Improvement of Live Migration mechanism for Virtual Machine based on Pre-copy

Fang Yiqiu[1], Chen Yuyang[1,*], Ge Junwei[1]

[1]College of Computer Science and Technology Chongqing University of Post and Telecommunication 400065, Chongqing, China

*chenyuyang8@126.com

Keywords：Virtual Machine(VM) live migration, dirty page rate, bandwidth adaptation

Abstract: Pre-copy algorithm based on probability prediction could predict the probability of memory pages becoming dirty and avoid the repeated transfer a large number of dirty pages. However, when the dirty pages increase dramatically, prediction of probability will fail. To solve this problem, an Adaptive Bandwidth Algorithm is introduced in the pre-copy algorithm based on probability prediction. When dirty pages rate suddenly increases, that algorithm could improve the transmission bandwidth to shorten transmission time. Experimental results show that the rate of dirty pages increase suddenly or is great, that strategy can improve the transmission bandwidth, shorten the total time of iterative transfer and downtime, so as to improve migration performance.

Introduction

By using virtual technology, cloud computing creates multiple VMs to load services and isolate environments to satisfy customers' demand of resources. Thus, cloud computing can effectively manage resources and services in the cloud computing platform[1]. To achieve dynamic load balancing and online system maintenance, we can use VM Live Migration[2] technology which means migrating the VM from the source host to the destination host on the premise of continuing to provide service. It's an important means of implementing green resource management[3]. So far, what is widely used in memory migration is Pre-copy mechanism which is copying the memory pages from source VM to destination VM in the form of cyclic iteration. Though Pre-copy algorithm performs well when the load is empty or in a low level, it doesn't perform well in high load.

Bandwidth Adaptive Algorithm Based On Markov Probability Prediction

The performance of the live migration of VM includes the total migration time and downtime. According to the migration process, the formula calculating migration time and downtime[4]:

$$T_{total} + T_{init} + \sum_{i=1} \Pr e - copy_i + T_{down} + T_{active} \quad (1)$$

$$T_{down} = T_{stopcopy} + T_{commit} \quad (2)$$

According formula (1) and (2), we know the T_{total} consists of the initialization time T_{init}, resource reservation time T_{res}, iterator time, downtime T_{down} and waking time T_{active}. And the downtime consists of the stopping-and-copy time $T_{stopcopy}$ and committing time T_{commit}. And the T_{init}, T_{res}, T_{commit} and T_{active} are constant. In order to shorten the migration time and downtime, we must focus on the iterator time. And the Pre-copy algorithm based on the probability prediction just did it.

Migration Strategy Based on Markov Probability Prediction

On account of disadvantages of the traditional migration strategy, literature[5] proposed Markov probability model to forecast the probability of dirty pages appearing. It adds probability forecast module and pages information monitor module to the traditional mechanism. It assumes that the total number of rounds is n, the dirty pages will be transfer is Di in every round, the bandwidth is used to transfer the dirty pages is BW_i and it costs T_i in each rounds, so we can get the formula as follow：

$$T_i + \frac{D_i}{BW_i} \quad , \quad 0 < i < n \tag{3}$$

During the last iterator, stopping the VM and enter the stopping-and-copy phase, and we can get Formula (4) from (2) as follow:

$$T_{down} = \frac{D_i}{BW_i} + T_{commit} \quad , \quad i = n \tag{4}$$

According the formula (1)(2)(3)(4), we can get:

$$T_{total} = T_{init} + T_{res} + \sum_{i=1}^{n-1} T_i + \frac{M_i}{BW_i} + T_{active} \tag{5}$$

Although the migration strategy based the probability prediction has greatly shortened the total migration time and the downtime, when the application running on the VM need to read and write a lot of memory pages suddenly, the probability prediction module will fail, which have a bad effect on migration performance.

Bandwidth adaptive algorithm based on probability prediction of Markov

During each iteration, the number of dirty pages is changing. So the bandwidth adjustment factor a is introduce[6], a $\in$[0,1], which consider the dirty pages rate that changes with time and business characteristics. We can get the current iteration of bandwidth usage in business by it. Therefore, the bandwidth usage in round k can be calculated as:

$$B\_bw = (1-a) \times avg(\sum_{i=1}^{k-1} U\_dirty_i) + a \times U\_dirty_k \tag{6}$$

U_dirty is dirty pages rate for history. Bandwidth allocation factor a reflects the bandwidth historical and current usage. When a $\in$[0,0.5), it means that the resource occupied by application business decreases, and the bandwidth average usage from round 1 to i-1 is bigger than round i (current round). So we should focus on historical usage of bandwidth. When a=0.5, it said that the current business is stable. When a $\in$ (0.5,1], it reflects that current business increases sharply. In order to prevent the occurrence of phenomenon that different businesses preempt bandwidth resource, bandwidth transferring dirty pages in current round should be consider the value of total physical bandwidth minus business bandwidth mainly. Before transferring memory pages every round, we could calculate the value of the current bandwidth used divided by total physical bandwidth as the value of the coefficient, which is appropriate.

