Hindawi Publishing Corporation
Journal of Applied Mathematics
Volume 2014, Article ID 297127, 7 pages
http://dx.doi.org/10.1155/2014/297127

Research Article

TSMC: A Novel Approach for Live Virtual Machine Migration

Jiaxing Song, Weidong Liu, Feiran Yin, and Chao Gao

Department of Computer Science and Technology, Tsinghua University, Beijing 100084, China

Correspondence should be addressed to Jiaxing Song; jxsong@tsinghua.edu.cn

Received 23 January 2014; Accepted 6 May 2014; Published 20 May 2014

Academic Editor: Young-Sik Jeong

Cloud computing attracted more and more attention in recent years, and virtualization technology is the key point for deploying infrastructure services in cloud environment. It allows application isolation and facilitates server consolidation, load balancing, fault management, and power saving. Live virtual machine migration can effectively relocate virtual resources and it has become an important management method in clusters and data centers. Existing precopy live migration approach has to iteratively copy redundant memory pages; another postcopy live migration approach would lead to a lot of page faults and application degradation. In this paper, we present a novel approach called TSMC (three-stage memory copy) for live virtual machine migration. In TSMC, memory pages only need to be transmitted twice at most and page fault just occurred in small part of dirty pages. We implement it in Xen and compare it with Xen's original precopy approach. The experimental results under various memory workloads show that TSMC approach can significantly reduce the cumulative migration time and total pages transferred and achieve better network IO performance in the same time.

1. Introduction

After the wave of pervasive computing and grid computing [1–3], the conception of cloud computing was officially proposed by Google. Since it appeared, cloud computing has a huge impact on the entire IT industry. There are many hot research areas in cloud computing. For example, resource management [4, 5] becomes more important in cloud computing. A lot of research works have been done worldwide [6–9].

Virtualization technology has played a very vital role in resource management of cloud computing and it develops rapidly in recent years. The resources of a single physical machine are divided into multiple isolated virtual resources by using some virtualization softwares [10]. The isolated virtual environment is called virtual machine (VM) [11]. It can provide application isolation, server consolidation, better multiplexing of data center resources, the ability to flexibly remap physical resources, and so on [12].

Live migration is the key point of virtualization technologies. It allows VMs fast relocation in data center and nonawareness of downtime. Lots of live migration techniques have been brought up these years [13, 14]. Most of them use precopy approach. It first transfers all memory pages to the target VM and then copies pages which are dirtied iteratively. However, great service degradation would happen in precopy phase because migration daemon continually consumes network bandwidth to transfer dirty pages in each round. Another approach called postcopy is also introduced into live migration of VMs. In this approach, all memory pages are transferred only once during the whole migration process and the baseline total migration time is achieved. But the downtime is much higher than that of precopy due to the latency of fetching pages from the source node before VM can be resumed on the target.

In this paper, we present an optimized memory copy approach for live virtual machine migration. We combine the advantages of active pushing and on-demand copy; first copy all memory pages to target and record dirty bitmap in this phase (full memory copy stage), then suspend the VM, transmit CPU state and dirty bitmap (dirty bitmap copy stage), and finally resume the new VM and copy dirty pages from source to target (dirty page copy stage). We call it TSMC (three-stage memory copy). The main goal of TSMC is to minimize total migration time and reduce network traffic. Most of the memory pages need to be copied once in full memory copy stage; only dirtied pages need to be copied twice. Many approaches have been proposed to

evaluate the performance of virtualization [15]. We chose to implement this TSMC approach on Xen [16] and compared it with original precopy method in Xen. The experiment results under various memory workloads show that our approach can significantly reduce the cumulative migration time and total pages transferred.

This paper is organized as follows. In Section 2, we describe related work. Then, in Section 3, we describe the design and implementation of TSMC and we present the experimental results in Section 4. Finally, we make a conclusion in Section 5.

2. Related Work

Precopy [17] live virtual machine migration approach was firstly proposed. In precopy approach, it first transfers all memory pages and then copies pages just modified during the last round iteratively, until writable working set (WWS) becomes small or the preset number of iterations is reached. Eventually, it suspends VM in source node and sends CPU state and the remaining dirty pages in the last round to the target, where the VM is restarted. There are many virtualization platforms using this approach, such as Xen [16], KVM [18], and VMware [19]. Precopy is the prevailing live migration technique to perform live migration of VMs, but in write-intensive workloads, memory pages will be repeatedly dirtied and may have to be transmitted multiple times.

