An Efficient Method by Using Prediction to Reduce Times in Live Migration of Virtual Machine

Boseob Kim¹, and Sungchun Kim¹
¹Computer Science and Engineering Dept, Sogang Univ, Seoul, South Korea

Abstract - Cloud system, especially IaaS has became hot issue because of its low TCO. In the field of IaaS, server virtualization has close connection due to benefits of resource sharing, fault tolerance, portability and cost efficiency. Live VM migration is one of technologies which manage virtualized server. Although live VM migration just moves virtual machine to another within few downtime, it can cause more network traffic because of longer migration time. So we suggest a method to reduce migration time through predicting iterations of pre-copy phase. Our method determines number of iterations using average dirty page rate and decides stop condition. We apply this method to Xen and demonstrate our method works efficiently.

Keywords: Virtual machine, VM migration, Cloud systems, fault-tolerance system

1 Introduction

Nowadays, cloud system now becomes popular in field of computer science and engineering. Especially, IaaS (Infrastructure-as-a-Service) cloud computing system allows user to take resources over web platform. Subscribers can take and launch virtual machines without lacking local devices and they can lease virtual machines cheaply. In a case of data centers, these have close connections with IaaS cloud technologies because cloud system has strength of reducing TCO (Total Cost of Ownership) and it has benefits of resource sharing, fault tolerance, portability and cost efficiency. In IaaS cloud, it is important to manage virtual machine, so VMM (Virtual Machine Monitor) must provide manage function like creating, copying, and deleting virtual machine efficiently to owner. Many of VMMs like Xen[2], or VMWare[3] provide these tools. VM migration is one of managing tools that provides copying or moving virtual machine. In particular, Live VM migration [1] is a reasonable method for migrating virtual machine. The main advantage of live VM migration is possibility of low downtime. While a running source virtual machine executes its processes and sends responses to user’s request, a state of destination virtual machine becomes almost ready-to-use. Because of this advantage, Live VM migration technology is used in VMMs and supported by the form of platforms. XenMotion[10] and VMotion[11] are example of platforms.

Nevertheless when source virtual machine continues to reply on user’s request, network traffic continues to takes some portion of network resource because source VMM sends packet which includes dirtied pages and destination host receives those in live VM migration. If network traffic of bringing pages takes up network resource continuously, it would bring lack of response and that would cause complain of users. One way to cut down the entire traffic is guaranteeing low migration time of live VM migration. If we save the migration time, stability of network resources would be improved. Many papers try to reduce the time of migration [5,6,7,8,12] or analyze cost of it [9]. Our goal is to guarantee low migration time.

In this paper we examine stages of VM migration and key factor of reducing migration time. After that, our method would be suggested, it is applied to the real-world case (Xen) and Experimental evaluation is presented at next chapter and finally we finish the paper with conclusion and future works

2 Related factor of Live VM migration

2.1 Stages of live VM migration

Live VM migration method tries to lessen suspending time of source virtual machine by pushing and pre-copying dirty pages. The core idea of live VM migration is convergence of pre-copy. While source virtual machine is running, VMM sends dirty pages (in first round, entire memory should be transferred) to destination VMM and constructs new virtual machine. After multiple rounds of constructing there must be few dirty pages enough to halt virtual machine and bring (maybe small number of) remaining pages. Then, start copied virtual machine on destination VMM and reattach resources to it. That kind of design is guaranteed to reduce downtime.

In more details, live VM migration consists of 6 phases [1]; those are as follow.

1) Initialization: source virtual machine is selected for migration. Also destination host is initialized for migration.

2) Reservation: resources at the destination host are reserved for destination virtual machine
3) **Iterative pre-copy**: pages in source virtual machine are modified during the previous iteration are transferred to the destination. The entire RAM is sent in the first iteration.

4) **Stop-and-copy**: the source virtual machine would be suspended for last transfer.

5) **Commitment**: the destination host indicates that it has received successfully a consistent copy of virtual machine.

6) **Activation**: resources are re-attached to the VM on the destination host.

