KVM 가상화 환경에서의 디스크 캐시와 메모리 스냅샷을 이용한 효율적인 라이브 마이그레이션 방법

박광용O, 오영섭, 성백재, 박찬익
포항공과대학교
{tmipyiong@postech.ac.kr, youngsup@postech.ac.kr, jays@postech.ac.kr. cipark@postech.ac.kr}

An Efficient Live Migration using Disk Cache and Memory Snapshot in KVM Virtualization Environment

Guang-Yong PiaoO, Young-Sup Oh, Baeg-Jae Sung, Chan-Ik Park
POSTECH

Abstract

Virtual machine live migration technique is an essential feature of virtualization technique and widely used in many cloud computing services. Reducing the virtual machine live migration time is very important because it consumes lots of computing and network resources. We proposed an efficient virtual machine (VM) live migration technique, by utilizing a snapshot of previous VM memory, and a disk cache which consists of frequently used disk data blocks. We implemented the novel migration technique on KVM, and reduced the virtual machine live migration time significantly.

1. Introduction

Cloud computing services are developing rapidly in recent years. The key of those services is virtualization technique, which allows multiple operating systems (OS) running on a physical machine concurrently. The crash of any Guest VM does not affect other Guest VMs running on the same physical machine, so the virtualization technique provides a strong isolation as well as the physical resource reusability.

Virtual machine live migration is an essential feature of virtualization technique. With this feature, virtual machines can be migrated among physical hosts without service disruption. So in the cloud environment, virtual machine live migration is used for load balancing, server consolidation, fault tolerance and energy saving [1, 2]. One of the cloud computing services, which highly depend on the VM live migration feature, is desk top as a service (DaaS) model [3]. In DaaS environment, the desktop VM is required to be migrated from a remote data center to a local data center or directly to the user’s physical machine via wide area network (WAN) to improve user experience. Because of the limited bandwidth of WAN, the migration would take more time than it processed inside a data center. When the VM migration last for a long time, a huge number of memory pages would become dirty, and transferring those memory pages consume extra network and computing resources. The goal of this paper is reducing the migration time for VMs which migrate back and forth frequently among hosts under a low network bandwidth.

In this paper, we proposed a novel live migration method, by utilizing a memory snapshot of previous VM memory and a disk cache which consists of frequently accessed applications’ binaries on virtual machine environment, and implemented the proposed idea on KVM.

The experimental result showed that our proposed method outperforms original KVM migration method by three times, when the VM migrate back to the original source after running 30 minutes on destination host.

The rest of the paper is organized as follows. Section 2 analyzes characteristics of memory to prove the possibility of our migration method. Section 3 explains the detailed design of our proposed migration technique. Section 4 shows the effectiveness of the novel migration method by presenting experimental results. Finally, Section 5 gives a brief summary of this paper.

2. The Characteristics of Memory Content

2.1 Duplication between Memory and Disk

Modern operating systems cache recently accessed disk data or read-only data blocks on the memory region (page cache in Linux), to reduce the latency of accessing the same data blocks from disk for multiple times. Besides, page prefetching techniques (Windows: superfetch, Linux: preload), which implemented in many operating systems for improving the interactive performance, additionally fetch more disk blocks into the memory [4, 5]. Because these cached blocks are aligned by a page size (4KB in common case) in the memory, if we can figure out the frequently used data blocks, and statically maintain these blocks in both sides, a mapping information between the memory page-frame number (PFN) and the offset of the duplicated data blocks inside the set of disk blocks, is enough for transferring a duplicated memory page. Therefore the transferring data size for those duplicated pages can be reduced from 4KB to 16Byte (64bit address, 64bit offset).

2.2 Duplication between Current and Previous Memory

After a VM live migration just ended up, the memory content of source and destination are the same, and this memory snapshot is easily to be maintained in both sides. When the VM wants migrate back to the original host after running for a time period, we can classify the current memory pages into three types, based on the relationship between the current memory page and the previous memory page with the same PFN: 1) identical; 2) similar; 3) different. For page type 1, we can reduce the transferring data size by only sending the PFN. Besides, for page type 2, the memory pages can be reconstructed on destination side, with the PFN and diff information. XBZRLE [6] is an efficient delta compression method which is suitable for memory live migration. To observe the amount of memory saved size by delta decompression method that described above, we performed following experiment. Ubuntu VM with 2GB memory ran a desktop workload (mix of office, movie and web browsing) for 10 hours. We took a snapshot of memory on time A (we define it as snapshot A), when almost every pages became dirty (non-zero), and took 8 additional snapshot hourly after that time point. Then we performed XBZRLE decompression between the snapshot A and the remaining 8 snapshots to observe how much data can be saved by the memory snapshot A. Even the VM ran for a long time, the transferring memory size can be reduced over 1GB memory by maintaining a memory snapshot (Fig. 1).