Postcopy [20] instead of precopy was proposed to solve this problem and reduce total migration time. Postcopy migration defers the memory transfer phase until after the VM's CPU state has already been transferred to the target and resumed there. Postcopy ensures that each memory page is transferred at most once, thus avoiding the duplicate transmission overhead of precopy. But the downtime is much higher than that of the precopy due to the latency of fetching pages from the source node before VM can be resumed on the target.

Jin et al. [21] proposed a new mechanism using adaptive compression of migrated data; different compression algorithms are chosen depending on characteristics of memory pages. They first used memory compression to provide fast VM migration and they also designed a zero-aware characteristics-based compression (CBC) algorithm for live migration. In the source node, data being transferred in each round are first compressed by their algorithm. When arriving on the target, compressed data are then decompressed. However, memory compression increases the system overhead.

To overcome the shortcomings of precopy and postcopy approaches, many other live migration methods [22, 23] are proposed, but almost all of them have their own limitations.

3. Design and Implementation

In this section, we introduce the phase of live migration and describe the design of TSMC approach and its implementation on Xen. The performance of any live virtual machine migration strategy could be gauged by the following metrics.

Downtime. The time during which the migrating VMs are not executed.

Readiness Time. The time between the start of migration and the start of downtime.

Recover Time. The time between resuming the VMs execution at the target and the end of migration.

Total Migration Time. The total time of all migration times from start to finish.

Pages Transferred. The total amount of memory pages transferred, including duplicates, across all periods.

3.1. Memory Migration Phases. Efficient synchronization of the memory state is the key issue of live virtual machine migration. Memory transfer can be achieved by following three phases [17].

Push. The source VM continues running while certain pages are pushed across the network to the new destination. To ensure consistency, pages modified during this process must be resent.

Stop-and-Copy. The source VM is stopped, pages are copied across to the destination VM, and then the new VM is started.

Pull. The new VM is executed and, if it accesses a page that has not yet been copied, this page is faulted in "pulled" across the network from the source VM.

Figure 1(a) shows precopy approach; it combines push copying and stop-and-copy. Another approach called postcopy in Figure 1(b) uses stop-and-copy and pull copying.

3.2. Design of TSMC. To solve the weaknesses of existing live migration methods, we propose a new approach called three-stage memory copy (TSMC) which combines three phases of memory transfer. The entire memory synchronization is divided into three stages. Figure 1(c) is the three-stage copy timeline.

Full Memory Copy. Copy all memory pages from source VM to destination when the source VM continues running and record pages modified during this process.

Dirty Bitmap Copy. Suspend source VM, copy recorded dirty bitmap to target node, and mark corresponding pages as dirty in destination VM.

Dirty Pages Copy. Resume new VM and then active push or copy dirty pages on demand from source VM to destination.

Compared with precopy, three-stage copy avoids iterative copy of dirty pages; most of the memory pages are just copied once and only dirtied pages in full memory copy stage need to be copied twice. It significantly reduces pages transferred, thus reducing the usage of network bandwidth. Meanwhile, only dirty bitmap and CPU state need to be transferred in suspend phase; downtime of VM is also shortened. Although

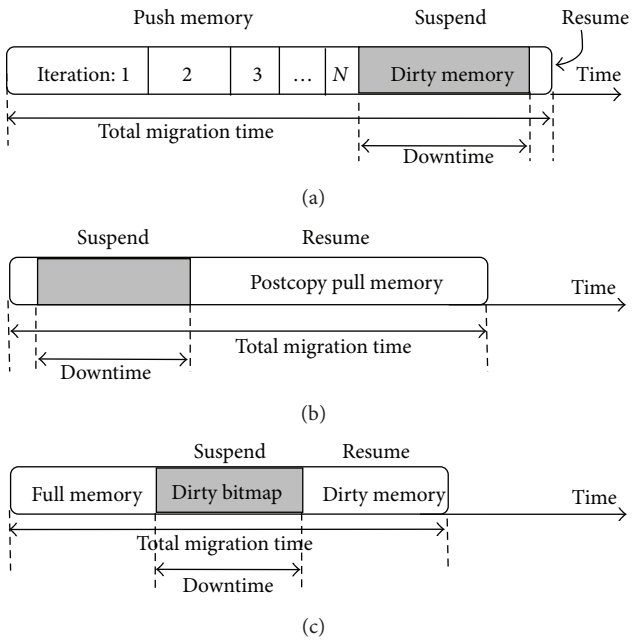

Figure 1: Timeline for live migration approach.

