between 0% and 30%. The state of a VM changes to active or non-active at 20% and 80% probabilities, respectively. A new state continues for a random duration between 60 and 300 seconds. The memory update intensity of workloads is set to 0.6[*2].

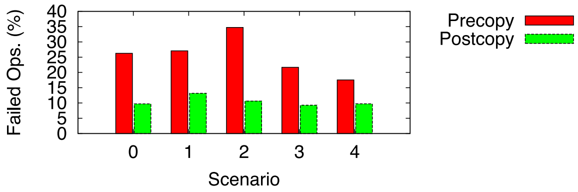

Fig. 10 The percentages of failed operations during each scenario.

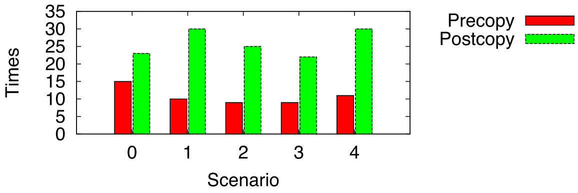

Fig. 11 The number of live migrations during each scenario.

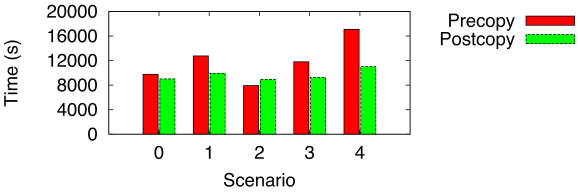

Fig. 12 The accumulated usage time of server nodes during each scenario.

As shown in **Fig. 10**, in any scenario, the consolidation system using postcopy live migration mitigated failed operations. The percentage of failed operations averages approximately 12%. In the precopy cases, they are between 18% and 35%. In **Fig. 11**, the number of live migrations performed during each one-hour scenarios is summarized. By using postcopy live migration, the consolidation system changed the locations of VMs more often; 22 to 30 times in the postcopy cases, and 8 to 15 times in the precopy cases. These results mean that postcopy live migration enables the consolidation system to optimize VM locations more aggressively than precopy.

Figure 12 shows the accumulated usage time of server nodes; we counted up how long each server node is active (i.e., hosting one VM or more), and then summed up the usage time of each server node. The accumulated usage time of server nodes was 10% shorter in most postcopy-based experiments.

Although we have not yet integrated the ACPI S3 feature into our consolidation system, we discuss energy use through approximate estimations. We measured the energy consumption of the server node and the network switch that were used in the experiments. We also measured the energy consumption of a live migration.

[*2] **Figure 14** shows the overview of the Scenario 0 workload. In Section 4.1.5, we use our micro benchmark program, not SPECweb. For Section 4.1.5, please interpret 350 sessions as 100% CPU usage.

© 2012 Information Processing Society of Japan

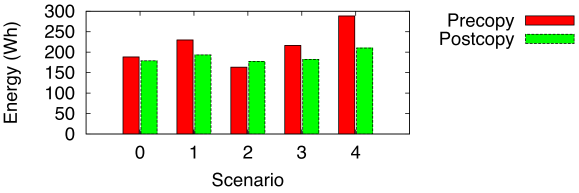

Fig. 13 The estimated energy use of server nodes during each scenario.

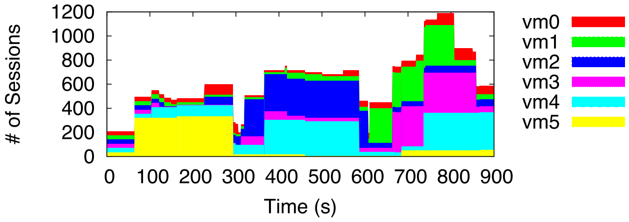

Fig. 14 The number of concurrent sessions generated by the load change scenario in the SPECweb experiments. 350 sessions correspond to 100% CPU usage.