Each stage takes one step except stage 3, iterative pre-copy stage. It would continue to execute unless there is stop condition that is satisfied.

### 2.2 Key factors of migration time

Iterative pre-copy phase effects execution of stop-and-copy phase and time for copying pages is almost every portion of migration time, although other stages do not require much data transfer. So to achieve lower migration time, we should reduce the time of iterative pre-copy phase. So it is important to decide stop condition. Stop condition is a condition that stops iterations. In normal case, stop condition of pre-copying is ‘the number of dirty pages is low enough to stop copying’. But a server environment which raises many I/O bounded events cannot stop pre-copying unless there is other stop condition in live VM migration. Because of that case, VMM is designed to limit the number of iterations or transferred capacity of memory.

The limitation could be static value. Static limitation would ensure stable migration time. At least, we can obtain maximum migration time without much measuring of scenarios. But, static limitation would have constraints of efficient migration time. So if we decide number of iterations dynamically or new stop condition ensures less number of iterations than limitation, our method can reduce the time.

### 3 Proposal schemes for stop condition

We propose new method which decides stop condition. Using average and changes of dirty page rate, our method decides whether to proceed next pre-copy phase. To give explanation of our model easily, we will describe how the pre-copy phase reduces downtime.

When source virtual machine reaches first pre-copy phase, entire memory is ready to copy and contents of memory (pages) are sent to destination host. During the first iteration, pages are dirtied and that might cause second iteration. Suppose dirty page rate is stable. The number of dirtied pages would be lower than that of whole entries of page. At second iteration, because transfer time of dirty page is lower than that of entire memory, transfer time is lower than earlier. Third iteration is done faster than second iteration because dirtied pages of second iteration is lower than those of first iteration. Likewise, fourth iteration is done faster than third iteration. As these iterations continue, the number of dirty pages goes to zero-value gradually. So these iterations have guaranteed satisfaction of simple stop condition, ‘The number of dirty pages is little enough to go to stop-and-copy phase’. Number of shrinked pages is lower than number of whole pages. So downtime is reduced. On the others, if we know how many iterations are required we can make a flexible limitation. If source virtual machine requires too many rotations we can stop pre-copy phase and go to stop-and-copy phase. It can reduce migration time with small penalty of downtime.

If dirty page rate has stability for that case, we can predict limitation that satisfies stop condition. Suppose source virtual machine has fixed dirty page rate. Then required number of pages to send in (n+1)-th iteration is below.

\[ D_{n+1} = R \times C \times \frac{D_n}{S} \]  

\[ D_n = D_1 \times \left( \frac{R \times C}{S} \right)^{n-1} \]

\[ T \geq D_1 \times \left( \frac{R \times C}{S} \right)^{N-1} \]

\[ \frac{T}{D_1} \geq \left( \frac{R \times C}{S} \right)^{N-1} \]

\[ \log\left( \frac{T}{D_1} \right) \geq (N-1) \times \log\left( \frac{R \times C}{S} \right) \]
\[
\frac{1}{N-1} \leq \frac{\log \left( \frac{R \times C}{S} \right)}{\log \left( \frac{T}{D_1} \right)} \\
N - 1 \geq \log \left( \frac{R \times C}{S} \right) \left( \frac{T}{D_1} \right) \\
N \geq \log \left( \frac{R \times C}{S} \right) \left( \frac{T}{D_1} \right) + 1 \\
N' = \left[ \log \left( \frac{R \times C}{S} \right) \left( \frac{T}{D_1} \right) + 1 \right]
\]

T is a threshold that decides ‘little number of pages’. In expression (5), N is minimum iterations to go to stop-and-copy phase. To hold expression (6) from expression (5), it must be satisfied that \( \log \left( \frac{T}{D_1} \right) \) is lower than 0. If T is greater than \( D_1 \), live VM migration acts like non-live because it indicates that our condition is always true and source VMM decides there is no needs for doing pre-copy iteration. So in live VM migration expression (6) holds for that assumption. Because the number of iterations is a natural value, it has to be converted. So, \( N' \) in expression (9) is actual number of iterations.