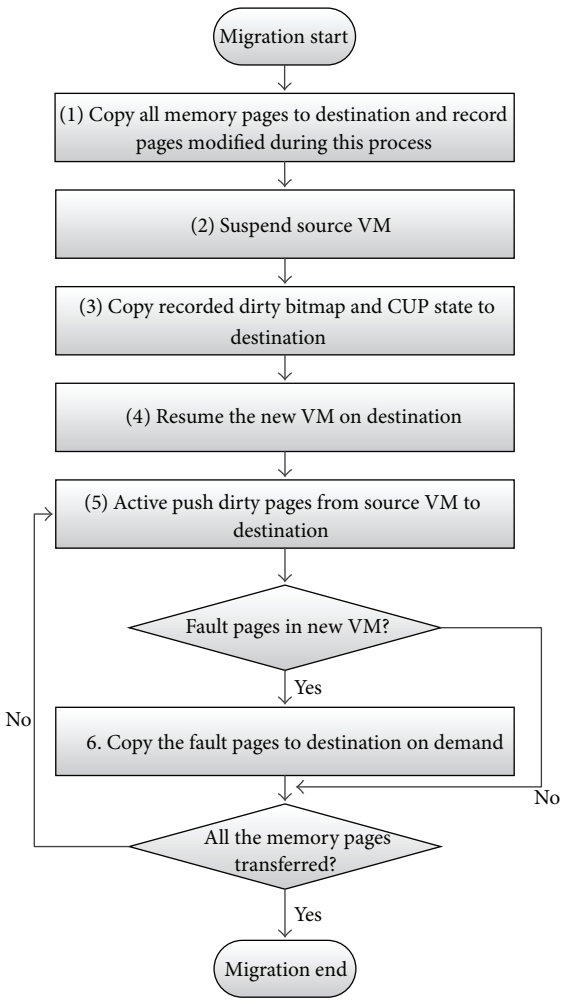

Figure 2: Procedure of TSMC.

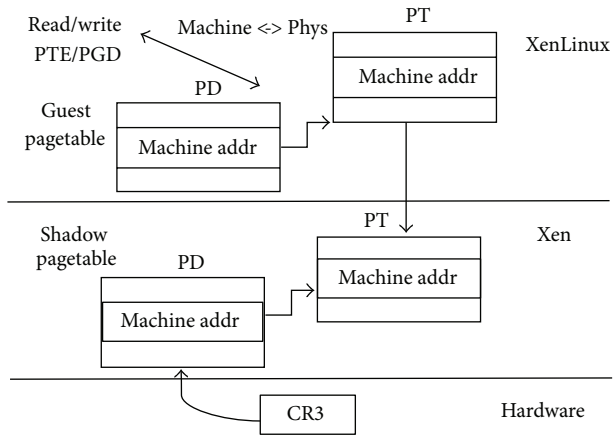

Figure 3: Shadow page table.

it would be interrupted in dirty pages copy stage because of page fault, but relative to full memory copy after resuming new VM in postcopy approach, three-stage copy just transfers dirtied pages after resuming, which significantly reduces the page fault rate and avoids obvious application degradation; also, it shortens the duration of the migration.

There are two methods used for transferring dirty page: on-demand copy and active push. Once the VM is resumed on the target, page faults would happen when memory access dirtied page; it can be serviced by requesting the referenced page over the network from the source node. However, page faults in new VM are unpredictable; on-demand copy would lead to longer resume time, so we combine it with active push whose source host periodically pushes dirty pages to the target in a preset time interval.

