Applied Mathematics & Information Sciences
An International Journal

http://dx.doi.org/10.12785/amis/072L38

HMDC: Live Virtual Machine Migration Based on Hybrid Memory Copy and Delta Compression

Liang Hu[1], *Jia Zhao*[1], *Gaochao Xu*[1], *Yan Ding*[1] *and Jianfeng Chu*[1]

[1]College of Computer Science and Technology, Jilin University, Changchun, China

Received: 7 Jun. 2012, Revised: 21 Sep. 2012, Accepted: 23 Sep. 2012
Published online: 1 Jan. 2013

Abstract: Live VM (virtual machine) migration has become a research hotspot of virtualized cloud computing architecture. We present a novel migration algorithm which is called HMDC. Its main idea includes two parts. One is that it combines memory pulling copy with memory pushing copy to achieve hybrid memory copy. The other one is that it uses a delta compression mechanism during dirty pages copy, in which source host makes use of memory cache to get XOR delta pages, and then compresses delta pages which are easy to compress by the XOR binary RLE (run-length encoding) algorithm. Source host transmits delta compression pages instead of dirty pages to target host. HMDC increases throughput and decreases total migration data by using delta compression and thus to achieve dirty pages copy quickly. The experimental results demonstrate that HMDC evidently reduces total migration time, VM downtime and total migration data compared with Pre-copy and XBRLE algorithm. It makes the process of memory migration more high-effective and transparent.

Keywords: Virtual Machine, Live Migration, Hybrid Memory Copy, Delta Compression, Pre-paging

1 Introduction

Cloud computing has become the most promising research direction in the field of distributed systems. Iaas (Infrastructure as a Service) [1] is considered to be a pioneer in cloud computing. The VM technology [2,3] is the most crucial to Iaas for achieving high flexibility and scalability. Live VM migration technology is the significant application and embody of the flexibility and availability of VM technology. Currently, live migration is widely used for the maintenance management in virtualized cloud computing data centers. When the workload of some host is too heavy, we can move its VMs which are providing services to an idle host to achieve load balance through live migration. When some host needs shutdown for maintenance, we can move its VMs to other hosts transparently through live migration. We can move the VMs providing services to clients through live migration in order to achieve high-efficient local services.

It is the key for live VM migration technology to achieve a high-efficient replication of memory status between source and target host. Therefore, a good memory migration algorithm for live VM migration is extremely important. It has abilities in directly determining whether live VM migration succeeds or not. The current most virtualization platforms all provide the live migration function. However, the algorithms are restrained by their own characteristics, which cant get the ideal migration efficiency. In addition, virtualized cloud computing systems need to achieve resources sharing, so in general it isnt allowed to exclusively use physical resources. The existing memory migration mechanisms in a resource-constrained environment often show either the inefficient performance or a failure sometimes. In addition, some migration operation may damage the demands of other entities for physical resources and itself be limited by the physical resources occupied by other entities in the environment with shareable physical resources.

Aiming at the lack of the performance of existing memory migration mechanisms and the characteristics of cloud computing environment that physical resources are shared and limited, this paper proposes a novel memory migration algorithm based on hybrid memory copy and delta compression to address some traditional issues and challenges of memory migration.

The rest of this paper is organized as follows. In Section 2, we present the related work of memory

* Corresponding author e-mail: zhaiyj049@sina.com

migration briefly and the prerequisites that should be satisfied are shown clearly. In Section 3, we introduce the main idea and implementation of HMDC in detail. In Section 4, the experimental results and analysis on Xen platform are given. Finally, in Section 5, we summarize the full paper and future work is put forward.

2 Related work

At present, there are mainly two kinds of popular memory migration algorithms based on memory-to-memory: Pre-copy and Post-copy.

