2011 International Conference on Network and Electronics Engineering
IPCSIT vol.11 (2011) © (2011) IACSIT Press, Singapore

Live Virtual Machine Migration with Efficient Working Set Prediction

Ei Phyu Zaw[1], Ni Lar Thein[2]

[1]University of Computer Studies, Yangon
Yangon, Myanmar

Abstract. Live migration is a key feature of system virtualization technologies. VM live migration basically consists of transferring its memory pages from a source server to a destination server. The amount of transferred memory pages affected the downtime and application performance of virtual machines. Live migration techniques in the state of the art mainly use pre-copy approach. In pre-copy based live migration, virtual machine's service downtime is expected to be minimal but total migration time is prolonged by iterative copy operations and the significant amount of transferred data during the whole migration process. In this paper, a framework for reducing the amount of transferred data in pre-copy based live migration is presented. In the framework, a working set prediction algorithm is proposed as a preprocessing step. The proposed working set prediction algorithm which based on Least Recently Used (LRU) policy consists of working set list that collects the most recent used memory pages and inactive list that collects the least recent used memory pages. In iterative copy operations of the proposed framework, it transfers only inactive list and in stop and copy operation, it transfers working set list and CPU state. Compared with pre-copy based live migration, the proposed framework can reduce the total migration time.

Keywords: Least Recently Used (LRU); Migration Time; Pre-copy based live migration; Virtual Machine;

1. Introduction

Today's IT departments are under increasing pressure to manage and support expanding computer resources while at the same time reducing costs. Virtualization technology, which lets multiple operating systems run concurrently on the same physical server, has become a broadly accepted method to meet these requirements. By converting under-utilized physical servers into virtual machines that run on a single physical server, organizations can reduce space, power and hardware costs in the data center. Because virtual machines are generally much faster to recover in a disaster than physical computers are, virtualization also increases server uptime and reliability.

Migrating operating system instances across distinct physical hosts is a useful tool for virtualization technology. It allows a clean separation between hardware and software, and facilitates fault management. Live migration of virtual machines is a useful capability of virtualized clusters and data centers. It can be done by performing while the operating system is still running. It allows more flexible management of available physical resources by making to load balance and do infrastructure maintenance without entirely compromising application availability and responsiveness.

The best technique for live migration of virtual machines is pre-copy [2]. It incorporates iterative push phases and a stop-and-copy phase which lasts for a very short duration. By 'iterative', pre-copying occurs in rounds in which the pages to be transferred during round n are those that are modified during round n-1. The number of rounds in pre-copy migration is directly related to the working set which are being updated so frequently pages. The final phase stop the virtual machine, copies the working set and CPU state to the destination host. The issue of pre-copy based live migration is that total migration time is prolonged [2].

In this paper, we propose the algorithm to predict the working set, the collection of recent used memory pages, in pre-copy based migration for virtual machines and then define working set list and inactive list. We also present the proposed framework for pre-copy based live migration to reduce the total migration time. From the testing result, we can reduce total migration time of virtual machines.

The rest of this paper is organized as follows. Section 2 describes the related work. In Section 3, we discuss the live migration of virtual machines. In Section 4, the proposed framework of pre-copy based live migration and calculation result are described. Finally, Section 5 describes the conclusion of this paper.

2. Related Work

The goal of Pre-copy based migration [2] is to keep downtime small by minimizing the amount of VM state that needs to be transferred during downtime. Pre-copy minimizes two metrics particularly well VM downtime and application degradation but it can reduce pre-copy's effectiveness and increase total migration time during migration process because pages that are repeatedly dirtied may have to be transmitted multiple times.

Many hypervisor-based approaches such as VMware [2], XEN [4] and KVM [5] is used the pre-copy approach for live migration of virtual machines. Remote Direct Memory Access (RDMA) [1] offers performance increase in VM migration by avoiding TCP/IP stack processing overhead. RDMA implements a different transfer protocol, where origin and destination VM buffers must be registered before any transfer operations, reducing it to "one sided" interface. Data communication over RDMA does not need to involve CPU, caches, or context switches.