The procedure of TSMC is shown in Figure 2. In full memory copy phase, the update of memory pages should be recorded to dirty pages bitmap. Otherwise, memory changes of applications during the process cannot be updated to the new VM. Operating system (OS) on VM maintains the mapping page table from VM's linear addresses to OS's physical addresses, while VM monitor maintains translation page table from VM's physical addresses to physical host's physical addresses. VM monitor cannot monitor the memory changes of VM directly due to the transparency demand. So we utilize translation page table in virtualization tools to monitor the update of pages. In dirty pages copy phase, the page faults also need to be captured by VM monitor, which also can be achieved in the same way.

On-demand copy is the easiest and the slowest way. When the VM on destination resumes, the page faults will be transferred to source VM via the network and request the corresponding memory pages. Although on-demand copy copies dirty pages only once, it lengthens recovery time and degrades software performance. So it is unacceptable to transfer memory pages using on-demand copy alone.

Active push can reduce recovery time efficiently. It also reduces the long-time occupation of source VM's resources. After new VM resumes, active push pushes dirty pages from source VM to destination in a preset interval. It avoids some page faults on new VM. When page faults occur, we request

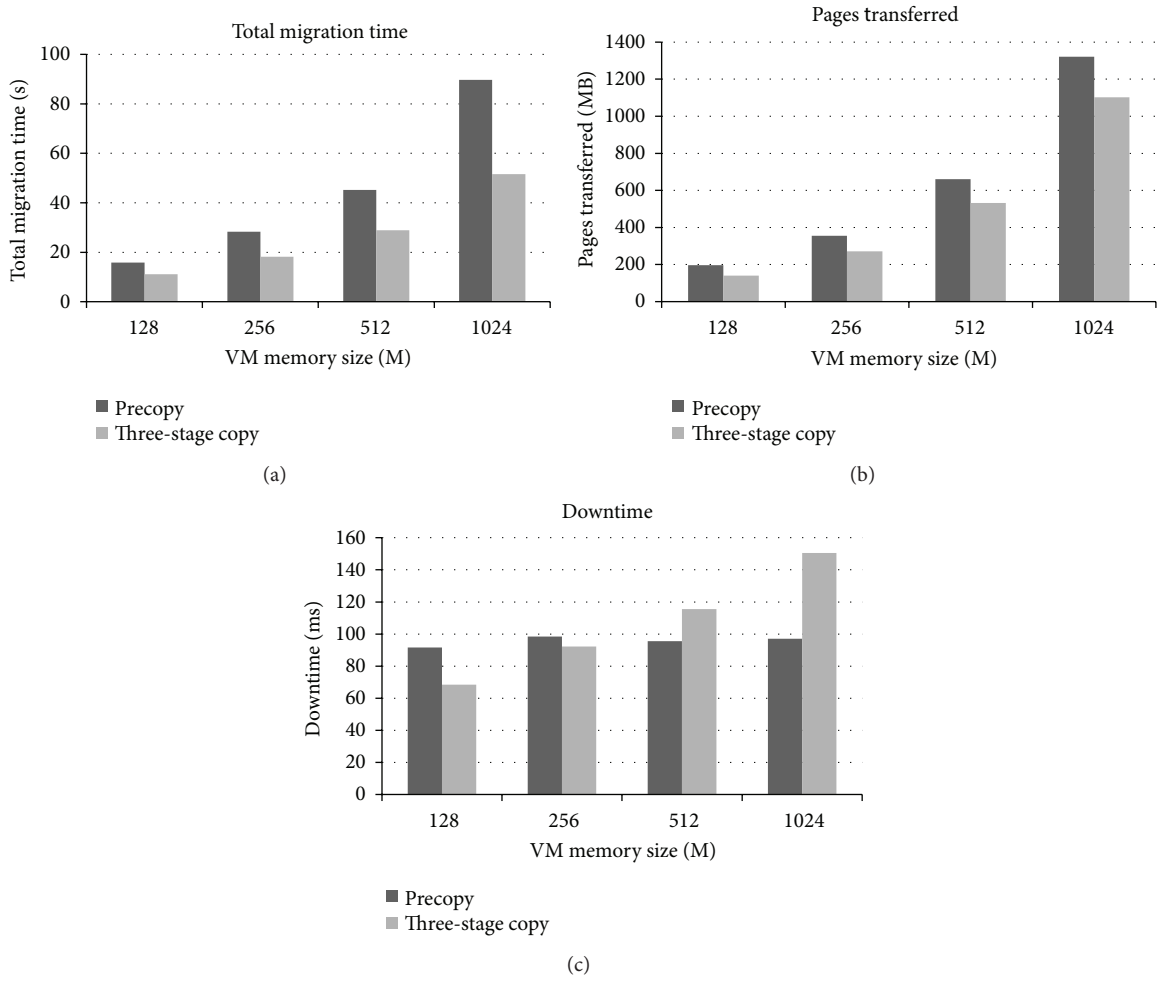

Figure 4: Comparison of total migration time, pages transferred, and downtime.

pages from source VM in the way of on-demand copy. The performance will be improved greatly by combining on-demand copy and active push.