Bandwidth adaptive algorithm

In the initialization phase, we need to obtain the bandwidth allocation rules in according to the number of rounds of iterations. When it's first round of the iteration, all the memory pages should be copied to the destination VM by the current initial network bandwidth. When it's the final round of the iteration, the VM will spend all spare bandwidth to transmit the remaining memory pages. When it 's in middle iteration phase, we need to calculate the reserved bandwidth or correct the

reserved bandwidth. Firstly calculating the bandwidth adjustment factor a , and the bandwidth reserved according to the dirty pages rate and the formula (6). Then we should to fix the reserved bandwidth. Because of the business changed on the VM causes the change of the dirty pages rate, which may result in the reserved bandwidth calculated is greater than the current spare bandwidth. When the reserved bandwidth calculated is greater than the current spare bandwidth, it's necessary to make a correction to the reserved bandwidth, namely the value of reserved bandwidth calculated divided by current bandwidth used to adjust the spare bandwidth. Finally, calculate the transmission bandwidth and start to send memory pages to the destination VM. Algorithm as follows:

Input: iterative rounds n

Output: this round allocate bandwidth E_bw

```
if (n==1)    //first round iteration
then  return E_bw //send the pages thought
        // initial bandwidth E_bw
else  //obtain the physical network
        bandwidth T_bw,
// physical bandwidth used U_bw
//calculate physical spare bandwidth
// F_bw= T_bw – U_bw
    if (n==last_iter) //final round iteration
        E_bw = F_bw
        return F_bw
    endif
//calculate bandwidth adaptive factor a
// a=U_bw/T_bw
//obtain this round dirty page rate
 //U_dirty[n]
//calculate average dirty page rate from 1
    //round
//to n-1 round
                  (∑_{i=1}^{n-1} U_dirty_i)
D_avg = ------------------------
                      (n-1)
//calculate business reserved bandwidth
B_bw=(1-a)×D_avg+a×U_dirty[n]
 if(B_bw>F_bw)
//correct the business reserved  bandwidth
    B_bw=F_bw×(B_bw/(B_bw+U_bw))
 endif
 Return  E_bw = F_bw-B_bw
endif
```

Simulation

We have experimented between two PC nodes within the LAN with the migration memory size is 256MB, 512MB, 1024MB. Virtual platform is Xen5.0. VM operation system is ubuntu11.0. There are with two kind of workload on VM. The first one is the compile-kernel to simulate the high dirty page. The second one is the intermittent compile-kernel to simulate the environment that the dirty pages rate sharp increases or reduces suddenly.

