ME2: Efficient Live Migration of Virtual Machine With Memory Exploration and Encoding

Yanqing Ma, Hongbo Wang, Jiankang Dong, Yangyang li, Shiduan Cheng
State Key Laboratory of Networking and Switching Technology
Beijing University of Posts and Telecommunications, Beijing, 100876, China
e-mail: myqbupt@gmail.com, { hbwang, dongjk, yyli, chsd } @bupt.edu.cn

***Abstract*—Live migration of virtual machineplays an important role in datacenter, which can successfully migrate virtual machine from one physical machine to another with only slight influence on upper workload. It can be used to facilitate hardware maintenance, load balancing, fault-tolerance and power-saving, especially in cloud computing datacenters. Although the pre-copy is the prevailing approach, it cannot distinguish which memory page is used, resulting in transferring large amounts of useless memory pages. This paper presents a novel approach Memory Exploration and Encoding (ME2), which first identifies useful pages and then utilizes Run Length Encode algorithm to quickly encode memory, to efficiently decrease the total transferred data, total migration time and downtime. Experiments demonstrate that ME2 can significantly decrease 50.5% of total transferred data, 48.2% of total time and 47.6% of downtime on average compared with Xen's pre-copy algorithm.**

***Keywords*— Data Center, Virtualization, Virtual Machine Migration, XEN.**

I. INTRODUCTION

Virtual machine(VM) not only works as efficient and secure resource container but also can migrate smoothly from one physical machine(PM) to another. Live migration of VM has been a powerful technique in VM-based datacenter. It has been wildly used in the following scenarios: hardware maintenance, load balancing, fault - tolerance and power-saving.

Three targets should be to achieve for live migration of VM: short disruption time, as less transferred data and migration time as possible. The disruption time refers to the period from the moment source VM shutdown to that the destination one begins to run normally, which is nearly dozens to hundreds of milliseconds. However, the migration time means the lasting time of the whole migration process starting from transferring the first byte, also known as occupying CPU time. And the transferred data consists of memory data, CPU register state, and network connection state etc. The memory data take up more than 99% of all the transferred data, in which more than half is duplicative or useless and seriously wastes network bandwidth.

In the state of the art, pre-copy is the predominant approach in live migration of machine [1-8]. In the first round, it transfers the whole memory i.e. checkpoint of the source VM. In the following rounds, it only transfers pages dirtied last time. When the dirtied pages lower to a threshold (e.g. 256KB), or the iteration times come up to a threshold (e.g. 30 times), or the pages dirty faster than the network bandwidth, it will transfer the remaining dirty pages, CPU register state and network connection state. And the destination VM restores the remaining data quickly and takes over all the upper services.

Through our experiments using Xen[9] as virtualization platform, we find that the Xen migration system simply transfers the whole memory,and do not distinguish useful memory pages from the useless. That means it transfer the VM memory pages one by one, including the unallocated pages. Such an approach leads to wasting much of network bandwidth, taking longer migration time and taking up much more PM's CPU resource.

In this paper, we present a novel approach ME2 which first jumps into the VM to explore the virtual machine's memory and then output a bitmap indicating which page is used/allocated or not so that the migration system can transfer only one byte instead of one page (usually 4KB) for those useless/unallocated pages. As for those useful pages, we try to adopt a RLE (Run Length Encode) algorithm to compress for further decreasing data amount. Compared with other complicated compression algorithms, RLE slightly takes upon CPU time and has little influence on other brotherhood-VMs, i.e. VMs on the same PM.

With four typical datacenter workloads running on the migrating VM, our experiments show that ME2 can decrease 47.6% of disruption time, 48.2% of migration time and 50.5% of total transferred data on average. And the amount of total transferred data impacts other two indexes.

The rest of this pager is structured as follows. Section II introduces related work, and Section III presents our design of ME2. Section IV presents our implementation and evaluation. Finally, Section V concludes the paper.

II. RELATED WORK

According to the sequence of migrating memory data and restarting VM, live migration of virtual machine includes two main categories: pre-copy and post-copy. In post-copy, when migrating VM, both the source PM and destination PM have the newest memory data.

Pre-copy is the predominant technique to perform live migration of VM and also Xen's default algorithm. Therefore, even the destination node crashes in the migration process, what the source node needs to do is aborting the migration and continue to run, which is reliable all the time. However, when the dirty rate is high, pre-copy must iteratively transfer large amounts of duplicated memory pages dirtied frequently.

In order to solve such problem, Clark et al. [1] simply detect the writeable work-set pages and only transfer the pages that are not dirtied last iteration. Trace-replay [5] is also introduced, which traces and logs the dirty data at the source VM and replays it at the destination one. Since the trace amounts are much smaller than that of all dirtied pages, trace-replay can significantly improve the migration performance in the second and later iteration while it can't decrease the transferred data of checkpoint, the main portion of total transferred. However, ME2 can significantly decrease the transferred data of the first round which makes up almost 20%~30% of total.