Prepull was first brought up to predict the recent working set of softwares and it was based on software's running history. In three-state copy, prepull is used to predict page faults in new VM. When page faults occur, then the pages around the missing page will probably be accessed, which leads to another page fault. Prepull increases the memory transfer window. When requesting pages from source VM, it transfers the pages around along with the request pages. In this way, less page faults will occur in the future.

3.3. Implementation on Xen. We implemented TSMC on Xen 4.1.4. The point of our approach is to capture and recode dirty pages. Shadow page tables are used by Xen's hypervisor to keep track of the memory state of guest OS; it can be used to capture dirty pages. Figure 3 shows the process of shadow page table. Shadow page tables are a set of read-only page tables for each VM maintained by the hypervisor that maps the VM's memory pages to the physical frames. Actually, it is equivalent to a backup of the original page tables; any updates in guest OS's page table will notify Xen's hypervisor by Hypercall.

Because all page tables in guest OS are mapped to read-only shadow page tables, any updates in page tables trigger page faults which would be captured by Xen's hypervisor. Xen checks the PTE access right of the guest OS and sets PTE in shadow page tables to writable if the guest OS is writable to the PTE. Then we can record the updates in shadow page tables into a dirty bitmap.

By this way, we will be able to capture the occurrence of dirty pages and obtain a dirty page bitmap. Xen provides an API function xc_shadow_control() to handle shadow page tables. This feature can be turned on by calling xc_shadow_control() and setting flag as XEN_DOMCTL_SHADOW_OP_ENABLE_LOGDIRTY before live migration and turned off by setting XEN_DOMCTL_SHADOW_OP_OFF flag after migration finished.

4. Experiment Results

In this section, we present an evaluation of three-stage copy on Xen 4.1.4 and compare it with Xen's original precopy approach.