Pre-copy [4] is proposed and implemented by Clark et al. Recently, the most popular virtualization platforms such as Xen, KVM, VMware and so on all implement the algorithm, thereby Pre-copy widely is used for live VM migration in Lan. In Pre-copy algorithm, memory migration is divided into three stages: total memory copy, iterative copy and stop-and-copy. Pre-copy algorithm not only shortens VM downtime but also avoids the unpredictable overhead and errors caused by too long VM downtime. However, Pre-copy has a convergence problem since it needs many rounds of iterative copy to achieve memory migration but Pre-copy cant predict or control memory update rate itself. Jin et al. [5] present memory compression mechanism to decrease total data transmitted. Hai Jin et al. [6] present a CPU scheduling approach to control memory update rate. Johan Tordsson et al. [7] present the dynamic page transfer reordering and compression mechanism and Bolin Hu et al. [8] present the K/N time-series mechanism. The two approaches are used to control total data transmitted of every round of iterative copy. However, these improvements are still based on Pre-copy and essentially dont break away from the migration pattern of Pre-copy. What is more, they fundamentally dont eliminate the dependency on iterative copy of dirty pages during the migration.

Post-copy [9] is implemented by Hines et al. It postpones memory migration until target VM begins running. Post-copy ensures that every memory page is only copied to target host once during the migration. However, in Post-copy algorithm memory migration is based on demand-paging, which makes target VM suspend and resume running frequently. Therefore, Post-copy has a performance problem. In addition, if some memory pages are never accessed by target VM, memory migration wont end within a long time. In the subsequent research, Hines et al. [10] present dynamic self-ballooning and pre-paging to improve its efficiency. However, these approaches just make Post-copy optimized in a certain range and dont resolve the existing problems.

Noack et al. present the idea of hybrid copy first in [11]. Its main idea is that source host performs total memory copy with source VM running. Subsequently source VM suspends running and transmits the situation about which memory pages are dirtied to target host. Then target VM resumes running and begins demand paging. The idea, which doesnt have a specific mechanism to be implemented, remains in the research stage. HMDC is a kind of specific implementation based on hybrid memory copy and delta compression.

In this paper, HMDC has two prerequisites: data transfer rate is less than network bandwidth; memory cache opened up by source host is large enough. We assume HMDC runs in the environment with enough network bandwidth. HMDC is affected by the size of memory cache. It is determined that the size of memory cache at least matches the set size of memory pages whose updating is the most frequent in VM. Pre-copy assumes that network transfer rate is faster than memory update rate. If not, dirty pages cant be converged during iterative copy. This will lead to that memory migration cant enter the next stage and fails. The prerequisite of Pre-copy is essential for it, whereas that of HMDC is only used to better show the performance.

3 The proposed HMDC Algorithm

3.1 Main idea

The main ideas are what combines hybrid memory copy with delta compression. It reduces total migration time, VM downtime and total transmitted data so that the efficiency is improved evidently.

Hybrid memory copy is what combines active push with demand paging to achieve fast memory copy. Memory migration is also divided into three stages by HMDC as illustrated in Fig.1. Active push of source host: in the first stage it transmits total memory to target host; in the second stage it transmits two bitmaps to target host with VM suspended; in the third stage source host periodically transmits dirty pages to target host. Demand paging of target host: it only works during dirty pages copy. The third stage begins with that target VM resumes running. Target host requests dirty pages from source host during target VM running. At the same time, pre-paging is integrated into demand paging. Delta compression

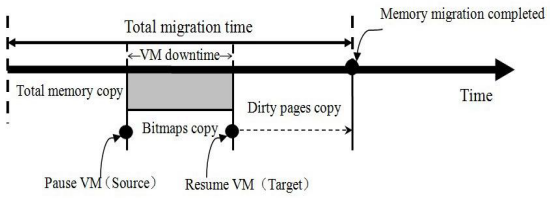

Fig. 1: The process of HMDC memory migration

works in the third stage of dirty pages copy. Its principle is that if in source host the old versions of dirty pages to

be transmitted are cached, source host will compute smaller delta compression pages of dirty pages and transmits them to target host. After target host receives the delta compression pages, will rebuild the new versions of dirty pages by using the delta compression pages.

3.2 Implementation

The implementation of three stages is as follows:

Stage 1(total memory copy): source host utilizes dynamic ballooning to recycle idle pages of source VM to reduce total data transmitted. Source host opens up memory cache of source VM. HMDC transmits total memory to target host with source VM running. During this process, if some page is to be overwritten, HMDC will copy its original data to memory cache at first. HMDC marks all pages dirtied to dirtypage_bitmap and idle dirty pages are marked using a special mark during total memory copy. While total memory copy is completed, Source VM stops running.