- The server node, running at its full CPU capacity, consumed approximately 100 W. The idle server consumed approximately 53 W; note the power saving feature (i.e., ACPI C state and DVFS) was enabled. The server node in the suspend state consumed approximately 7 W.
- We performed a precopy live migration of an idle VM with the same configuration as the experiments. It took approximately 50 seconds to be completed. During the migration, the power consumption of a source/destination node increased only 3 W or less, respectively. The energy consumption of the network switch did not show a visible increase; the network switch kept consuming approximately 18 W through all the experiments. Through these values, we roughly estimated the energy use of one live migration as 0.07 Wh.

The actual energy overhead of a live migration will not be the same. For example, this overhead will depend on the type of migrations and workload behaviors. However, we emphasize that *the energy use of a live migration is far less than the power saving gain of making a server node to the suspend state*.

To make this point clear, we roughly estimated the energy use of server nodes during each scenario, which is calculated from the CPU utilization log of server nodes, the number of live migrations, and the above power consumption values. At the time of t, the power consumption of a running server node i is roughly estimated to be $p_{it} = 53 + (100 - 53) * L_{it}$ (W) where L_{it} is the CPU load of the server node [*3]. The power consumption of a not-in-use server node is to be $p_{it} = 7$ (W), where we assumed the server node is suspended. We estimated the total energy use of each scenario to be $P = (\sum_i \sum_t p_{it}) + 0.07 * M$, where M is the number of live migrations shown in Fig. 11.

Figure 13 shows the estimated energy use of each scenario in the precopy/postcopy cases. In most scenarios, the consolidation system using postcopy live migration consumed approximately the same or less energy use compared to that of using precopy live migration. In our experiments, we confirmed that frequent live migrations contributed to reducing performance degradation and did not adversely affect the energy use of the consolidation system.

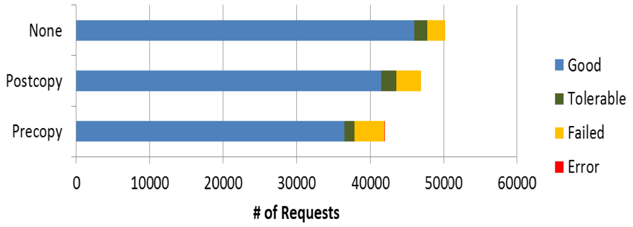

Fig. 15 The total number of performed requests in the SPECweb experiments. In the case of using precopy migration, there are 50 Error responses.

4.2 Application Benchmark

Next, we evaluated the feasibility of our consolidation system by using an application benchmark. In this subsection, we used the E-Commerce benchmark of SPECweb2005, which emulates a busy on-line shopping web server. In each VM, we set up the server side program of the SPECweb benchmark, individually; i.e., each VM hosts a different on-line shopping site. We added a client node in the public network of our experimental environment (See Fig. 4), in which we set up the client side program of SPECweb.

The client program generates the different number of simultaneous sessions to each web server, respectively. The number of simultaneous sessions is changed as shown Fig. 14. We used the same load change scenario as Scenario 0 in Section 4.1.5. In experiments, when the load of a VM is set to 100%, the client program generates 350 sessions on the VM.

Figure 15 shows the total number of performed requests during the first 900 seconds, which sums up the number of requests performed with each web server. It also shows user experiences estimated by the benchmark; if a state is Failed, a user will leave the on-line shopping site without doing the shopping, because he/she feels the web site hangs up. In the case of None (no consolidation), all VMs are statically distributed on different server nodes and never consolidated. In this case, approximately 50,000 requests were performed and most responses were in the Good state. In the case of using precopy migration, however, only 42,000 requests were performed. It should be noted that the SPECweb client program generates new requests as conforming to the specified number of concurrent sessions. If a target web server is heavily loaded, then each session takes longer time to be processed; in such situations, a client program can perform fewer requests through a period of time. In the case of using precopy, the total number of Good responses was reduced by 21% in comparison with that of the "no consolidation" case. In addition, there were 50 Error responses, which mean some TCP connections were unexpectedly closed due to the lack of resource. On the other hand, the consolidation system using postcopy live migration successfully alleviated performance degradation; the number of Good responses was reduced only by 10%, and there were no Error responses.