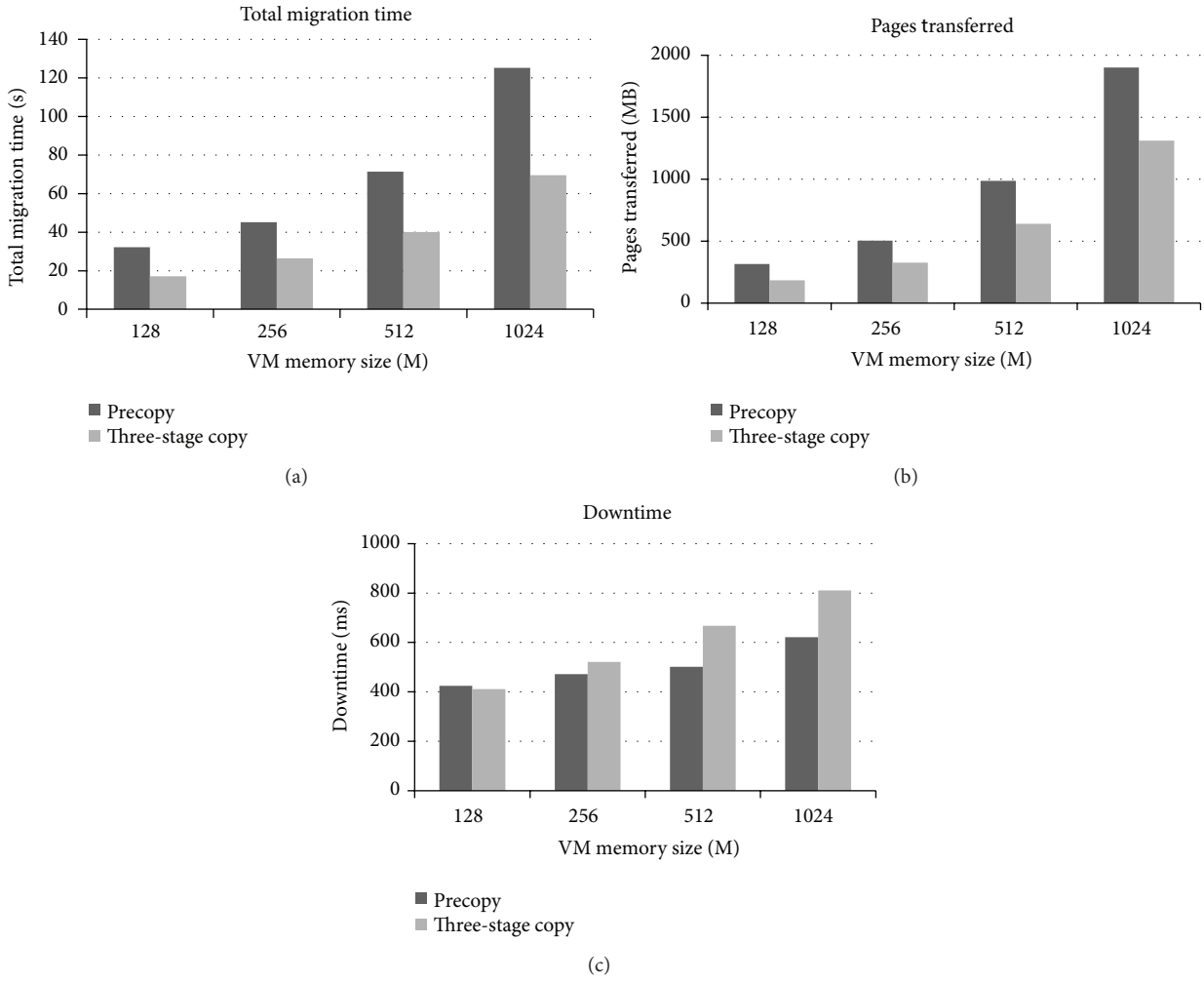

Figure 5: Comparison of total migration time, pages transferred, and downtime.

We conduct our experiments on two identical server-class machines, each with 2-way quad-core Xeon E5506 2.13 GHz CPUs and 32 GB DDR RAM, connected via a Gigabit Ethernet switch. All VM images are stored in a NFS server. We use Ubuntu 12.04 (Linux version 3.5.0-23) as guest OS and the privileged domain OS (domain 0). The host kernel is the modified version of Xen 4.1.4. Both the VM in each experiment and the domain 0 are configured to use two VCPUs. Guest VM sizes range from 128 MB to 1024 MB. And we use memtester [24] in virtual machine to generate high memory usage.

Each experiment is repeated five times and every test result comes from the arithmetic average of five values. In migration process, we evaluate three primary metrics discussed in Section 3: downtime, total migration time, and page transferred.

4.1. Low Dirty Pages Rate. Figure 4(a) shows that three-stage copy significantly reduces the total migration time for diverse VM memory size compared with precopy. With memory size increasing, the total migration time is reduced more. It reduces total migration time by average of 36.2%. In clusters or data centers, less total migration time of VMs would get higher flexibility.

Figure 4(b) shows that three-stage copy approach also has the advantage in pages transferred; this should be attributed to less data transferred and lower network bandwidth needed. Experimental results show that the three-stage copy reduces pages transferred by average of 22%.

Evaluation in downtime Figure 4(c) shows that precopy could get stable downtime and three-stage copy's downtime would increase along with the increase of memory size. At low memory environment, three-stage copy needs less downtime than precopy, but it needs more downtime in large memory environment. It is due to the three-stage copy need to transfer dirty bitmap in suspend phase that large memory size would have more dirty pages. Nevertheless, the tradeoff between total migration time and downtime may be acceptable.

4.2. High Dirty Pages Rate. Figure 5(a) shows that in the case of high dirty pages rate the total migration time increases obviously in precopy while three-stage copy still maintains a shorter migration time. The three-stage copy reduces total migration time by average of 44.1% compared with precopy.