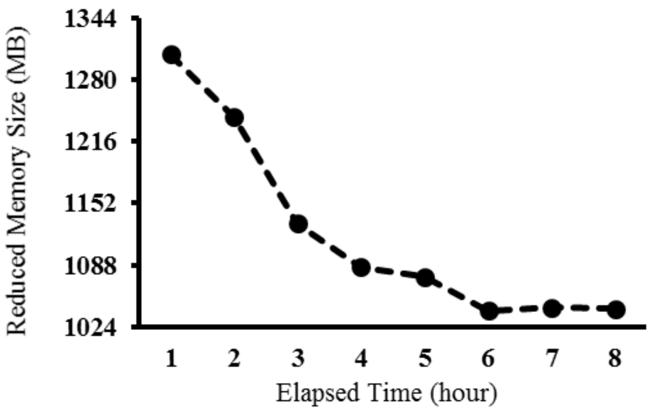

Figure 1. The Saved Memory Size after Performing XBZRLE Compression between a Memory Snapshot and Future Memory Snapshots.

3. Efficient Memory Live Migration Technique

The description of each component in our live migration architecture is as follows:

Disk Cache: In order to rebuild the memory by page-level data deduplication with disk, we should maintain an identical set of disk blocks on both sides. However, maintaining a large scale disk is a significant overhead. The data blocks stored in the disk cache mainly consist of frequently used shared libraries or binary files.

Cache Tree: SHA-1 hash value store of disk cache. We can retrieve the offset of a data block in disk cache with the hash value of memory block from the cache tree.

Memory Snapshot: Memory snapshot of the VM, when the live migration just ended up at last time. It can be used to perform page-level data deduplication and XBZRLE delta encoding with current transferring memory page.

Snapshot Tree: SHA-1 hash value store of memory snapshot. It is similar with the cache tree.

Type Header: One-byte-header which identifies the type of transferring data.

Since the cache and snapshot tree can be generated by a background process, the overhead of building these two trees should be omitted from the migration overhead.

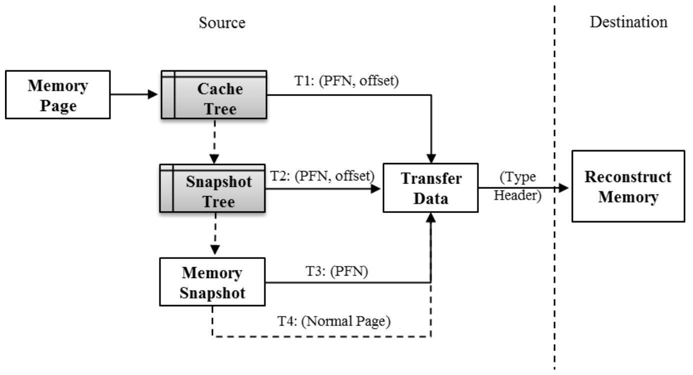

Figure 2. The Architecture of Proposed Live Migration Method. (Arrow: the hash tree lookup or XBZRLE encoding was succeed, Virtual Arrow: the hash tree lookup or XBZRLE encoding was failed)

When a VM migrates to another host for the first time, only a disk cache can be utilized to reduce the migration time. In this case, we calculate SHA-1 hash value of each transferring memory page, and check whether the hash value exists in the cache hash tree. If the hash value exists in the cache hash tree we only transfer the PFN and offset of the duplicated block inside the disk cache (T1 in Fig. 2). On destination side, the memory page with the same PFN will be filled up with the data blocks from the disk cache based on the offset. Otherwise, if there is no matched hash value in the cache hash-tree, the memory page will be sent as a normal page (T4 in Fig. 2).

