An Enhanced Approach for Estimating Writable Working Set by Pre-copy and Prediction

Tin-Yu Wu[1], Wei-Tsong Lee[2], Jhih-Siang Huang[2], Chien-Yu Duan[2],Tain-Wen Suen[3]

[1]*Institute of Computer Science and Information Engineering, National Ilan University, Taiwan, R.O.C.*

[2]*Department of Electrical Engineering, Tamkang University, Taiwan, R.O.C.*

[3]*Chung-shan institute of Science and Technology, Taiwan, R.O.C.*

tyw@mail.tku.edu.tw, wtlee@mail.tku.edu.tw, hjs770502@hotmail.com, jason84195@hotmail.com, qoosuntw@hotmail.com

Abstract- Because cloud computing and cloud-based applications have attracted more and more attention in recent years, more and more servers are needed to provide services. However, each traditional server can execute only one application at one time. In order to satisfy user requirements and increase the server utilization rates, server virtualization has thus presented to the public. For clients to experience seamless services, live migration has become a significant research topic. Previous researches have revealed that two important parameters affecting live migration are downtime and total migration time. Traditional pre-copy approaches can reduce the downtime but the total migration time is too long due to repeated retransmissions. Therefore, to decrease the downtime and total migration time, our proposed scheme sets the threshold values based on memory modification prediction technique to determine whether the pages must be transferred in the stop-and-copy phase.

Keywords: Virtual Machine, Live Migration , Pre-copy

I. INTRODUCTION

With the increased adoption of cloud-based applications in recent years, virtualization technology for cloud computing, which is most often applied to computers with hierarchical memory, becomes more and more important. The lately development of virtualization technology allows users to establish several virtual machines (VMs) on one physical machine to simultaneously run multiple operating systems (OSs), namely OS virtualization.

Virtualization at the server is server virtualization. At present, one server executes one task only, which leads to low server utilization rates, while server virtualization enables higher utilization rates of the server by partitioning one physical server into multiple virtual servers. Although server virtualization can improve the service availability and achieve dynamic resource management, the heavy workloads or maintenance on the server might occur and terminate the service. In order to serve the customers incessantly, service providers have developed live migration technique for the customers to experience seamless services.

Currently, memory migration included three major steps:

1) Push Phase: The source VM continues to run and push modified memory pages to the destination.
2) Stop-and-copy Phase: The source VM is stopped and all the remaining memory pages are copied to the destination.
3) Pull Phase: The new VM executes. If the new VM accesses a memory page that has not been copied yet, the page fault occurs and this page is copied from the source VM.

Two main approaches at present are pre-copy, which integrates the former two phases, and post-copy, which integrates the latter two. There are two important parameters of the modified version of pre-copy:

1) Total migration time: the time from the migration start to the migration end; namely, the time from the iterative pre-copy phase to the commitment phase.
2) Downtime: the time from when the source VM stops to when the destination resumes; namely, the time from the stop-and-copy phase to the commitment phase.

When customers experience the downtime, it means that the service is terminated. In other words, when the downtime of a service tends to zero, the optimal live migration can be attained.

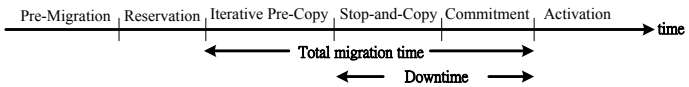

Figure 1. Total migration time and downtime of pre-copy

Though decreasing the downtime efficiently, pre-copy migration transmits all memory pages in the first round of iteration and repeats iterative transmissions, which causes unnecessary retransmissions of certain memory pages, prolongs the total migration time and reduces the performance.

To solve the above-mentioned problems, we propose a prediction model to predict the probability of memory modification and monitor the relative modification rate of each memory with the aim to reduce the memory amount in the first round and unnecessary iterative migration in the subsequent rounds.