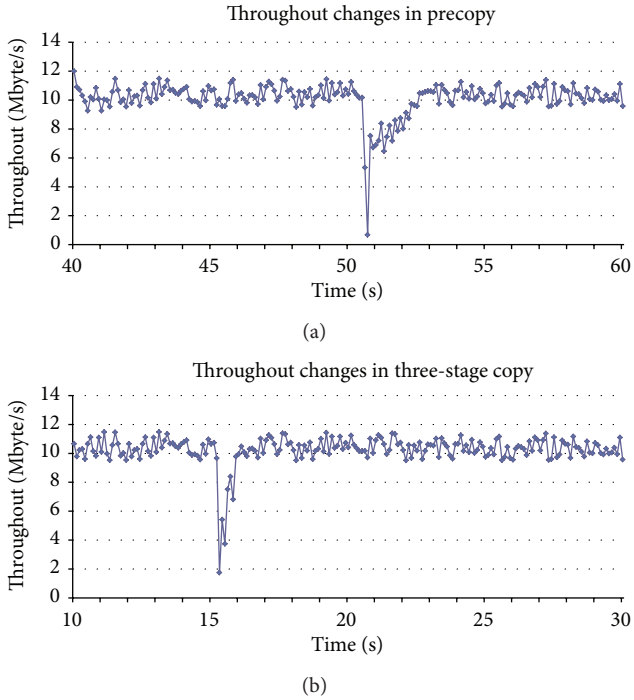

Figure 6: Comparison of total migration time, pages transferred, and downtime.

Figure 5(b) shows that three-stage copy approach still has the advantage in pages transferred over the precopy in the case of high dirty pages rate. As shown in the figure, it reduces pages transferred by average of 35.7%, which has a better optimization effect than in the case of low dirty pages rate.

Evaluation in Figure 5(c) shows that both precopy and three-stage copy have a large increase in downtime. This is because more dirty pages or dirty bitmap need to be transferred in downtime. In this test, longer downtime is needed in three-stage copy, but it still remains below 1 second in the worst case.

4.3. Network IO. We focus on the network throughput in the network IO test. As shown in Figure 5, the throughputs of both precopy and three-stage copy stay stable over a period of time and then suddenly drop and recover soon. The time when throughputs suddenly drop is the stage when the VMs down and copy pages. We can see that the duration of throughputs volatility is shorter in the three-stage copy. It means that throughput recovers faster from the bottom. It is because three-stage copy speeds up the transmission by the means of active push and prefetch pages after the VMs resume. As shown in Figure 6, it costs about 3 seconds to recover to normal throughput in precopy while only less than 1 second in three-stage copy.

5. Conclusions

This paper presents a three-stage memory copy (TSMC) approach for live virtual machine migration. In TSMC approach, the entire memory copy is divided into three stages: full memory copy, dirty bitmap copy, and dirty pages copy. Most of the memory pages are just copied once; only dirtied pages need to be copied twice. It can significantly reduce the total pages transferred and cumulative migration time. Furthermore, because the TSMC approach just transfers dirty bitmap in stop phase of virtual machine, downtime is also shortened. Experimental results show that the TSMC approach could get better performance than Xen's precopy.

Conflict of Interests

The authors declare that there is no conflict of interests regarding the publication of this paper.

References

[1] T. Ohkawara, A. Aikebaier, T. Enokido, and M. Takizawa, "Quorums-based replication of multimedia objects in distributed systems," *Human-Centric Computing and Information Sciences*, vol. 2, article 11, 2012.

[2] S. Silas, K. Ezra, and E. B. Rajsingh, "A novel fault tolerant service selection framework for pervasive computting," *Human-Centric Computing and Information Sciences*, vol. 2, article 5, 2012.

[3] B. Meroufel and G. Belalem, "Dynamic replication based on availability and popularity in the presence of failures," *The Journal of Information Processing Systems*, vol. 8, no. 2, pp. 263–278, 2012.

[4] N. Y. Yen and S. Y. F. Kuo, "An intergrated approach for internet resources mining and searching," *The KITCS/FTRA Journal of Convergence*, vol. 3, no. 3, pp. 37–44, 2012.

[5] F. Xhafa, "Processing and analysing large log data files of a virtual campus," *The KITCS/FTRA Journal of Convergence*, vol. 3, no. 2, pp. 1–8, 2012.