In paper [6], T. Wood et al. utilize Content Based Redundancy (CBR) elimination technique which compares the memory with another copy of last iteration and only transfers the differences. So [6] split the generally 4KB page into smaller fixed size block for finer granularity distinguish. [6] exhibits a significant performance because it only transfers dirty bytes rather than the whole 4KB page, similar to trace-replay. But [6] would maintain another copy of memory data, while ME2 only utilize 4MB memory as buffer.

There are also other choices. [7, 8] utilize compression algorithm or hash based fingerprints technique in live migration of VM and can distinguish zero-page or similarity-page. However, the migration performance would shrink when the VM works for a long time and the memory is allocated and reclaimed repeatedly. So the amount of zero-page will become fewer and fewer. For those reclaimed memory pages filled with non-zero data, [7, 8] can not recognize, But ME2 could walk through the unallocated page list so as to correctly recognize.

Rather than conflicting with above-mentioned algorithms, our research can combine with any of the methods above and complement with each other, e.g ME2 can work with trace-replay[5] in the first round.

III. Design and Implementation

In this section, we first introduce the design of our migration system including memory exploration and memory Run Length Encode (RLE) algorithm. Then we demonstrate the implementation with Xen-3.4.3 as virtual engine and Linux as Guest OS.

A. Memory Exploration

From our previous work with Xen-3.4.3, we found that Xen simply migrates the whole memory of source VM to destination node without any work of distinguishing which page is allocated or not. The total migration time is in direct proportion to the memory size of VM without any workload. Suppose there is a two- Giga-Bytes virtual machine with half of memory used and another half free. When migrating such a VM, it is certain that one-Giga-bytes memory is unnecessary to transfer, which would lead to a huge waste of network bandwidth.

But in the state of the art, Xen can't provide the memory utilization of Guest OS. Hence, we utilize a white box approach to probe and design an agent module to perform such an exploration which depends on the memory allocation mechanism. The migration daemon will decide which page to send according to the bitmap in the first iteration.

```
Open log dirty mechanism;
while(true){
   bitmap = get bitmap form VM;
   for each page{
         //page is unallocated
      if(!bitmap[page])
         send 1byte NODATA;
      else{
         size =
try_RLE([in]page,[out]compressed_page);
         if(size < page_size){
            send 1byte COMPRESSED;
            send compressed_page;
         }
      else{
         send 1byte NOCHANGE;
         send the original page;
      }
    }
  }
  If(last iteration) break;
}
Transfer CPU context and network connection state;
Suspend VM;
Switch to destination VM:
```

Table 1. Pseudo Code of ME2

B. Memory Encode

In our ME2 migration system, we divide memory pages into two groups: unallocated and allocated. Memory Exploration can accomplish such work perfectly so as to prevent transferring unallocated memory pages significantly, which is the major contribution of decreasing traffic amounts. In our previous work, we found that VM memory has a characteristic that there are lots of similar pages. For further optimization and with little influence on host machine, we adopt RLE algorithm to compress those allocated pages.

RLE is an algorithm that only runs the whole memory once and utilizes only one copy to present consecutive same data. It consumes little CPU time to encode and decode resulting in slight influence on host machine and other brotherhood-VM. Comparing with ignorable cost, RLE can operate 20%-30% performance optimization in the following experiments.

C. Implementation

As a concrete implementation, we adopt Linux as Guest OS with kernel-2.6.18.8. Therefore, in the Memory Explorer module, we can walk through the whole VM memory by Linux memory management mechanism Buddy System which maintains each unallocated page's meta-data. Then we can translate the meta-data's address to the real page's address. After that, we can get the page number in the whole VM memory and set corresponding bit of the bitmap.

As Table 1 demonstrates, each time before pre-copy's iteration, the Migration Daemon would request the Memory Explorer for bitmap. According to the bitmap, the Migration

Daemon only sends one byte NODATA for those unallocated pages to the destination daemon. And for those allocated pages the Migration daemon tries to encode the page using RLE algorithm. If such encoding is cost-efficient, i.e. the compressed data is smaller than page size (generally 4KB), the migration daemon sends one byte COMPRESSED so that the destination migration daemon can decode the compressed data. Otherwise, it sends one byte NOCHANGE to the destination. However, the compressed data is buffered and accumulates up to 4MB to send. The pseudo code is as Table 1 illustrates.

However, one point is very important that we must open the mechanism of logging dirty pages first before exploring the guest OS's memory so that the dirtied pages caused by exploration module can be marked and correctly transferred in the next iteration.