Stage 2(bitmaps copy): according to dirtypage_bitmap and memory cache, HMDC generates a new bitmap called cache_bitmap. Source host copies the two bitmaps to target VM. After target host receives them, performs some settings and then target VM begins running.

In HMDC,a novel approach, whose purpose is to resolve the contradiction between demand paging and delta compression, is proposed. This approach involves two bitmaps and a mapping table. The specific processes are as follows: after source VM stops running, source host looks up cache blocks of dirty pages marked in dirtypage_bitmap. If the old version of a dirty page is cached in memory cache, the flag bit of the memory page is denoted by ”1”. If not, is denoted by ”0”. After all dirty pages have been checked, source host generates a new bitmap called cache_bitmap, which marks all dirty pages whose old versions are cached in source host. Then source host copies the two bitmaps to target VM. As illustrated in Fig.2, after target VM receives them, according to the number of dirty pages marked in cache_bitmap, target host opens up memory cache and caches the corresponding memory pages to memory cache. At the same time, an AMT (address mapping table) to maintain the cache mapping is created. Finally, HMDC sets the lowest EPT (Extending Page Table) items of all dirty pages to non-present according to dirtypage_bitmap. This approach not only resolves the problem of the automatic page request of dirty pages but also provides original data for delta compression. Memory cache of source and target will be recycled once its data has been used so that the approach doesnt generate extra space overhead. Stage 3(dirty pages copy): Once target VM resumes running, source host begins pushing dirty pages periodically. HMDC sets a timer. If the timer times out, according to dirtypage_bitmap, HMDC copies non-idle dirty pages to a pushing queue until it is full or dirty pages are exhausted. After a process of active push is

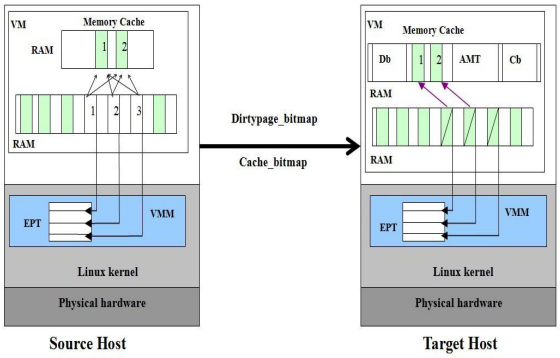

Fig. 2: The process of bitmaps copy

completed, the timer restarts. If during timing receiving a page request, source host immediately suspends the timer and copies the page requested to the pushing queue. Subsequently according to pre-paging, also copies its left and right neighbor dirty pages to the pushing queue until it is full or the dirty pages are exhausted. The timer resumes timing after sending the queue. If the dirty pages to be transmitted have the cache of old versions, HMDC performs delta compression on the dirty pages. HMDC firstly computes the delta page by applying XOR on the current and previous version of a page, and then get the delta compression page by compressing the delta page using XOR binary RLE algorithm [12]. Finally, a delta compression flag is set in the page header. Source host replaces dirty pages with their delta compression pages and sends the queue to target host. Finally, the corresponding cache blocks are recycled by VMM (virtual machine monitor). If the old versions of pages dont exist in memory cache, source host directly transmits dirty pages to target host. Delta compression should consume a minimum of CPU resources, both for cache hits and cache misses to not slow down the migration process or disrupt the performance of VM. Therefore, we employ a 2-way set associative caching scheme.

At the same time when target VM resumes running, target host begins performing demand paging. While the pages marked in dirtypage_bitmap are accessed, memory access faults will occur and then be fallen into VMM kernel to be captured by HMDC. If the page isnt an idle dirty page, with target VM suspended HMDC will send a page request to source host. Source host transmits a set of pages which include the requested page to target host as its response. Target host receives the response and updates memory pages accordingly. Then target VM resumes running. If the requested page is an idle dirty page, target host doesnt send the request to source host but directly allocates a memory page to target VM from the local. Subsequently target VM immediately resumes running. During target VM running, target VM will receive dirty pages from source VM periodically. For target VM, both

the response of a dirty page request and the active push of source host are checked whether they have delta compression flags. If yes, HMDC firstly decompresses the pages to get delta pages and then rebuilds the new pages by applying XOR on the delta pages and the old versions cached in target host. HMDC updates the memory pages using the new rebuilt pages and recycles the cache blocks to VMM. If no, HMDC directly updates the pages. According to dirtypage_bitmap, while all dirty pages have been synchronized, memory migration ends.