[6] Y. Pan and J. Zhang, "Parallel programming on cloud computing platforms—challenges and solutions," *The KITCS/FTRA Journal of Convergence*, vol. 3, no. 4, pp. 23–28, 2012.

[7] E.-H. Song, H.-W. Kim, and Y.-S. Jeong, "Visual monitoring system of multi-hosts behavior for trustworthiness with mobile cloud," *The Journal of Information Processing Systems*, vol. 8, no. 2, pp. 347–358, 2012.

[8] B. J. Oommen, A. Yazidi, and O.-C. Granmo, "An adaptive workflow scheduling scheme based on an estimated data processing rate for next generation sequencing in cloud computing," *The Journal of Information Processing Systems*, vol. 8, no. 4, pp. 191–212, 2012.

[9] C. Waldspurger, "Memory resource management in VMware ESX server," in *ACM Operating Systems Design and Implementation*, pp. 181–194, VMware, 2002.

[10] R. P. Goldberg, "Survey of virtual machine research," *IEEE Computer*, pp. 34–45, 1974.

[11] G. H. S. Carvalho, I. Woungang, A. Anpalagan, and S. K. Dhurandher, "Virtual machine history model framework for a data cloud digital investigation," *The KITCS/FTRA Journal of Convergence*, vol. 3, no. 4, pp. 15–22, 2012.

[12] T. Wood, P. Shenoy, A. Venkataramani, and M. Yousif, "Black-box and gray-box strategies for virtual machine migration," in *Proceedings of the 4th USENIX Symposium on Networked Systems Design and Implementation*, pp. 229–242, 2007.

[13] D. Kapil, E. S. Pilli, and R. C. Joshi, "Live virtual machine migration techniques: survey and research challenges," in *Proceedings*

of the Advance Computing Conference (IACC '13), pp. 963–969, 2013.

[14] P. G. J. Leelipushpam and J. Sharmila, "Live VM migration techniques in cloud environment—a survey," in *Proceedings of the Information & Communication Technologies (ICT '13)*, pp. 408–413, 2013.

[15] X. Xie, H. Jiang, H. Jin, W. Cao, P. Yuan, and L. Yang, "Metis: a profiling toolkit based on the virtualization of hardware performance counters," *Human-Centric Computing and Information Sciences*, vol. 2, article 8, 2012.

[16] P. Barham, B. Dragovic, K. Fraser et al., "Xen and the art of virtualization," in *Proceedings of the 19th ACM Symposium on Operating Systems Principles (SOSP '03)*, pp. 164–177, October 2003.

[17] C. Clark, K. Fraser, S. Hand et al., "Live migration of virtual machines," in *Proceedings of the 2nd conference on Symposium on Networked Systems Design & Implementation (NSDI '05)*, pp. 273–286, 2005.

[18] A. Kivity, Y. Kamay, and D. Laor, "KVM: the linux virtual machine monitor," in *Proceedings of the Ottawa Linux Symposium*, pp. 225–230, 2007.

[19] M. Nelson, B. Lim, and G. Hutchines, "Fast transparent migration for virtual machines," in *Proceedings of the USENIX Annual Technical Conference*, pp. 391–394, 2005.

[20] M. R. Hines and K. Gopalan, "Post-copy based live virtual machine migration using pre-paging and dynamic self-ballooning," in *Proceedings of the ACM SIGPLAN/SIGOPS International Conference on Virtual Execution Environments (VEE '09)*, pp. 51–60, March 2009.

[21] H. Jin, D. Li, S. Wu, X. Shi, and X. Pan, "Live virtual machine migration with adaptive memory compression," in *Proceedings of the IEEE International Conference on Cluster Computing and Workshops (CLUSTER '09)*, September 2009.

[22] L. Haikun, J. Hai, L. Xiaofei, H. Liting, and Y. Chen, "Live migration of virtual machine based on full system trace and replay," in *Proceedings of the 18th ACM International Symposium on High Performance Distributed Computing (HPDC '09)*, pp. 101–110, June 2009.

[23] H. A. Lagar-Cavilla, J. A. Whitney, R. Bryant et al., "SnowFlock: Virtual machine cloning as a first-class cloud primitive," *ACM Transactions on Computer Systems*, vol. 29, no. 1, article 2, 2011.

[24] A utility for testing memory, http://pyropus.ca/software/memtester/.