Figure 16 shows the time-series behavior of the benchmark

[*3] We assumed that the power consumption is linearly proportional to the CPU usage. In the real world, the model of the power consumption is more complex. We consider, however, this rough estimation is sufficient to discuss the energy overhead of live migrations here.

© 2012 Information Processing Society of Japan

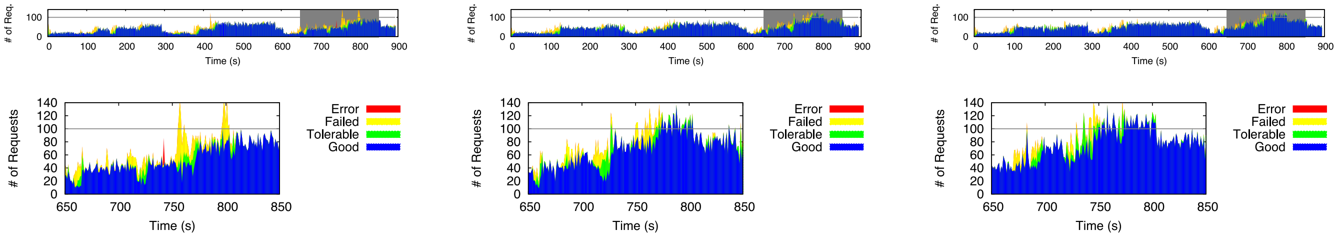

Fig. 16 The QoS score of the SPECweb experiments. Left: using precopy live migration, Center: using postcopy, Right: no live migration. The upper graphs show the high-level view of overall results. The below graphs show details between 650 and 850 seconds. In the case of using precopy live migration, there are Error responses approximately at 740 seconds.

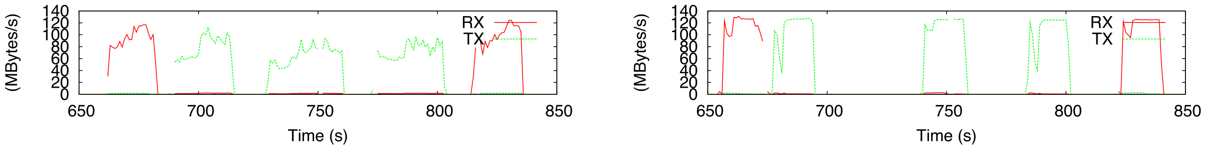

Fig. 17 The traffic of the migration network segment between 650 and 850 seconds; RX: migration traffic used for consolidation, TX: migration traffic used for distribution. The left graph shows the case of using precopy live migration and the right shows that of using postcopy.

results. The upper graphs show the overall view of the results during 900 seconds. The left column is the case of using precopy live migration, the center is that of using postcopy, and the right is the "no consolidation" case. At a glance, the overall results are approximately the same among these cases; the consolidation system successfully repacked VMs in response to load changes. However, as shown in the below graphs focusing between 650 and 850 seconds, the consolidation system using precopy migration was adversely affected due to the slowness of migration, in comparison with the cases of postcopy and no consolidation.

Approximately from 675 to 750 seconds, the benchmark client was stepping up the number of concurrent sessions. In response to this load increase, the consolidation system sent out 3 VMs to Dedicated Servers one by one. **Figure 17** shows the traffic of the migration network segment. Each spike corresponds to one live migration; the migrations of RX were performed for consolidation and those of TX were for distribution. In the precopy case, the consolidation system could not remove the overloading state, because each live migration took long time due to dirty page tracking and iterative memory copying. During this period, the Shared Server was continuously over-loaded, which resulted in the serious performance degradation. In the postcopy case, each live migration was completed in a shorter period of time, and the overloading of the Shared Server was quickly removed one by one. Through these experiments, we confirmed that postcopy live migration contributed to a higher level of performance assurance for web-hosting VMs in our dynamic consolidation system.