In post-copy live migration of virtual machines [3], 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 pre-copy due to the latency of fetching pages from the source node before VM can be resumed on the target.

MECOM [9] first introduces memory compression technique into live VM migration. Based on memory page characteristics, MECOM design a particular memory compression algorithm for live migration of VMs. The total migration time and downtime are both decreased significantly because the smaller amount of data is transferred but compression operations cause performance bottleneck of live migration.

3. Background Theory

3.1. Live Migration of Virtual Machines

Virtual machine migration takes a running virtual machine and moves it from one physical machine to another. This migration must be transparent to the guest operating system, applications running on the operating system, and remote clients of the virtual machine. Live Migration approach migrate the OS instances including the applications that they are running to alternative virtual machines freeing the original virtual machine for maintenance. A key challenge in managing the live migration of OS instances is how to manage the resources which include networking, storage devices and memory.

Networking: In order for a migration to be transparent all network connections that were open before a migration must remain open after the migration completes. To address these requirements, the network interfaces of the source and destination machines typically exist on a single switched LAN.

Storage devices: We rely on storage area networks (SAN) or NAS to allow us to migrate connections to storage devices. This allows us to migrate a disk by reconnecting to the disk on the destination machine.

Memory: Memory migration is one of the most important aspects of Virtual machine migration. Moving the memory instance of the VM from one physical state to another can be approached in any number of ways. The memory migration in general uses one or two phases from the push phase, stop and copy phase and pull phase.

3.2. Least Recently Used (LRU) Replacement Algorithm

It associates with each page the time of that page's last use. When a page must be replaced, LRU chooses the page that has not been used for the longest period of time. The problem is to determine and order for the

fames defined by the time of last use. Two implementations are feasible: Counters and Stack. We apply Stack implementation that keeps a stack of page numbers and most recent memory page move to the top. So, the memory pages used in the recent past is always on the top of the stack. By dynamically monitoring memory accesses and constructing the LRU list, we can predict the Working Set list of a virtual machine.

4. Proposed Framework for Pre-copy based Live Migration

In this Section, we present the proposed framework for pre-copy based live migration to reduce the total migration time. The proposed framework consists of the following phases as shown in Fig. 1.

(1) ***Preprocessing Phase:*** The system applies the proposed working set prediction algorithm as the pre-processing phase. This algorithm is based on Least-recently-used (LRU) replacement algorithm and collects the most recent used memory page as the working set list and collects the least recent used memory pages as the inactive list.

(2) ***Push Phase:*** The system transfers memory pages except working set list in first iteration By 'iterative', pre-copying occurs in rounds in which the pages to be transferred during round n are those that are modified during round n-1.

(3) ***Stop and Copy Phase:*** This phase consists of four steps. The system suspends the source virtual machine for a final transfer round and transfers last modified pages and CPU state. Then it discards the source VM and activates the Target VM.

The design involves iteration though multiple rounds of copying in which the VM memory pages that have been modified since the previous copy are resent to the destination. The main contribution of the paper is that to propose a working set prediction algorithm to minimize the amount of iterations and reduce the total migration time.

In the source VM, the proposed framework applies the prediction algorithm for working set as the pre-processing step. It predicts and defines the working set which is the collection of memory pages in future use to reduce not only the amount of modified pages during the push phase but also the rounds of copying. Prediction time is performance bottleneck of addition overhead introduced by prediction operation. So, the proposed algorithm uses LRU (Least Recent Used)-based prediction method to get the prediction operation without a notable execution time.