The rest of the paper is organized as follows. Section II introduces related work and approaches. Section III elaborates our proposed scheme that improves the file size for the first round and avoids unnecessary retransmissions to decrease the total migration time and downtime. Section IV presents the simulation results and performance analysis by comparing our scheme with pre-copy[1] and Time-Series Based Pre-copy [2]. Finally, the conclusion is given in Section V.

II. BACKGROUND AND RELATED WORK

As the approach currently adopted by VMware, Xen and KVM, pre-copy migration proposed by [1] include the following six steps:

1) Pre-Migration: Make sure that the resources required to receive migration are guaranteed on the destination host.
2) Reservation: Confirm the available resources on the destination host and notify the source host to start the migration operation.
3) Iterative Pre-Copy: The source host starts this step by iteratively copying all the memory pages to the destination host. All memory pages are transferred to the destination host in the first round of iteration while pages dirtied in the previous iteration are transferred in the subsequent iterations.
4) Stop-and-Copy: The VM running on the source host is suspended and the remaining memory pages, pages with high modification rate, are transferred to the destination host.
5) Commitment: The destination host confirms that all data of the migrated VM has been received successfully and acknowledges the source host to discard the original VM.
6) Activation: VM on the destination host is now activated.

Generally, pre-copy migration judges whether the pages have been migrated by the following three kinds of bitmap:

- to_send: the dirty pages that have been modified in the former iteration and might be sent to the destination host in the current iteration.
- to_skip: the dirty pages that have been modified in the current iteration and would be skipped.
- to_fix: the pages with high modification rate and should be transferred in the stop-and-copy phase.

Because the pages modified in the current iteration must be transferred in the subsequent iterations, pre-copy migration reduces the downtime efficiently but prolongs the total migration time due to unnecessary repeated retransmissions. This paper further defines the pages with high probability of modification as Writable Working Sets (WWS) and finds that once the memory pages have been modified, it is very possible that the adjacent pages would be modified also. The administrator chooses a minimum and a maximum bandwidth limit and transfers the pages at the minimum bandwidth in the first pre-copy round in case the decrease in system performance. Supposing the dirtying rate of the pages is higher than the maximum bandwidth, the pages are of high dirtying rate and should be transferred in the stop-and-copy phase.

Based on the three kinds of bitmap described above, [2] gives the following equations to decide when not to send dirty page, p. Supposing dirty page p meets the three equations, p should not be sent to the destination host in this iteration:

$$\begin{aligned} &p \in to\_send \;\&\; p \in to\_skip \\ &p \notin to\_send \;\&\; p \in to\_skip \\ &p \notin to\_send \;\&\; p \notin to\_skip \end{aligned} \tag{1}$$

If p meets Equation (2) in the iteration,

$$p \in to\_send \;\&\; p \notin to\_skip \tag{2}$$

and

$$\sum_{i=1}^{N} (p \in to\_send\_h[i]) \geq K \tag{3}$$

Where K means the threshold of high dirty pages, p should be sent in the stop-and-copy phase.

Although this method reduces the number of iterations, the optimal value of K is 3, which means that if the memory has been successively modified for 3 times, the page is considered high dirty rate and will be transferred in the stop-and-copy phase that increases the file size in this phase and the downtime also.[3] integrates Markov prediction model with Bayes conditional probability to predict the probability of modification in the next rounds of iterations. Traditional memory pages are marked as 0 to denote "read" and "not modified," but to improve the prediction accuracy, the authors moreover set 0, 1 and 2 to denote "not modified," "read" and "written."

Because the system monitoring starts from the pre-migration phase, the modification probability of memory pages can be predicted in the first round of iteration. Through the predicted probability, a threshold value is selected to determine whether to send the page or not. In every iteration, the pages with higher dirty rate than the threshold value will not be transferred.

The above-mentioned memory prediction technique simply predicts the probability of memory modification. To accurately estimate whether to execute migrations or not and to avoid unnecessary retransmission, the probability must be adjusted according to the current memory, namely, to compare with the skip-table to decide whether to migrate or not. To reduce the transmission in the first round of iteration, the authors starts the monitoring right before the migration and determines whether to migrate or not after the beginning of the first iteration.