Let us consider pre-paging above in another view. It is also a K-means clustering process. The reason is: firstly we can regard the set of dirty pages which a pre-paging process refers to as a cluster and regard all dirty pages which are transmitted from source to target host through pre-paging as N data objects of a K-means clustering. During memory migration, K dirty pages to cause demand paging are K clustering central points of the initial clusters. Pre-paging is the process that it copies dirty pages, which relate to the page requested, to target host in order to reduce subsequent demand paging. It is a clustering process to decide which of dirty pages are carried by which one of all requested pages of demand paging. If we regard K pre-paging processes as a whole, exactly it is a K-means clustering.

Starting from this intuition, HMDC gives consideration to both total migration time and VM downtime to balance the migration process so that HMDC has a high availability. Our starting point is what combines the advantages of Pre-copy with the advantages of Post-copy, abandons their drawbacks and further has complementary advantages to implement a new migration framework with hybrid memory copy. On beginning memory migration, with source VM running, HMDC copies total memory pages of source VM to target host in order to make most memory pages synchronized through once copy. Within VM downtime, to be copied and sent is only two small bitmaps, whose sizes dont change under the different workload conditions. In order to reducing redundant traffic and thus to increase migration throughput, we employ the idea of delta compression to transmit the dirty pages. The dirty pages copy is finished by source and target host together to utilize the computing power of target to improve the efficiency and reduce migration time. During memory migration, it is consistent with original intension. The overhead of an XOR operation is negligible. Moreover, the XOR binary RLE compression algorithm, whose overhead is quite small, is well-known to be fit to efficiently compress memory pages with the binary characteristic. All the features guarantee the feasibility and superiority of HMDC.

At the beginning of total memory copy, there are some idle pages in source VM. We should take measures to deal with them. The way is an open issue. In HMDC, we employ dynamic ballooning mechanism to recycle the idle pages to VMM in order to decrease total migration data.

At the beginning of total memory copy, there are some idle pages in source VM. We should take measures to deal with them. The way is an open issue. In HMDC, we employ dynamic ballooning mechanism to recycle the idle pages to VMM in order to decrease total migration data.

At this time when HMDC sets the mapping states to non-present in the target host, target host needs to maintain a copy of dirtypage_bitmap in VMM kernel to avoid that the state is frequently switched between kernel and user state to look over the bitmap. What is more, the copy is used to determine whether memory migration is finished. Once target host receives dirty pages, HMDC updates the bitmap copy. And then HMDC checks it. If it shows that all dirty pages are synchronized successfully, target host sends a signal which indicates memory migration is finished to source host. According to the signal and the bitmap of dirtypage_bitmap which has been kept in source host, source host judges whether memory migration is completed. If it shows all dirty pages are transmitted completely, source host sends an acknowledgement to target host. After target host receives, memory migration ends.

4 Evaluation

In this section, we will experimentally verify the proposed HMDC algorithm. On XEN [13] platform, we compare HMDC with Pre-copy and XBRLE [14]which combines delta compression with Pre-copy by total migration time, VM downtime and total data transmitted. We prepare four different kinds of test cases of VMs. The results demonstrate that HMDC reduces not only total migration data and the loss of the VM performance but also migration time compared to Pre-copy and XBRLE. Especially for the test case under the intensive workload, HMDC evidently shows the stability and high-efficiency.

4.1 Experimental Scenarios

The VMs to migrate is as follows:

- –The VM of 1GB RAM, 1 VCPU, fast Ethernet, running two instances of the LMbench [15] memory write benchmark of 256 MB each;
- –The VM of 1GB RAM, 1 VCPU, fast Ethernet, running a HD video transcoding server;
- –The VM of 8GB RAM, 4 VCPUs, Gigabit Ethernet, running a SAP CI ERP system server with one hundred concurrent users;
- –The VM of 1GB RAM, 1 VCPU, Gigabit Ethernet, running a apache2 server with 50 concurrent users downloading the file.