But practically, dirty page rate would not be fixed due to characteristics of virtualized server. Sometimes the server is in idle state or be in working state. Idle state would not make many dirty pages but working state would make many dirty pages. The problem is we cannot predict when the I/O event occurs and is handled. So extracting stable value of dirty page rate is too hard. Therefore, for predicting number of iterations efficiently we have to guess rate from measured dirty pages for each of iterations. In this paper, we predict dirty page rate by average value of dirty pages and pre-copying times collected on previous iterations. After first iteration is done, we obtain number of dirty pages that is dirtied and pre-copying time. Then we can obtain average dirty page rate from that parameters. Number of dirtied pages and second pre-copying time are measured after second iteration. Then parameters of obtaining average dirty page rate would increase and we can get expected value of dirty rate. In one sentence we predict expected dirty page rate of next iteration as following expression.

\[
R_p = \frac{\sum_{k=2}^{n+1} DM_k}{\sum_{k=1}^{n} TM_k}
\]  

DM\(_k\) is the measured number of dirty pages and TM\(_k\) is measured pre-copying time from k-th iteration. In summary, terms of expression are described in Table I.

<table>
<thead>
<tr>
<th>Table I: Terms of Expression</th>
</tr>
</thead>
<tbody>
<tr>
<td>(D_n)</td>
</tr>
<tr>
<td>(D_1)</td>
</tr>
<tr>
<td>R</td>
</tr>
<tr>
<td>C</td>
</tr>
<tr>
<td>S</td>
</tr>
<tr>
<td>T</td>
</tr>
<tr>
<td>N</td>
</tr>
<tr>
<td>DM(_k)</td>
</tr>
<tr>
<td>TM(_k)</td>
</tr>
</tbody>
</table>

With expected dirty page rate \( R_p \), we can get expected \( N' \) (we call that \( N_E \)). If \( N_E \) is equal or lower than number of iterations that is already done we have to stop pre-copy phase because it must satisfy threshold of dirty pages. And when \( N_E \) is greater than previously obtained \( N_E \) (from first iteration to (n-1)-th iteration) we could expect there would be I/O events. Then we should stop pre-copying phase and start stop-and-copy phase because I/O events would write to pages and it may need many iterations of pre-copying.

On the other hand, \( N_E \) can be negative value because of many I/O events. In live VM migration, \( T/D_1 \) is lower than one. But when too many I/O events rise, base of logarithm is greater than one because transfer speed would be lower than dirty page rate in unit time. So, whole value of \( N_E \) cannot be positive value. We should consider that situation. When there are too many I/O events so that exceeds more than transfer speed, we would rather to stop pre-copying phase and go to stop-and-copy phase.

Considering described situations, our algorithm is shown in algorithm I. first we start it with first iteration of pre-copying, which transfers whole pages. Transfer time and the number of dirtied pages in first iteration would be measured and using that value, we obtain average transfer speed, \( R_p \) and \( N_E \). if \( N_E \) is equals or lower than 0 or greater than limitation L (At first it is pre-defined value), the source VMM decide to stop pre-copy phase. If value of \( N_E \) is between 0 and L, L is updated to \( N_E \). When number of iterations is greater than L, there would be small number of pages. Then it is also the stop condition. Among all conditions, algorithm I is described below.
Algorithm I: Algorithm which controls iteration

T=Threshold of going next phase; // threshold must be defined
C=size of page;
D1=(memory size)/C;
TP=log(T/D1);
L=Limitation of iteration; // limitation must be defined
PRC=1; // PRC is number of iterations that is occurred.
transfer whole memory;
S=(memory size)/(transfer time in first iteration);
P[1]=(number of dirtied pages in first iteration);
T[1]=(transfer time in first iteration);
while(PRC < N)
{
   RR=0;
   TT=0;
   for(i=1;i<=PRC;i++)
   {
      RR+=P[i];
      TT+=T[i];
   }
   RR/=TT;
   NE=TP/log((RR*C)/S)+1;
   if(NE > L || NE<=0)
      break;
   else
   {
      L=NE;
      PRC++;
      transfer dirtied pages; // do next iteration
      P[PRC]=(number of dirtied pages for PRC-th iteration);
      T[PRC]=(transfer time of PRC-th iteration);
      S=(P[PRC-1] * C)/(transfer time in PRC-th iteration);
   }
}

Threshold and maximum number of iteration must be defined beforehand. When we define high threshold of dirtied pages, iterations would be lessened.