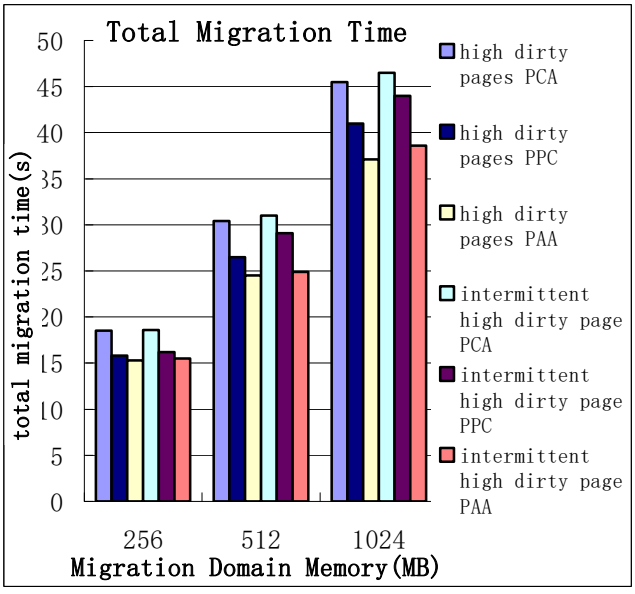

Table 1. Total migration time

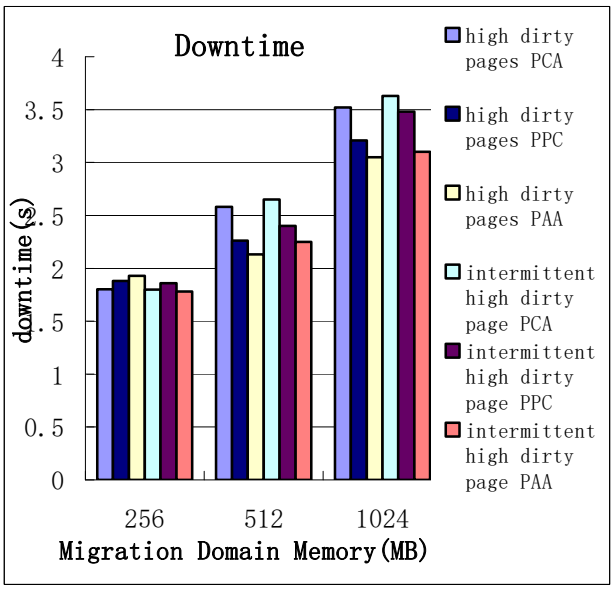

Table 2. Downtime

Take the average of the experimental results after 10 times, and shows in table 1 and table 2, which reflect the results of experiments using Xen original pre-copy algorithm (PCA), the improved probability prediction algorithm (PPA) and pre-copy algorithm based probability prediction and bandwidth adaptation (PAA).

The table 1 and Table 2 shows that in the case of dirty page rate is high, compared to PCA, the PPC costs less total migration time and downtime. Under the environment of intermittent high dirty page, the PPC shows its helplessness , while the PAA performs well. The strategy proposed by this paper can significantly reduce the total migration time and the downtime. In case of high dirty page, when migration domain memory is 1024MB, the test migration iterations are 14times. After experiments, the data are averaged, the experimental results shown in table 3.

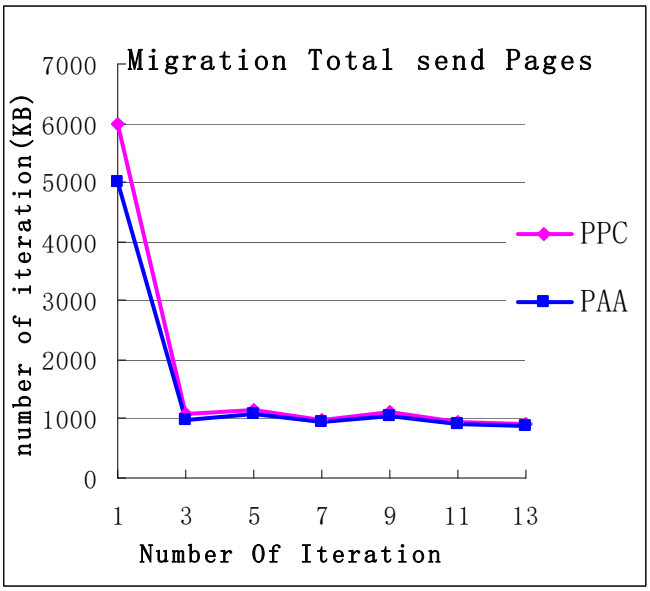

Table 3. Total Send Pages of Migration

The table 3 shows that PAA use the probability prediction, in the process of iteration for the first time, data transmission of the page to decrease is obvious, in later iterations for many times, the total number of transmission is also reduced.

Conclusions

In this paper, according to analyze PCA and PPC, we proposed a pre-copy algorithm based on the probability predicting and bandwidth adaptation, and experimented on the Xen platform. The result shows that the total migration time of VM, the downtime and the iterative rounds are reduced in varying degrees. The next step is to consider to implement the bandwidth adaptation algorithm the complex network.

References

[1] Nathuji R, Schwan K. Virtual power: Coordinated power mannagemet in virtualized enterprise system.//Proceedings of ACM Symposium on Operating Systems Principles(SOSP'07). USA, 2007:265-278.

[2] Clark C, Fraser K, Hand S, et al. Live migration of virtual machines//Proceedings of the 2nd Symposium on Networked Systems Design and Implementation NSDI' 05. Boston, USA, 2005: 273-286.

[3] Wood T P, Shenoy P A, Venkataramani A, Yousif M. Black-box and gray-box strategies for virtual machine migration//Proceeding of the 4th USENIX Symposium on Networked Systems Design and Implementation (NSDI' 07). Cambridge, USA, 2007: 229-242

[4] Sun Mingsong, Ren Wenwen. Improvement on Dynamic Migration Technology of Virtual Machine Basedon Xen[C]//Proceedings of the 8th International Forum on Strategic Technology. Washington D. C. USA: IEEE Press,2013:124-127.
[5] Sun Guofei，Gu Jianhua，Hu Jinhua. The improvement of Virtual Machine live migration mechanism based Pre-copy [J]. Journal of Computer Engineering；2011，32(10)：36-39.
[6] Liu Shihai, Sun Yuqing, Liu Guyue. Bandwidth adaptation allocation algorithm virtual machine migration of business-oriented features[J]. Chinese Journal of Computers, 2013,36(9):1816-1825.