The ERP case is performed on two Intel 3GHz 4x Dual Core Xeon servers with 32GB of RAM running

Ubuntu10.4. The size of 2-way set associative cache is 1GB. Other cases are performed on two Intel 2.66 GHz core2quad servers with 16GB of RAM running Ubuntu9. The size of 2-way set associative cache is 512MB. All VM disks are stored on an iSCSI server with 16GB of RAM and gigabit Ethernet card. All VMs are connected to Ethernet using bridge connection. Linux kernel version is 2.6.32-24. Xen version is 3.4.4. Every test is run ten times and the result is the average of ten times.

4.2 Live migration under memory press workload

Aiming to verify the efficiency and availability of HMDC under different memory press workload, a benchmark is used. It continuously updates memory pages by using the LMbench memory write tool in a specified rate. After 15 seconds, VM begins migrating. As illustrated in Fig.3, when the rate increases from 0 to 10240p/s, the performance of Pre-copy will decline sharply. Its total migration time rises from the initial 8.26s to 185.75s. In the tests, it is found that when memory update rate is greater than 15978p/s, the migration of pre-copy cant be finished correctly and even fails. Thus, we only give the experimental results under this case that memory update rate is less than 10240p/s. The above results show that

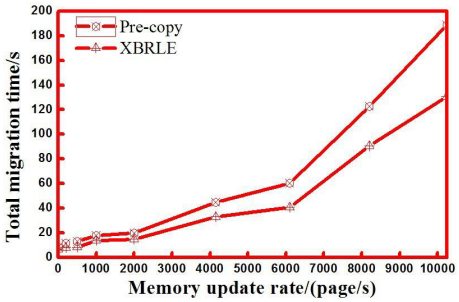

Fig. 3: Total migration time of Pre-copy and XBRLE with memory update rate rising

memory update rate evidently restricts the migration performance of pre-copy. As illustrated in Fig.4, the VM downtime of pre-copy is roughly the same under the different workload because of being limited by the minimum remaining dirty pages. However, the stability is on this condition that the remaining dirty pages can converge and at the cost of extending the total migration time. XBRLE employs delta compression, so both VM downtime and total migration time are shorter than those of pre-copy. However, as mentioned earlier, the improvement is limited. On the contrary, HMDC doesnt rely on iterative copy. Although memory update rate is much faster than network transfer rate, HMDC also has

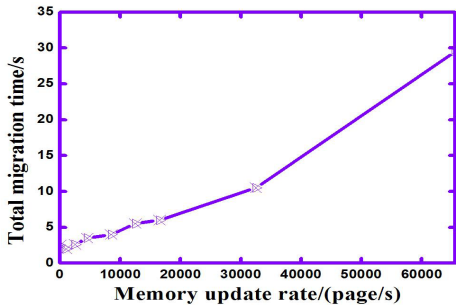

Fig. 4: Total migration time of HMDC with memory update rate rising

the abilities in completing the migration efficiently. As illustrated in Fig.4, when the rate rises to 65536p/s, the average migration time is 29.5s. Fig.5 shows the comparison on VM downtime. We find that VM downtime of HMDC is significantly shorter, while the rate is small. This is because HMDC only needs to generate and transmit two bitmaps and the delay of setting cache and mapping states is shorter than that of transmitting the rest of dirty pages of the other two. With the rate further rising, the number of dirty pages increases and the delay increases in VM downtime. When the rate rises to a certain level, VM downtime of HMDC becomes longer. However, we find an interesting feature in the tests. Even if the rate rises to 65536p/s, HMDC only causes 280ms of downtime. As illustrated in Fig.6, we give the comparison about total migration data and can see it clear that with the rate rising, the gap between HMDC and the other two is widened rapidly. The reason is that with memory update rate rising, for pre-copy and XBRLE the number of memory pages which are repeatedly copied sharp increases during the migration, whereas for HMDC not only dirty pages as one part of memory pages merely are copied additional once but also are performed delta compression handling. Moreover, the result of dealing with the idle pages and the idle dirty pages further decrease total data transmitted.