Also, we execute memory exploration in each iteration mainly because the applications running on guest OS apply for and release memory dynamically. As a result, the pages dirtied last iteration could be reclaimed by the operating system and such pages are unnecessary to transfer, though they are dirtied. So memory exploration can prevent such useless work.

IV. Evaluation

In this section, we demonstrate our ME2 migration system with some various characteristic workloads on server, and evaluate the performance and improvement of our ME2 compared with Xen's default pre-copy algorithm.

A. *Overview of Environment and Workloads*

In this section we describe our ME2 system implementation in detail. Xen (version 3.4.3) runs on two machines, one with an Intel Core 2.93GHz dual processor and 4G RAM, and another with Intel Core i5 2.80GHz quad processor and 8G RAM. The two machines are connected with 100Mbps bandwidth and use NFS (Network File System) as share storage of VM disk. And we use another machine act as the NFS storage server. And the VM's memory size is 1G.

None: in this scenario, we just migrate VM without any workload running on it so as to compare with other workloads.

Static Web: We use another server running siege to emulate 100 clients concurrent connected to the migrated VM which runs apache version 2.2.3. And siege can generate random Delay to simulate human activity when requesting for static web files.

Dynamic Web: In order to emulate dynamic web service, we run an open-source software PhpBB3 as forum and use MySQL as database. With 50 concurrent connections, the database data will be read frequently and will cause large quantities of dirty pages.

Stream Video: We emulate 10 users simultaneously requesting for video contents and the video data is transferred with http. And such workload takes upon much memory for buffer.

Kernel Compile: Migrating a VM while it is compiling kernel source code. This is supposed to be with very fast dirty rate. However, the dirty rate is not as fast as we thought because of the time slice Round-Robin.

B. *Performance Evaluation of ME2*

Figure 1 shows the total transferred data of ME2 with different workloads. Compared with Xen's pre-copy algorithm, ME2 can decrease nearly 50.5% of the transferred data on average, in which Memory Exploration contributes 30% while RLE 20%. Figure 2 illustrates the total migration time of ME2 and Xen. In spite of the low bandwidth, ME2 still has an average 48.2% improvement. Figure 3 demonstrates the average downtime improvement is so close to that of total transferred and total time, nearly 47.6%.

From Figure 1 and 2, it is found that the total transferred data of kernel compile is less than that without workload while the total migration time is converse. Because the total transferred data and and different workloads which cause different dirty rates.

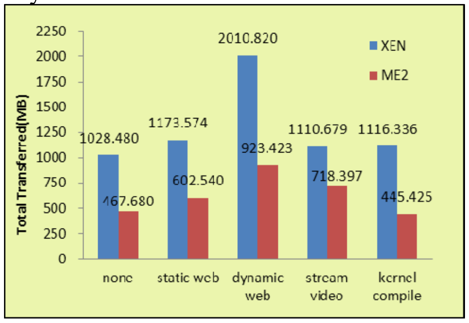

Figure 1. The transferred data of ME2 compared with Xen running various workloads (VM size: 1GB)

Therefore, the reason for such situation is that the free memory size of kernel compile is larger than that without any workload. And why migration of kernel compile lasts longer than the one with none workload? This is because the memory pages dirtied this time will be postponed to transfer next time. Hence, kernel compile will need more iterations than none workload, 13 and 5 iterations respectively in our experiments.

From our experiments, we see ME2 can improve all the three targets nearly 50%. Figure 4 gives the answer. Taking dynamic web workload as an example, ME2 can decrease more than half of network traffics in each iteration.

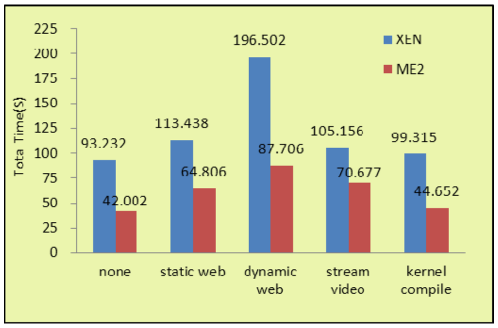

Figure 2. The transferred time of ME2 compared with Xen running various workloads (VM size: 1GB)

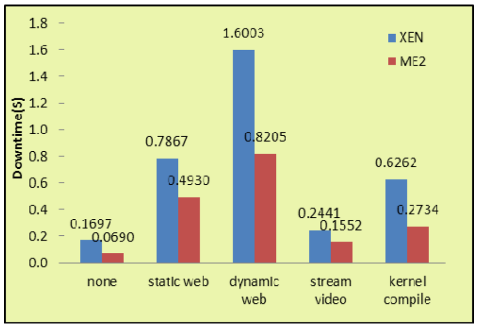

Figure 3. The downtime of ME2 compared with Xen running various workloads (VM size: 1GB)