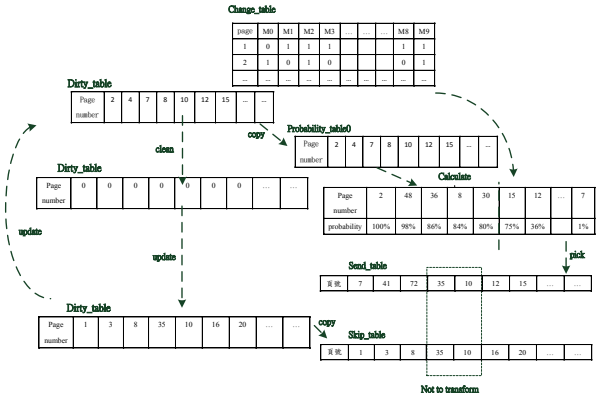

Figure 2. The flowchart of memory prediction in [3]

III. PROBLEM STATEMENT AND PROPOSED METHOD

As described in the first section, the total migration time for

pre-copy is too long because of two reasons:

1) All memory pages are transferred in the first round of iteration. Since the file size is too large, the transmission time of the first round will be prolonged.
2) Once the pages are modified in this iteration, they will be transferred in the subsequent iteration, which leads to unnecessary retransmissions.

To solve the file size problem in the first iteration mentioned in 1), we will use Markov prediction model proposed in [3] to predict the memory. Next, by integrating the predicted probability of memory presented in [3] and our defined related dirty rate, we can redefine the K value proposed in [2] to determine whether to transfer the memory to the stop-and-copy phase. The pages with higher dirty rate and the higher predicted probability of dirty rate will be chosen to be transferred in the stop-and-copy phase.

In the following, we will discuss predicted probability of memory modification and related dirty rate, respectively.

A. Predicted Probability of Memory Modification

[3] predicts the memory modification according to probability theory but its proposed prediction model is different from traditional memory operations. Although there are only two traditional memory statuses: "read" or "not modified," [3] presents three kinds of memory statuses: "not modified," "read" and "written." However, in our opinion, the read action during the memory-copying process does not influence the migration efficiency directly. Therefore, we here modify the memory prediction model.

1) Supposing the memory has been modified, its status is E_1. Supposing the memory has been read or not modified, its status is E_0. Thus, the probability of state change can be

$$P(E_i \to E_j) = P(E_i \mid E_j) = P_{ij} \tag{4}$$

2) If there are n statuses of the memory, they will be $E_1, E_2, E_3, ..., E_n$. The probability for E_i to turn into E_i can be

$$P(E_i \to E_j) = P(E_i \mid E_j) = P = \begin{bmatrix} P_{11} & P_{12} & \cdots \\ P_{21} & P_{22} & \cdots \\ \vdots & \vdots & \vdots \\ P_{n1} & P_{n2} & \cdots \end{bmatrix} \tag{5}$$

Because there are only two traditional memory statuses, n=2.

3) Suppose the initial status of the memory is k=0. After k times of status modification, the probability of E_j in the kth iteration will be

$$\sum_{j=1}^{n} \pi_j(k) = 1 \tag{6}$$

Based on the non-aftereffect property of Markov process and conditional probability formula given by Bayes' formula, we get:

$$\pi_j(k) = \sum_{i=1}^{n} \pi_j(k-1) P_{ij} \quad ,j=1,2,...,n \tag{7}$$

and the row vector

$$\pi(k) = \left[\pi_1(k), \pi_2(k), \pi_3(k), ..., \pi_n(k)\right] \tag{8}$$

Therefore,

$$\begin{aligned} &\pi(1) = \pi(0)P \\ &\pi(2) = \pi(1)P = \pi(0)P^2 \\ &\vdots \\ &\pi(k) = \pi(k-1)P = \pi(0)P^k \end{aligned} \tag{9}$$

Where $\pi(0) = \left[\pi_1(0), \pi_2(0), \cdots\right.$ is the probability vector of the initial status.