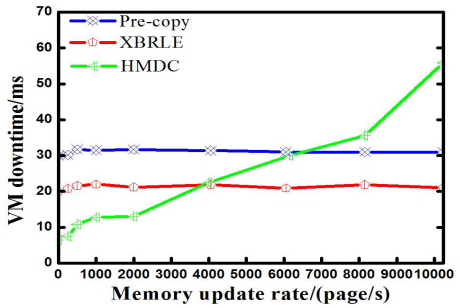

Fig. 5: VM downtime with memory update rate rising

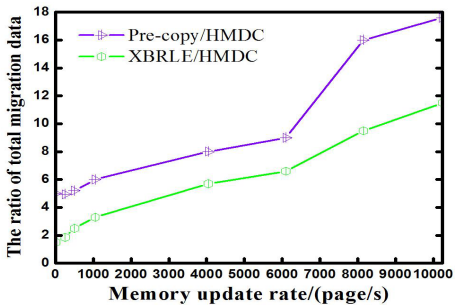

Fig. 6: The ratio of migration data with memory update rate rising

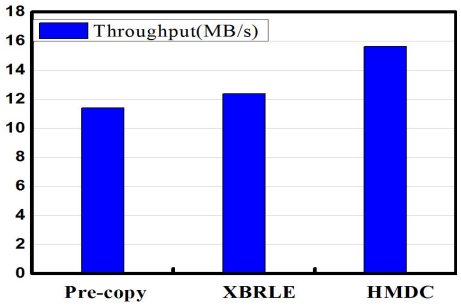

Fig. 8: Migration throughput

4.3 Live migration of HD video transcoding server VM

The video service is a typical service that is widely used in the network. Its memory update rate is quite great. The data is not very compressible as the content of the streaming buffer changes constantly and the difference between subsequent video frames can be large. So delta compression cant fully show its advantage in the case. However, HMDC employs the way to combining pushing with pulling and integrates the idle pages optimizing into our algorithm, so HMDC still reduces migration time and increases the migration throughput evidently. As illustrated in Fig.7 and Fig.8, compared to Pre-copy, the total migration time is decreased from 26s to 15s and the VM downtime is decreased from 2s to 28ms and the migration throughput is increased from 11.39MB/s to 15.63MB/s; compared to XBRLE, the total migration time is decreased from 28s to 15s and the VM downtime is decreased from 0.9s to 28ms and the migration throughput is increased from 12.38MB/s to 15.63MB/s.

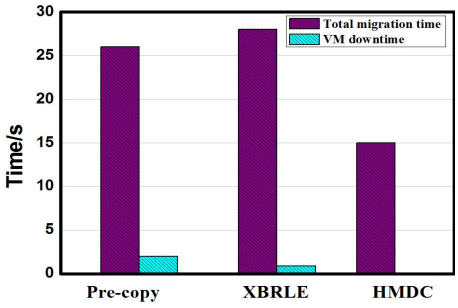

Fig. 7: Migration time

4.4 Live migration of ERP system VM

Aiming to further test its performance, we select the VM running a SAP CI ERP system server as the migration object. It has a CPU-intensive and memory-intensive workload. In addition, it is sensitive to network time-out since the ERP system depends on transactions. As illustrated in Fig.9, compared to Pre-copy, total migration time is decreased from 235s to 95s and the VM downtime is increased from 3s to 3.8s; compared to XBRLE, the total migration time is decreased from 139s to 95s and the VM downtime is increased from 1.5s to 3.8s. The ERP system VM is considered notoriously hard to migrate. In the tests HMDC reduces total migration time evidently. For VM downtime, as mentioned earlier, the downtime of HMDC is longer. However, it should be clarified that the migration using Pre-copy has failed for several times and total migration time of Pre-copy and XBRLE is longer than that of HMDC. After all, VM downtime is one part of total migration time. If the downtime doesnt seriously affect the performance of VM itself, total migration time is more important. Therefore, the performance and efficiency of HMDC is much better than that of the other two algorithms.