5. Discussion

5.1 The Drawback of Using Postcopy Live Migration

A disadvantage of using postcopy migration is that a consolidation system needs to be carefully designed to handle migration traffic. In our prototype system, the dedicated network segment for migration traffic is added to isolate it from other data transfer. Without this isolation, we experienced that bursty migration traffic prevented the system to control server nodes and sometimes suppressed workload traffic. This bursty migration traffic includes the on-demand page retrieval, which should be transferred as soon as possible to avoid performance degradation. One option is to use the dedicated network segment for migrations as shown in our prototype. However, this design incurs additional cost. Another option is to integrate an intelligent traffic control mechanism. In another project, we are working on reducing migration traffic by intelligently caching memory pages among server nodes.

In postcopy migration, an on-going live migration cannot be canceled. It is difficult to implement this feature because the execution host is already switched to a destination node. The consolidation system cannot abort an on-going repacking plan when the plan becomes out-of-date due to sudden load change. If the consolidation system uses precopy migration, it is possible to develop a packing algorithm that can revise an on-going repacking process.

If the network between source/destination nodes becomes down, for example, due to hardware failure, an on-going postcopy migration cannot be gracefully canceled. The system temporarily stops the migrating VM until its migration session is reestablished. Some IaaS service providers, presenting service availability agreement, may be concerned about this limitation. It is however possible to consider this kind of failure is very rare and can be recovered by a backup mechanism of networking. Through discussion with commercial IaaS providers, we have the first impressions that this limitation is acceptable in their datacenters.

In postcopy migration, the key to reduce performance degradation is to precache important memory pages. This mechanism, however, does not effectively work if the current workload reads or writes a wide range of memory pages in a random manner. In precopy migration, the completion time of migration greatly increases especially if the current workload intensively writes a wide range of memory pages. However, if the workload reads a wide range of memory pages and rarely writes memory pages, the completion time does not increase very much. In the latter case, we consider that the consolidation system should use precopy migration; although the drawback of precopy will not be in-

© 2012 Information Processing Society of Japan

tensified, that of postcopy will become serious. At this moment, our consolidation system cannot dynamically choose the type of live migrations. However, we are now trying to merge postcopy migration to the mainline of Qemu/KVM, which will allow consolidation systems to select precopy/postcopy migrations seamlessly. We hope further detail will be discussed with the merged implementation of postcopy migration.

5.2 The Threshold Parameters of the Packing Algorithm

This subsection adds more details regarding the threshold parameters of the packing algorithm. The reason behind these parameters is based on the lessons we learned through the development and evaluation of the prototype system.

5.2.1 The Threshold at 90% in the Overloading Detection

Although the CPU utilization of a server node is below 100%, some VMs on the server node may feel performance degradation due to scheduling latency; although the host operating system of the node gives all requested time slots to all the VMs on the node, some of the VMs may need to wait for a bit longer period until getting their time slots. This will adversely affect, for example, the response time of applications. If we choose a small value for this parameter, however, the system cannot put many idle VMs into Shared Servers.

The threshold at 90% includes 10% margin to alleviate the above problem.

5.2.2 The Threshold at 50% in the Idling Detection

If the latest average value of CPU utilization is below this threshold, the system tries to move back the VM to a Shared Server. This parameter configures how likely the system assigns a Dedicated Server to a VM. At this moment, we choose 50%; because there are not so many VMs in experiments, there is a large possibility that a Shared Server has CPU resource that can host mid-active (i.e., consuming around 30% CPU usage) VMs.

If we perform experiments with the large number of server nodes and VMs, we will decrease this value. Many idle VMs on a Shared Server consume substantially large CPU resource, and there is no room to move back mid-active VMs.