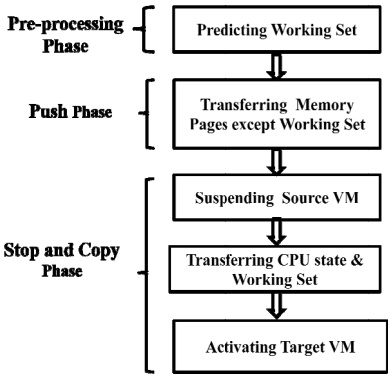

Fig. 1: Proposed Framework for pre-copy based live migration

4.1. Working Set Prediction Algorithm

In pre-copy based live migration, the system first transfers all memory pages and then copies pages just modified during the last round iteratively. VM service downtime is expected to be minimal by iterative copy operations but total migration time is prolonged that caused by the significant amount of transferred data during the whole migration process.

We propose the working set prediction algorithm in the framework which can reduce the amount of transferred memory pages in pre-copy based migration process. The proposed algorithm predicts the most recently used memory pages that directly affects the total migration time.

```
if (Request page j)
 {
        if (! Working Set list is full)
        {
                place the page j in the Working Set  using the linear order;
        }
        else
        {
                if (request page j is not in Working Set)
                {
                        remove page i whose is located at the end of the Working Set to the inactive list;
                        insert request page j at the top of the Working Set;
                }
                else
                {
                        find request page j in Working Set and move to the top of Working Set;
                }
        }
 }
```

Fig. 2: Proposed Working Set Prediction Algorithm

In the proposed algorithm as shown in Fig. 2, memory of virtual machine is divided into two lists where the most recently used pages are placed in the Working Set list while least recently used pages are in inactive list. When a new page is faulted into main memory, it is placed in the Working Set list. If the Working Set list is full, the least recently used page is moved into the inactive list to place the most recent used page. If the inactive list is full that is the memory of virtual machine is full, the least recent page of inactive list is discarded. The Working Set list is managed by least recently used (LRU) algorithm. The inactive list is ordered by each page's time of removal from the Working Set list. When any reference to the inactive list forces to cause a page fault, the VM moves it into the Working Set list.

The proposed algorithm organizes the Working Set list as a doubly linked list where each node has two pointers. Whenever a memory page is requested, three operations will be performed on the Working Set list. Firstly, the proposed algorithm finds the requested page in the Working Set list. If the requested page is already exist in Working Set list, move the requested page to the top of the Working Set list. If the requested page is not exist in the Working Set list, delete the page which locate at the end of the Working Set list and insert the requested page at the top of the Working Set list.

4.2. Calculation Results

In this section, we describe the calculation of the total migration time in the propose framework and pre-copy based live migration. Assuming that the memory could be processed in the propose framework as pages with a fixed size equal to $\boldsymbol{P}$ bytes, $\boldsymbol{M}$ is the memory size, the bandwidth available for the transfer is constant and equal to $\boldsymbol{B}$ bytes per second, $\boldsymbol{W_S}$ is the working set list and $\boldsymbol{W_M}$ is the modified page per second.

Therefore, the time cost to transfer the memory pages in the proposed framework is as follows

Table. 1: Annotation for Calculation Results

$\boldsymbol{P}$	Page Size (4KB)
$\boldsymbol{M}$	Memory Size (131072 page or 512 MB)
$\boldsymbol{B}$	Transfer Rate (5.625 bytes per second for slow connection)
$\boldsymbol{W_M}$	Modified Pages per second (512 pages per second)
$\boldsymbol{W_S}$	Size of Working Set list
$\boldsymbol{T_i}$	Execution time of i^{th} iteration for *push phase*
$\boldsymbol{T_{stop}}$	Execution time for *stop phase*
$\boldsymbol{T_{pre}}$	Execution Time for *preprocessing phase*
$\boldsymbol{T_{total}}$	Total Migration Time