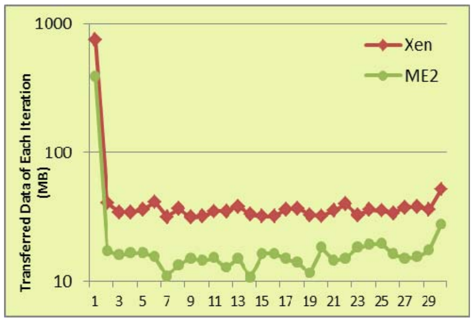

Figure 4. The transferred data of each iteration compared with Xen and ME2 running dynamic web (VM size: 1GB)

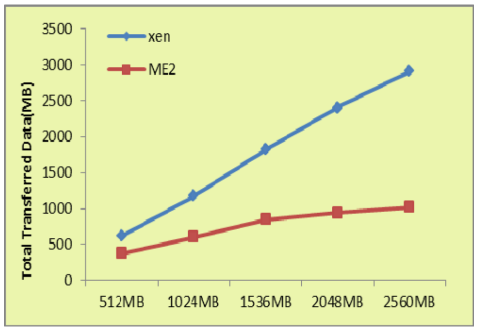

Figure 5. The transferred data of different VM size of Xen and ME2 with workload static web

In order to evaluate the migration efficiency of ME2 with different VM size, we design another group of vertical experiments. As Figure 5 illustrates, it is found that in the same situation, i.e. with same running time, workload and equal number of connections, the total transferred data of ME2 tends to be constant which would have much more obvious advantages than Xen's pre-copy.

Suppose migrating a 1GB-SIZE VM, the time overhead consists of the time of scanning buddy system, transferring bitmap and RLE for memory copy. For the space overhead, we only apply for 4MB as send-buffer in order to transfer encoded memory in batches. And the bitmap size is only 32KB which is so small that it can be ignored. Also the time of scanning and transferring bitmap is in nanosecond level, so it has slight influence on downtime which is in millisecond level. As to the RLE algorithm, it only scan memory once and the memory transfer rate is much faster than that of network bandwidth, it takes up little time, about 3.4% on average for various workloads.

V. Conclusions

In our ME2 system, the virtual machine's useless memory can be differentiated. As a result it can significantly decrease total transferred memory. However, for further reducing transferred data, RLE algorithm is utilized which can quickly compress data and occupy little CPU time compared with other complicated compression algorithms. And also ME2 can combine with other migration methods, e.g. trace and replay [6]. However, ME2 needs to write Exploration Module for different Guest OS. In spite of this, there are only a few prevailing server OS, such as Windows, Unix, Linux and Netware. Therefore, it deserves us to develop such module to improve 50% of the migration traffic, total time and server's SLA.

Acknowledgment

This work is supported by the National Natural Science Foundation of China(No. 61002011); the Open Fund of the State Key Laboratory of Software Development Environment(No. SKLSDE-2009KF-2-08); the 973 Program of China(No. 2009CB320505);the 863 Program of China(No.s 2011AA01A102).

References

[1] C. Clark, K. Fraser, S. Hand, J. G. Hansen, E. Jul, C. Limpach, I. Pratt, and A. Warfield. "Live migration of virtual machines". In Proc. NSDI '05.

[2] M. Nelson, B. Lim, and G. Hutchins. "Fast Transparent Migration for Virtual Machines". In Proc. USENIX ,2005.

[3] R. Bradford, E. Kotsovinos, A. Feldmann, and H. Schioberg, "Live wide-area migration of virtual machines including local persistent state," in Proceedings of the 3rd International Conference on Virtual Execution Environment ,2007.

[4] 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 ,2007.

[5] H. Liu, H. Jin, X. Liao, L. Hu, and C. Yu, "Live migration of virtual machine based on full system trace and replay," in Proceedings of the 18th International Symposium on High Performance Distributed Computing ,2009.

[6] Timothy Wood, Prashant Shenoy, K.K. Ramakrishnan , Jacobus Van der Merwe, CloudNet: Dynamic Pooling of Cloud Resources by Live WAN Migration of Virtual Machines, 2011.

[7] Hai Jin, Li Deng, Song Wu, Xuanhua Shi, Xiaodong Pan. "Live Virtual Machine Migration with Adaptive Memory Compression", in Cluster Computing and Workshops, CLUSTER 2009.

[8] Xiang Zhang, Zhigang Huo , Jie Ma, Dan Meng, "Exploiting Data Deduplication to Accelerate Live Virtual Machine Migration", CLUSTER 2010.

[9] 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. SOSP'03.