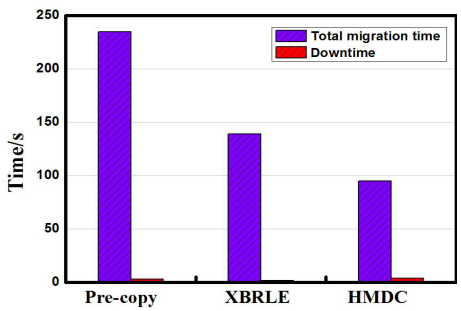

Fig. 9: Migration time

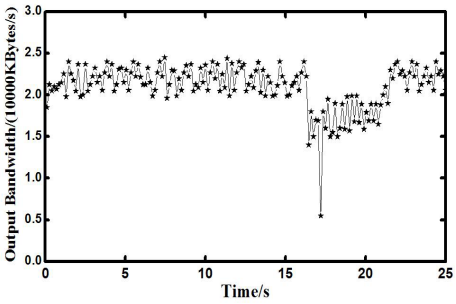

Fig. 10: Pre-copy

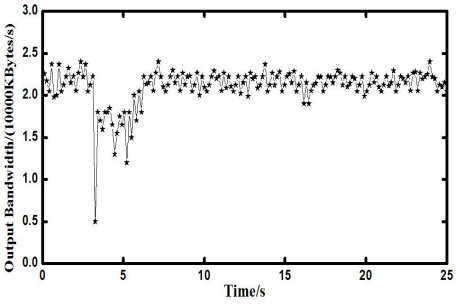

Fig. 11: XBRLE

4.5 Live migration of Apache2 server VM

We track and analyze the change of output bandwidth in this case of concurrent access. In the experiments, we simulate 50 concurrent clients on the hosts to download a 512KB static file continuously. As illustrated in Fig.10, Fig.11 and Fig.12, the three output bandwidth curves all have instantaneous decline and recovery. However, the duration of HMDC from the lowest bandwidth to the normal fluctuation is shorter. This reason is that before target VM begins running, in advance HMDC establishes the address mapping for dirty pages in the EPT tables by setting the non-present states for dirty pages. When later VM again accesses the pages, doesnt need to again enter the kernel to deal with the mapping states. For Pre-copy and XBRLE as the EPT tables are empty when the migration is finished, they need to dynamically establish EPT tables for memory pages. For this reason, after the migration ends, the time which it takes to resume VM performance completely is as follows. Pre-copy costs 6s; XBRLE costs 4s; HMDC merely costs 2s. The above results demonstrate even though VM has a high workload of concurrent I/O, HMDC still can make VM applications have a more stable and high-efficient I/O performance.

5 Conclusion and future work

In this paper, a novel memory migration algorithm HMDC is proposed and we give its main idea, implementation and evaluation. It employs hybrid memory copy and delta compression. The hybrid memory copy makes full use of the computing power of two sides by active push and demand paging to achieve fast memory migration. During dirty pages migration, we employ delta compression to increase the migration throughput and improve the migration efficiency. While source host responds to the page request of target host, will also copy its neighbor dirty pages to target host together. Pre-paging reduces the number of demand paging to guarantee the performance of target VM and quicken dirty memory migration. HMDC achieves the efficiency and transparency of memory migration. It not only evidently shortens migration time but also increases the migration throughput, decreases total data transmitted and protects the performance of VM running. The final experimental results show that HMDC is an effective memory migration algorithm.

Aiming to further improve the performance of HMDC, we plan to study the robustness of HMDC in the next step work. If the power outage or crash, etc. happens, HMDC should have abilities in recovering the original VM. In this paper, HMDC is used for memory migration in LAN. In the future, we will extend HMDC in WAN.

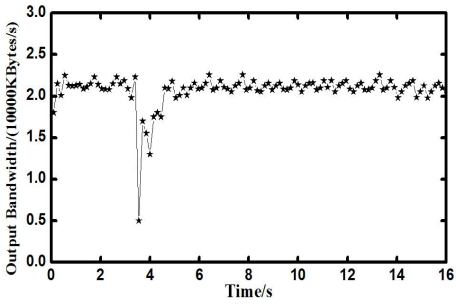

Fig. 12: HMDC

Acknowledgement

The authors would like to thank the editors and anonymous reviewers for their valuable comments.