After the first migration ended up, an identical memory snapshot could be generated on both sides. From that point the return back migration can be additionally accelerated by utilizing the memory snapshot. In this case, if there is no matched hash value of memory block after looking up the cache hash tree, we traverse the snapshot tree to detect page-level duplicated data with memory snapshot (T2 in Fig. 2). Similar with the disk cache case, the PFN and offset of the data blocks in snapshot is enough for reconstructing the memory on destination side. Even there is no duplicated page-level data blocks in memory snapshot, the XBZRLE delta encoding with the memory page with the same PFN in memory snapshot, would figure out sub-page level duplicated memory contents (T3 in Fig. 2). Finally, if all of the above attempts were failed, we transfer the page with no modification (T4 in Fig. 2).

4. Evaluation

4.1 Environment

The source and destination host machine has 12 Intel Xeon E5-2530 CPUs, 4TB disk and 48GB of RAM, 100Mbps NIC, ran QEMU-KVM 1.2.0 on Ubuntu desktop 12.04 64Bit. The target-migration guest was assigned to a vCPU, 2GB memory and 20GB virtual storage which ran Ubuntu 12.04 64Bit. We executed multiple applications for 30 minutes, took a snapshot, and copied the snapshot to the destination side. We treated this snapshot as a previous memory snapshot of target migration VM. After that, we ran other desktop applications for another 30 minutes and took a snapshot on qcow2 disk, to recover the same migration environment. We compared the migration performance between original KVM and the modified version of KVM (implemented the proposed migration method).

4.2 Experimental Result

Our proposed migration method outperforms original KVM over three times (Table 1). Both of data deduplication and delta encoding methods performed well to reduce the transferring memory size (Table 2).

Table 1. Live Migration Time and Transferred Data Size of Original KVM and Proposed Method

	Original KVM	Proposed
Migration Time	166.4s	50.6s
Transferred RAM	1691MB	402MB

Table 2. The Detail of Saved Transferring Data Size in Proposed Migration Method

Dedup with Disk Cache	Dedup with Snapshot	XBZRLE with Snapshot
287MB	816MB	135MB

However, the data transferring rate (Transferred RAM/Migration Time) of proposed method was much less than the original KVM. The combination effect of calculation time and disk access time was the main reason for this phenomenon. The calculation time includes SHA-1 hash computation for each memory page, hash-tree lookup time and XBZRLE encoding and decoding time. The disk access time includes snapshots and disk cache reading time both in source and destination side (Table 3). The sum of source and destination side overhead represent the total overhead caused by our migration method. To outperform the original KVM migration, at least 180MB of data should be saved by our proposed migration method. In addition, on the destination side, page-level duplicated memory pages are rebuilt directly from the disk, so the disk access time on destination side is much larger than the source side.

Table 3. The Computation and Disk Access Overhead of Proposed Migration Method

Source Side Overhead (s)

XBZRLE Encoding	Hash Computing	Hash Tree Lookup	Disk Access
2.86	7.37	0.53	1.81

Destination Side Overhead(s)

XBZRLE Decoding	Disk Access
0.87	5.29

5. Conclusion

The virtual machine live migration time is very important in terms of improving the user's experience and saving the computing resources. To reduce the VM live migration time, we analyzed the characteristics of memory content, and proposed an efficient live migration method, by utilizing previous memory snapshot and a disk cache consists of frequently used binaries and shared libraries. The experimental result proved that with page-level data deduplication and XBZRLE delta encoding, our idea outperformed the original KVM migration method significantly.

6. Acknowledgement

This research was supported by the IT R&D program of MSIP/KEIT, [K1400011001, Human Friendly Devices (Skin Patch, Multi-modal Surface) and Device Social Framework Technology], and Brain Korea 21 PLUS project for POSTECH Computer Science & Engineering Institute.

References

[1]. T Das, P Padala, VN Padmanabhan, R Ramjee, KG. Shin. "LiteGreen: Saving Energy in Networked Desktops Using Virtualization." In USENIX ATC, 2010.

[2]. J Reich, M Goraczko, A Kansal, J Padhye. "Sleepless in Seattle No Longer." In USENIX ATC, 2010.

[3]. A. Kochut and H. Shaikh. "Desktop to cloud transformation planning. " In IEEE IPDPS, May 2009.

[4]. Eunbyung Park, Bernhard Egger, and Jaejin Lee. Fast and space efficient virtual machine checkpointing, VEE, 2011

[5]. Optimizing Live Migration for Virtual Desktop Clouds, Changyeon Jo,Bernhard Egger VEE, 2013

[6]. Hudzia, B., Svard, P., Tordsson, J., Elmroth, E.: Evaluation of Delta Compression Techniques for Efficient Live Migration of Large Virtual Machines, VEE, 2011