In addition, there is another reason for this. We observed that a live migration sometimes made the execution of the current workload on the migrating VM slow down. In some cases, after the migration was completed, the CPU utilization of the VM temporarily increased because the workload started to process pending jobs. This is more likely happen when the CPU utilization of the migrating VM is higher. The VM that just moved back to a Shared Server may pop out to a Dedicated Server again due to this temporal increase, and then the VM will move back to a Shared Server again. In the worst case, even though the workload does not change, this round trip continues repeatedly. Therefore, the threshold value should be small enough to avoid such problems.

5.2.3 The 5 Seconds Average for the Overloading/Idling Detection

We use the latest 5-seconds average value of CPU utilization, because we want to prevent the system to excessively respond a spiky increase/decrease of CPU utilization. If we choose a small value for this parameter, the system sometimes performs unnecessary migrations. Otherwise, the system sometimes does not perform necessary migrations. For now, we choose 5 seconds for this, considering trade-off between them.

5.2.4 The 20 Seconds Migration Margin in the Idling Detection

As written in Section 3.3.2, the system does not move the migrated VM for at least 20 seconds after the last migration ends. This parameter is intended to prevent the system from doing over-reaction.

Just after a live migration is completed, the behavior of the workload on the VM sometime becomes unstable. First, the workload may temporarily consume more CPU resource to process pending jobs. As explained above, this problem is alleviated by the threshold value in the idling detection.

Second, the workload may temporarily consume less CPU resource than expected. When a VM is popping out from a Shared Server, the overload of the Shared Server and the overhead of the migration make the VM temporarily slow down. Some kinds of applications, which change their behaviors in response to how ongoing jobs are processed, may temporarily run slowly by themselves just after the migration; after a couple of seconds, they start running fast as expected. Therefore, the migration margin in the idling detection is used to mitigate this problem. Considering the trade-off between the problem and the responsiveness of the system, we currently choose 20 seconds, which basically works fine through experiments.

5.3 ACPI S3 Integration

We are now implementing the suspend/resume feature to our consolidation system. Although a suspend server node is resumed approximately in 5 seconds, this transition time will give a negative impact for performance assurance; a live migration is postponed until a target server node is resumed. To alleviate this problem, we are considering the improved packing algorithm where a few of server nodes are resumed in advance. The consolidation system can invoke live migrations without resuming a target destination node. However, this strategy requires additional power consumption to keep some unused server nodes in the power-on state. Further discussion regarding energy efficiency will be given in our upcoming work.

6. Related Work

SnowFlock [11] provides a VM cloning system enabling developers to easily program distributed systems. A postcopy technique is used to rapidly copy the state of a master VM to worker VMs. Reference [5] developed a postcopy live migration mechanism for the paravirtualization mode of Xen, which exploited the swap-in/out code of the Linux kernel for on-demand memory transfer. As described in our previous work [6], we have developed a postcopy live migration mechanism for KVM. We will publish its source code under an open source license [1].

To the best of our knowledge, our study is the first work exploiting postcopy live migration for dynamic VM consolidation. The following studies regarding VM consolidation are based on precopy live migration.

Reference [16] showed that using workload-specific activity data, such as request arrival rates and response time, makes more optimized relocations possible; resource demand of VMs is estimated and predicted by queuing theory and autoregression analysis. However, this approach is not suitable for IaaS datacenters where various customers can run any type of workloads in their VMs. In this paper, we have tackled this issue by developing a reaction-based consolidation, in which postcopy live migration greatly contributes to performance assurance.