References

[1] M. Armbrust, A. Fox, R. Griffith, A. Joseph, R. Katz, A. Konwinski, G. Lee, D. Patterson, A. Rabkin, I. Stoica and M. Zaharia, Above the clouds: A Berkeley view of cloud computing, Technical Report EECS-2009-28 (2009).

[2] P. Barham, B. Dragovic, K. Fraser, S. Hand, T. Harris, A. Ho, R. Neugebauer, I. Pratt and A. Warfield, Xen and the Art of Virtualization, In Proc. of the 19th ACM Symposium on Operating Systems Principles, 164-177 (2003).

[3] Y. Li, W. Li and C. Jiang, A survey of virtual machine system: Current technology and future trends, Electronic Commerce and Security (ISECS), 2010 Third International Symposium, 332-336 (2010).
[4] C. Clark, K. Fraser, S. Hand, J. Hansen, E. Jul, C. Limpach, I. Pratt and A. Warfield, Live migration of virtual machines, In Proc. of the 2nd Conference on Symposium on Networked Systems Design & Implementation, 273-286 (2005).
[5] H. Jin, L. Deng, S. Wu, X. Shi and X. Pan X, Live Virtual Machine Migration with Adaptive Memory Compression, IEEE International Conference on Cluster Computing (2009).
[6] H. Jin, W. Gao, S. Wu, X. Shi, X. Wu and F. Zhou, Optimizing the live migration of virtual machine by CPU scheduling, Journal of Network and Computer Applications, 1088-1096 (2011).
[7] P. Svärd, J. Tordsson, B. Hudzia and E. Elmroth, High performance live migration through dynamic page transfer reordering and compression, In Proc. of 2011 3rd IEEE International Conference on Cloud Computing Technology and Science, 542-548 (2011).
[8] B. Hu, Z. Lei, Y. Lei, D. Xu and J. Li, A Time-Series Based Precopy Approach for Live Migration of Virtual Machines, 2011 IEEE 17th International Conference on Parrallel and Distributed Systems, 947-952 (2011).
[9] MR. Hines, U. Deshpande and K. Gopalan, Post-copy live migration of virtual machines, SIGOPS Operating Systems Review, 14-26 (2009).
[10] MR. Hines and K. Gopalan, Post-Copy Based Live Virtual Machine Migration Using Adaptive Pre-Paging and Dynamic Self-Ballooning, In Proc. of the 2009 ACM SIGPLAN/SIGOPS International Conference on Virtual Execution Environments, 51-60 (2009).
[11] M. NOACK, Comparative evaluation of process migration algorithms, Masters thesis, Dresden University of Technology - Operating Systems Group (2003).
[12] D. Pountain, Run-length encoding. Byte, 317-319 (1987).
[13] XEN, http://xen.org, visited March (2012).
[14] P. Svard, B. Hudzia, J. Tordsson and E. Elmroth, Evaluation of delta compression techniques for efficient live migration of large virtual machines, In Proc. of the 7th ACM SIGPLAN/SIGOPS International Conference on Virtual Execution Environments, 111-120 (2011).
[15] L. McVoy, S. Graphics, C. Staelin and Hewlett-Packard Laboratories, Lmbench: Portable Tools for Performance Analysis, In Proc. of the USENIX 1996 Annual Technical Conference (1996).

Liang Hu was born in 1968. Currently, he is the professor and PhD supervisor of College of Computer Science and Technology, Jilin University, China. His main research interests include distributed systems, computer networks, communications technology and information security system, etc.

Jia Zhao was born in Changchun of Jilin province of China in 1982. Currently he is a PhD candidate of the college of computer science and technology of Jilin University. His main research interests include distributed system, cloud computing, network technology. He has participated in several projects.

Gaochao Xu was born in 1966. Currently, he is the professor and PhD supervisor of College of Computer Science and Technology, Jilin University, China. His main research interests include distributed system, grid computing, cloud computing, Internet of things, etc.

Yan Ding was born in Yichun , Heilongjiang , China in 1988. Now he is a postgraduate candidate of the college of computer science and technology of Jilin University. His main research interests include distributed system, cloud computing and virtualization technology.

Jianfeng Chu was born in 1978, Ph.D. , Now he is the teacher of the College of Computer Science and Technology, Jilin University, Changchun, China. His current research interests focus on information security and cryptology.