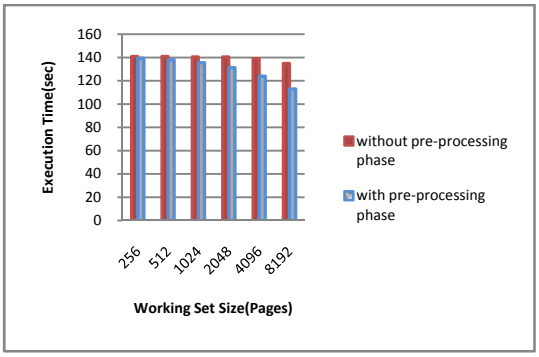

Fig. 3. Execution Time for Push Phase of the Proposed Framework

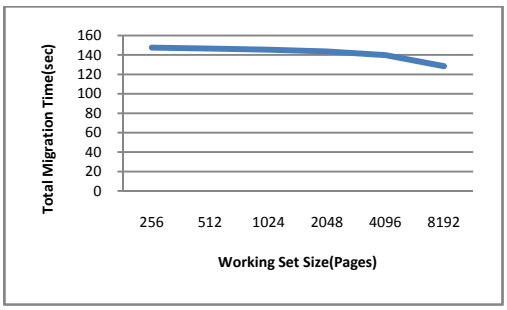

Fig. 4: Total Migration Time for Proposed Framework

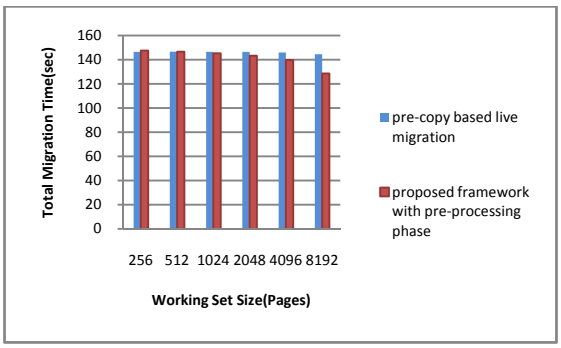

Fig. 5: Total Migration Time of the proposed framework with pre-processing phase Vs pre-copy based live migration

Execution time of i[th] iteration for *push phase* of the proposed framework:

$$T_i = \frac{(M_i - W_s)P}{B} \quad (1)$$

where i=1,2,3,…n and $M_n \leq W_M$ and $M_1 = M$ for first iteration and then modified memory page size is updated by:

$$M_{i+1} = T_i W \quad (2)$$

Execution time for *stop phase* of the proposed framework:

$$T_{stop} = \frac{W_s P}{B} \quad (3)$$

Total Migration Time of Proposed Framework = $T_{pre} + \sum_i^n T_i + T_{stop}$ (4)

And the time cost to transfer the memory pages in pre-copy based live migration is as follows:

Execution time of i[th] iteration for *push phase* of pre-copy based live migration:

$$T_i = \frac{M_i P}{B} \quad (5)$$

where i=1,2,3,…n and $M_n \leq W_M$ and $M_1 = M$ for first iteration and then modified memory page size is updated by:

$$M_{i+1} = T_i W_M \quad (6)$$

Execution time for *stop phase* of Pre-copy based live migration:

$$T_{stop} = \frac{W_s P}{B} \quad (7)$$

Total Migration Time of Pre-copy live migration = $\sum_i^n T_i + T_{stop}$ (8)

In push phase of the proposed framework as shown in (1), the system first transfer the memory page except the Working Set list W_s that include the most recent pages during the migration time. In (5) for the push phase of the pre-copy based live migration, the system first transfers the all of the memory pages. The proposed framework reduces the transfer memory page using the working set list during the push phase of live migration. But the total migration of the proposed framework include not only the execution time for push phases and stop and copy phase but also the overhead for pre-processing phase that apply the working set prediction algorithm.

From the equations, we give a specific, numerical example to evaluate the total migration time of the proposed framework. Based on the experiment with Quake3 Server workload, we make the assumption that before migration as shown in Table. 1. Then we have the results in Figures. From these results, the proposed framework can migrate a virtual machine in less time than the pre-copy based live migration.