4) For example:

Table 1. The first 20 status changes of a dirty page

Serial No.	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
Status	1	0	1	0	0	1	0	1	1	0	1	0	0	1	1	0	0	1	1	0

Table 1 show that ten processes start from status 1, in which seven of them changes from status 1 to 0 and three of them changes from status 1 to 1. Thus:

Similarly, we can use this formula to calculate the probability from status 0 to 0 and from status 0 to 1. Finally, we get the status change matrix:

$$P = \begin{bmatrix} 0.667 & 0.334 \\ 0.7 & 0.3 \end{bmatrix} \tag{10}$$

After 20 times of iterations, we get the status matrix of the dirty page, P. To predict the status probability of the dirty page only, we deem that the initial status is $\pi(0) = [0 \quad 1]$. After k rounds of status changes, the status probability would be $\pi(k)$. k=1.

$$\pi(1) = \pi(0)P = [0 \quad 1]P = [0 \quad 1]\begin{bmatrix} 0.667 & 0.334 \\ 0.7 & 0.3 \end{bmatrix} = [0.7 \quad 0.3] \tag{11}$$

Therefore, after this iteration, the probability that the dirty page will be modified again is 30%.

B. Related Dirty Rate

Because the principle for memory migration is that the lower the memory modification rate is, the faster the dirty page will be migrated; the higher the memory modification rate is, the later the dirty page will be migrated, and even later than the stop-and-copy phase. Instead of predicting the memory page

modification rate only, our proposed method sets the related dirty rate to find out the pages with the highest number of modification. Thus, we can more accurately figure out the pages suitable to be transferred in this iteration and the pages to be transferred till the next iteration.

To find out the pages suitable to be transferred in each iteration, we start the monitor before the migration, just like the foregoing predicted probability of memory modification. According to the data obtained from the monitor, we get the pages with higher number of modification, the related dirty rate namely.

To record the number of modifications of each page, we further add the to_send_num to the pre-copy approach, compare the to_send_num of each page after the beginning of every iteration, and find out the maximum of the to_send_num, the max_to_send_num namely. By the formula given below, we can calculate the related dirty rate to find out the page with the highest number of modifications.

$$the\ ralated\ dirty\ rate = \frac{to\_send\_number}{\max\_to\_send\_number}$$

According to the definition, the maximum of the related dirty rate is 1 and the minimum is 0 ($\max\_to\_send\_num \neq 0$). The closer to 1 the related dirty rate is, the more page modification times there will be. On the contrary, the closer to 0 the related dirty rate is, the less page modification times there will be. With the basic understanding of the related dirty rate, we integrate the predicted probability of memory modification with the related dirty rate to define a new K.

$$K = probability \times the\ ralated\ dirty\ rate$$

Because the related dirty rate continuously records memory modifications, the record will not be updated because of the transmission in the previous iteration. In addition, the related dirty rate is based on the number of modifications and thus can be regulated dynamically. Therefore, compared with the K value presented in [2], our proposed new K by integrating the predicted probability of memory modification with the related dirty rate can be adjusted dynamically also. Since our method covers memory prediction, no matter the iterative pre-copy phase is long or short, when the pages with the value higher than K need to be transferred in the stop-and-copy phase, it means that the pages have high related dirty rate and high predicted probability of changes in the subsequent iterations. This proves that by using our defined K, we can find out the pages to be transferred in the stop-and-copy phase more accurately.

However, with a new threshold value K, it is still possible that the page transmission threshold is lower than K, which means that the page with low predicted modification rate and low related dirty rate cannot reach the destination in the iterative pre-copy phase.

First, we discuss the conditions for transferring pages. Whether to transfer the pages or not is not determined simply by the send_table based on the K value. We have to compare the send_table with the skip_table to find out whether the same pages have been transferred. The serial numbers of the pages on the skip_table reveal that when we determine whether to transfer the pages or not, the pages have been modified. In order to avoid unnecessary retransmissions, the pages on the skip_table would not be transferred in this round of iteration. Therefore, if a page appears both on the send_table and the skip_table, this page will not be transferred in this iteration.