In Ref. [9], a consolidation system uses a threshold value of resource usage to trigger VM repacking; if the CPU usage of a host exceeds this value, the system re-optimizes VM locations, so that mitigates the risk that application response times (e.g., service level agreement in this study) are adversely affected. Reference [2] exploits an anomaly detection technique based a stochastic model, which determines the VMs and hosts subject to significant state changes. This study argues that a threshold-based algorithm incorrectly detects overloading and mistakenly determines a reconfiguration plan. Reference [14] discusses the way of finding turning points of resource demands, where reconfiguration of VM locations is advisable. This technique aims to determine whether repacking is required or not with small calculation cost. Entropy [4] is a VM packing management system exploiting constraint programming techniques. It first determines the minimum number of nodes that are necessary to host all VMs, and then computes an optimal order of migrations to minimizing the overall reconfiguration time. Reference [15] presents a network-aware migration scheduling algorithm, which tries to minimize the bandwidth usage while holding migration deadlines.

We consider that the basic ideas in these techniques are basically orthogonal to our study. It is possible to extend these techniques to cover postcopy migrations. We believe that researchers will make further studies by using our publicly-available code of postcopy live migration.

7. Conclusions

In this paper, we have proposed a reactive VM consolidation system exploiting postcopy live migration. Postcopy live migration makes a higher level of performance assurance possible for dynamic VM consolidation than does precopy live migration. Sudden overloadings of server nodes are quickly resolved by switching the execution hosts of VMs within one second. We developed a prototype of the proposed consolidation system and conducted experiments to verify its feasibility. Our micro benchmark program, designed for the metric of performance assurance, showed the proposed system greatly alleviated performance degradation; the percentage of failed operations averaged under 12% or less, even for memory intensive workloads. This is less than half the level when using precopy live migration. The SPECweb benchmark program showed that our consolidation system with postcopy live migration achieved only 10% performance degradation from an ideal case, which was greatly alleviated from the case of using precopy live migration (21%).

Acknowledgments This work was partially supported by KAKENHI (20700038 and 23700048) and JST/CREST ULP.

Reference

[1] AIST Cloud Computing Research, available from ⟨http://grivon.apgrid.org/⟩.

[2] Andreolini, M., Casolari, S., Colajanni, M. and Messori, M.: Dynamic load management of virtual machines in a cloud architectures, *Proc. IEEE Conference on Cloud Computing*, pp.201–214 (2009).

[3] Clark, C., Fraser, K., Hand, S., Hansen, J.G., Jul, E., Limpach, C., Pratt, I. and Warfield, A.: Live migration of virtual machines, *Proc. 2nd Symposium on Networked Systems Design and Implementation*, pp.273–286, USENIX Association (2005).

[4] Hermenier, F., Lorca, X., Menaud, J.-M., Muller, G. and Lawall, J.L.: Entropy: A consolidation manager for clusters, *Proc. 5th International Conference on Virtual Execution Environments*, pp.41–50, ACM Press (2009).

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

[6] Hirofuchi, T., Nakada, H., Itoh, S. and Sekiguchi, S.: Enabling Instantaneous Relocation of Virtual Machines with a Lightweight VMM Extension, *Proc. 10th IEEE/ACM International Conference on Cluster, Cloud and Grid Computing*, pp.73–83, IEEE Computer Society (2010).

[7] Hirofuchi, T., Nakada, H., Itoh, S. and Sekiguchi, S.: A Guest-transparent, Post-copy-based VM Migration Mechanism with a Lightweight Extension to KVM, *IPSJ Transactions on Advanced Computing Systems*, Vol.ACS31, pp.248–262 (2010).

[8] Hirofuchi, T., Nakada, H., Ogawa, H., Itoh, S. and Sekiguchi, S.: Eliminating Datacenter Idle Power with Dynamic and Intelligent VM Relocation, *Distributed Computing and Artificial Intelligence* (*7th International Symposium*), Advances in Intelligent and Soft Computing, Vol.79, pp.645–648, Springer (2010).

[9] Khanna, G., Beaty, K., Kar, G. and Kochut, A.: Application Performance Management in Virtualized Server Environments, *Proc. IEEE/IFIP Network Operations and Management Symposium*, pp.373–381 (2006).