4 Experimental evaluation

In environment of Xen, stop condition of pre-copy phase is represented below.

1) Number of dirtied pages of previous iteration is less than 50.

2) Number of iterations is greater than 30.

3) Capacity of transferred page is greater than 3 times of entire memory.

First condition is considered because transfer of less pages is satisfying low downtime of live VM migration. On the other hand, second condition and third condition are considered for environment with many I/O events. Too many I/O events cause long migration time with long downtime, so these conditions are to reduce migration time unless there is a risk of long downtime.

With our idea we don’t want to increase downtime and migration time when source virtual machine is in that state because with that state migration time is almost equal as transfer time of entire memory while it has low I/O events. At least it ensures downtime. (Also in case of periodic I/O events that causes fixed R) So we assign T=50. On the other hands we have to consider second and third condition. We have to reduce the migration time when server is in the state with many I/O events. To prove our efficiency, we assign L=30 that is equal as static limitation of Xen.

To simulate our idea, we constructed two situations of virtual machine. First virtual machine is in idle state that few pages are modified. Automatically dirty page rate of that virtual machine is low. Second virtual machine is in environment of doing kernel compile. Kernel compile causes I/O events so that dirty page rate is higher than idle state. Both machines are generated with 512MB and 1024MB memory and set with 32-bit virtual machine. Storage are constructed with NFS [4] so we don’t have to consider storage migration time. We measured migration time and downtime. And network card and router has maximum speed of 100Mbps. The version of Xen is 4.1.0. We use default value of parameters in formal Xen. We expect migration time is improved when it is in kernel compile. When migration time in idle with our method is similar to Xen.

Result of evaluation is shown below. We have compared Xen and our idea. Results of experiment when source virtual machine is in idle state is shown in Figure 1. Upper result shows memory size=512MB and lower shows case of 1024MB. There were not much significant changes of migration time and downtime. Its environment has low I/O events. So number of iterations is easily expected and that number is equal as that of Xen. Change of migration time is a little value so that is increased or decreased by changes of network traffic.
Efficiency of our idea was evaluated when there is many I/O bounded processes. In Figure 2, migration time is improved by 17 percent in 512MB virtual machine (upper result) and 14 percent in 1024MB virtual machine. Iterations are expected and reduced when I/O events are occurred. Many I/O bounded events increases dirty page rate and once it causes increased iteration, our machine stops pre-copy. After pre-copying is stopped, VMM remains many dirtied pages. Therefore we reduced number of iterations and got lower migration time. But in trade-off, increased downtime is measured.

5 Conclusions
As a result of our experiment, live migration time of virtual machine is reduced by proposed scheme so that we can expect that users could have a chance to get more network resources. But as downtime is increased, response time is also increased. So we have to find efficient value of threshold and limitation. Also we did not consider network speed or other I/O speed. To ensure efficient downtime, actual network speed would be important factor of migration because more transfer speed with stable downtime causes allowance for more dirty pages. So we will add a method for threshold. This would be reduce migration time with reasonable downtime. And we had obtained average dirty page rate of source machine. If we catch all I/O bounded event, we can predict incoming event by making a model per process. It increases accuracy of iterations. We will focus on those issues and trade-off between migration time and downtime.

Acknowledgments
This research was all-supported by Basic Science Research Program through the National Research Foundation of Korea(NRF) funded by the Ministry of Education, Science and Technology(2012R1A1A2009558), and part - supported by Technology Innovation Development Program funded by Small & Medium Business Administration(S2057016)

6 References