In Fig. 3, we present results comparing the execution time for push phase of the proposed framework with pre-processing phase and without pre-processing phase using various working set size. Although we need less execution time for push phase with pre-processing phase, the system has the overhead for pre-processing phase.

In Fig. 4, the calculation results show the total migration time of the proposed framework with various working set size. According to the calculation results, 32 MB (8192 pages) is suitable for Working Set list of the proposed framework. If the size of Working Set list increased, the execution time for pre-processing phase also increased. The system need to be balance the effect and overhead using pre-processing phase. Fig. 5 shows the comparison of total migration time for the proposed framework with pre-processing phase and pre-copy based live migration using various working set size. From these calculation results, we can reduce the total migration time in the proposed framework than pre-copy based live migration of virtual machine while the working set size is 1024 pages and above. In 256 pages of working set, the preprocessing phase can be overhead for total migration time

5. Conclusion

In this paper, we have presented the framework which includes the pre-processing phase, push phase and stop phase for pre-copy based live migration. Pre-copy based live migration ensures that keep downtime small by minimizing the amount of VM state. It provides to abort the migration should the destination VM ever crash during migration because the VM is still running at the source. It needs to reduce the total migration time and impact of migration on performance of source and destination VMs. We proposed the working set prediction algorithm which based on Least Recent Used (LRU) policy collects the most recent used memory pages as the Working Set list and collects the least recent used memory pages as the inactive list in pre-processing phase. The proposed framework migrate the inactive list in push phase and lastly it migrate the Working set list in stop phase to reduce the amount of transferred data and iteration times. From the calculation results, the proposed framework can reduce the total migration time in live migration of virtual machines.

6. References

[1] W. Huang, Q. Gao, J. Liu, and D. K. Panda, “High Performance Virtual Machine Migration with RDMA over Modern Interconnects”, Proceedings of IEEE Conference on Cluster Computing (Cluster 2007), Austin, Texas. September, 2007

[2] C. Clark, K. Fraser, S. Hand, J. G. Hansen, E. July, C. Limpach, I. Pratt, and A. Warfield, “Live Migration of Virtual Machines”, Proceedings of the 2nd USENIX Symposium on Networked Systems Design and Implementation, 2005

[3] 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”, Proceedings of the Nineteenth ACM Symposium Operating System Principles (SOSP19), pages 164.177. ACM Press, 2003.

[4] M. Nelson, B. Lim, and G. Hutchines, “Fast transparent migration for virtual machines,” in Proceedings of the USENIX Annual Technical Conference (USENIX’05), 2005, pp. 391– 394.

[5] A. Kivity, Y. Kamay, and D. Laor, “kvm: the linux virtual machine monitor”. In Proc. of Ottawa Linux Symposium (2007).

[6] M. R. Hines and K. Gopalan, “Post-copy based live virtual machine migration using adaptive pre-paging and dynamic self-ballooning”, in Proceedings of the ACM/Usenix international conference on Virtual execution environments (VEE’09), 2009, pp. 51–60

[7] R. Goldberg, “Survey of virtual machine research,” IEEE Computer, pp. 34–45, Jun. 1974.

[8] P. Lu and K. Shen, “Virtual machine memory access tracing with hypervisor exclusive cache”. In ATC’07: 2007 USENIX Annual Technical Conference on Proceedings of the USENIX Annual Technical Conference,pages 1–15, Berkeley, CA, USA, 2007. USENIX Association. ISBN 999-8888-77-6.

[9] H. Jin, L. Deng, S. Wu, X. Shi, and X. Pan. Live virtual machine migration with adaptive memory compression. In Proceedings of the 2009 IEEE International Conference on Cluster Computing (Cluster 2009), 2009

[10] T. Yang, E. Berger, M. Hertz, S. Kaplan, and J. Moss. Automatic heap sizing: Taking real memory into account, 2004. URL citeseer.ist.psu.edu/article/yang04automatic.html.