According to above-mentioned conditions for transferring pages, it is possible that although the memory modification rate and the related dirty rate is low, the pages are modified only when we determine to transfer the pages or not. In such a manner, certain memory pages with less number of modifications and low related dirty rate still cannot reach the destination in the iterative pre-copy phase and must be transferred in the stop-and-copy phase.

To reduce the occurrence of such a situation, before the stop-and-copy phase, namely the last iteration of the iterative pre-copy phase, we use both the send_table and the skip_table to decide to transfer the pages or not. Supposing in the last iteration, the serial number of the pages in the foregoing situation does not appear in the skip_table, we will transfer the pages in the last iteration.

IV. PERFORMANCE EVALUATION

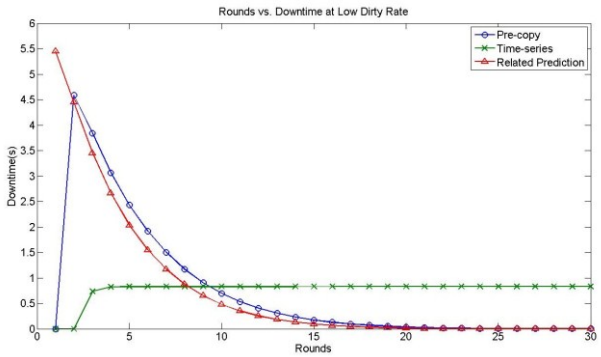

Figure 3. Downtime under low dirty rate

Figure 3 displays the downtime of different memory migration mechanisms under low dirty rate environment. In the first round of iteration, traditional pre-copy and time-series approaches operate the memory migration from the source host to the target host. Supposing the VM moves to the stop-and-copy phase right after the first round of iteration, there will be no pages for transmission in the stop-and-copy phase and thus the downtime of the pre-copy and time-series approaches is 0. According to the predicted probability of memory modification, our proposed method chooses to migrate the pages with high modification probability in the stop-and-copy phase and the downtime in the first round of iteration therefore is higher than the other schemes.

Time-series approach considers the pages modified for 3 times as high dirty rate. Because no such pages appear in the second round of iteration, the downtime of the time-series approach in the second round remains as 0.

In the following rounds, low dirty rate gradually minimizes

the number of modified pages and leads to the accurate WWS. The figure reveals that after several rounds of iteration, the downtime of the pre-copy approach and our method gradually reduces for finding out the WWS. Since the number of modified pages is minimized, the pages modified for 3 times become less and the curve of the time-series approach tends to increase slowly.

Because the pre-copy approach aims to minimize the downtime to offer clients seamless services and the time for a persistence of vision lasts for 0.1~0.4 seconds, the optimal downtime for the pre-copy approach ranges between 0.1~0.4 seconds. Figure 3 shows that the downtime of the pre-copy approach does not approach 0.4 seconds until the 12th rounds of iteration and the downtime of the time-series approach increases slowly after the 7th round of iteration, 1.15 seconds approximately. As for our method, the downtime can be reduced to around 1.15 seconds in the 7th round of iteration.

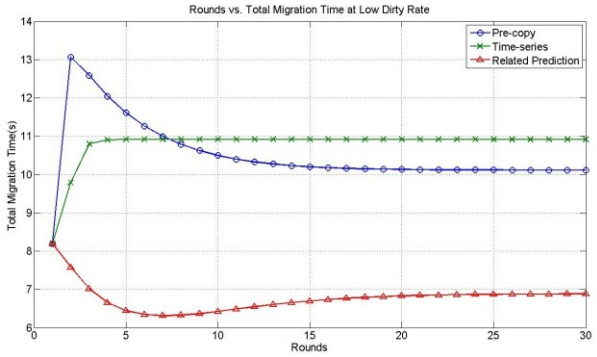

Figure 4. Total migration time under low dirty rate

Figure 4 shows the total migration time of different memory migration mechanisms under low dirty rate environment. It is revealed that in most rounds of iteration, our method outperforms the pre-copy and time-series approaches because our method can figure out the pages with higher modification probability based on the predicted probability of memory modification and migrate the pages in the following rounds of iteration.