[10] Kivity, A., Kamay, Y., Laor, D. and Liguori, A.: kvm: The Linux Virtual Machine Monitor, *Proceedings of the Linux Symposium*, The Linux Symposium, pp.225–230 (2007).

[11] Lagar-Cavilla, H.A., Whitney, J.A., Scannell, A., Patchin, P., Rumble, S.M., de Lara, E., Brudno, M. and Satyanarayanan, M.: SnowFlock: Rapid Virtual Machine Cloning for Cloud Computing, *Proc. 4th ACM European Conference on Computer Systems*, pp.1–12, ACM Press (2009).

[12] Mirkin, A., Kuznetsov, A. and Kolyshkin, K.: Containers checkpointing and live migration, *Proc. Linux Symposium*, The Linux Symposium, pp.85–92 (2008).

[13] Nelson, M., Lim, B.-H. and Hutchins, G.: Fast transparent migration for virtual machines, *Proc. USENIX Annual Technical Conference*, pp.391–394, USENIX Association (2005).

[14] Setzer, T. and Stage, A.: Decision Support for Virtual Machine Reassignments in Enterprise Data Centers, *Proc. 5th IEEE/IFIP International Workshop on Business-driven IT Management*, pp.88–94 (2010).

[15] Stage, A. and Setzer, T.: Network-aware migration control and scheduling of differentiated virtual machine workloads, *Proc. 2009 ICSE Workshop on Software Engineering Challenges of Cloud Computing*, pp.9–14, IEEE Computer Society (2009).

[16] Wood, T., Shenoy, P.J., Venkataramani, A. and Yousif, M.S.: Black-box and Gray-box Strategies for Virtual Machine Migration, *Proc. 4th Symposium on Networked Systems Design and Implementation*, pp.229–242, USENIX Association (2007).

Takahiro Hirofuchi is a researcher of National Institute of Advanced Industrial Science and Technology (AIST) in Japan. He is working on virtualization technologies for advanced cloud computing and Green IT. He obtained a Ph.D. of Engineering in March 2007 at the Graduate School of Information Science of Nara Institute of Science and Technology (NAIST). He obtained a B.S. of Geophysics at Faculty of Science in Kyoto University in March 2002. He is an expert of operating system, virtual machine, and network technologies.

Hidemoto Nakada graduated the University of Tokyo, in 1995. He earned a Ph.D. in Computer Engineering from the University of Tokyo and joined Electrotechnical Laboratory (ETL). Now he is working for National Institute of Advanced Industrial Science and Technology (AIST) as a Senior Research Scientist. His research interests include distributed and parallel computing, grid and cloud computing. Member of IPSJ and ACM.

Satoshi Itoh obtained a Ph.D. in physics from University of Tsukuba, Japan, in 1987. From 1987 to 2002 he worked for high-performance and parallel computing in the both area of material science and business application at Central Research Laboratory, Hitachi, Ltd. In 2002, he moved to National Institute of Advanced Industrial Science and Technology (AIST), Japan and has researched on grid computing, cloud computing, and green IT. He is currently the Deputy Director of Information Technology Research Institute, AIST.

Satoshi Sekiguchi received a B.S. from the University of Tokyo, a M.E. from University of Tsukuba, and a Ph.D. in Information Science and Technology from the University of Tokyo, respectively. He joined Electrotechnical Laboratory (ETL), Japan in 1984 to engage research in high-performance computing widely from its system architecture to applications. His expertise also includes applying IT-based solutions to many of society's problems related to global climate change, environmental management and resource efficiency. He served as the director of Grid Technology Research Center, National Institute of Advanced Industrial Science and Technology (AIST) in 2002–2008, and is currently the Director of Information Technology Research Institute, AIST. He has been contributing to the Open Grid Forum as a member of board of directors, is a member of IEEE, SIAM, and IPSJ.

© 2012 Information Processing Society of Japan