The total migration time refers to the migration time of all rounds plus the downtime. Together with Figure 4, we find that the downtime of the pre-copy and time-series approaches is 0 in the first round of iteration, which means that the total migration time of the pre-copy and time-series approaches in the first round is used for memory migration. However, the downtime of our proposed method occupies most of the total migration time from the beginning.

The performance evaluation reveals that in the same low dirty rate environment, the pre-copy approach spends more rounds of iteration to obtain the better downtime but result to too long total migration time. Although the time-series approach can decrease the total migration time by reducing the rounds of iteration, our method can attain the same downtime but the better total migration time in the same rounds of iteration. To compare with traditional pre-copy approach, on the other hand, our proposed method can result to the same downtime in less rounds of iteration but the less total migration time. To sum up, our proposed method obviously outperforms the other two schemes.

V. CONCLUSIONS

By integrating the predicted probability of memory modification with our proposed related dirty rate, we can predict the probability of memory modification, find out the pages with the highest number of modifications, and determine the pages suitable to be transferred in the iterative pre-copy phase. In this way, we can redefine the suitable file sizes for each phase to avoid the prolongation of the total migration time and the downtime.

The simulation result displays that in the same low dirty rate environment, our method can reach the downtime and the less total migration time in less rounds of iteration while comparing with the pre-copy approach, and reach almost the same downtime in the same rounds of iteration but the less total migration time while comparing with the time-series approach. This clearly shows that our method excels the other two schemes.

Nevertheless, our method encounters the same problems like the pre-copy scheme under high dirty rate. All we can do is to decrease the repeated iterative transmissions and move into the stop-and-copy phase as early as possible to reduce unnecessary retransmissions.

References

[1] Christopher Clark, Keir Fraser,“Live migration of virtual machines”, NSDI'05 Proceedings of the 2nd conference on Symposium on Networked Systems Design & Implementation, Vol. 2, pp. 1-14. (2005)

[2] Bolin Hu , Zhou Lei , Yu Lei , Dong Xu , Jiandun Li , “A Time-Series Based Pre-copy Approach for Live Migration of Virtual Machines”, ,2011 IEEE 17th International Conference on Parallel and Distributed Systems (ICPADS),pp. 947 – 952, 7-9 Dec , (2011)

[3] SUN Guo-fei, GU Jian-hua,HU, Jin-hua, ZHAO Tian-hai ,“Improvement of Live Memory Migration Mechanism for Virtual Machine Based on Pre-copy”, Computer Engineering, Vol.37, No.13, July(2011)

[4] Michael R. Hines, Umesh Deshpande, Kartik Gopalan, “Post-copy live migration of virtual machines”, ACM SIGOPS Operating Systems Review, Vol. 43, Issue 3, July (2009)

[5] Zhaobin Liu,Wenyu Qu ,Tao Yan ,Haitao Li ,Keqiu Li,“Hierarchical Copy Algorithm for Xen Live Migration”,2010 International Conference on Cyber-Enabled Distributed Computing and Knowledge Discovery, pp.361-364, Oct (2010)

[6] Kuno, Y. , Nii, K. ,Yamaguchi, S.,“A Study on Performance of Processes in Migrating Virtual Machines”,2011 10th International Symposium on Autonomous Decentralized Systems (ISADS), pp.567–572, March (2011)

[7] Haikun Liu, Hai Jin, Xiaofei Liao, Chen Yu, Cheng-Zhong Xu, “Live Virtual Machine Migration via Asynchronous Replication and State Synchronization”, IEEE Transactions on Parallel and Distributed Systems, Vol. 22, pp.1986-1999, December (2011)

[8] Fei Ma, Feng Liu, Zhen Liu , “Live virtual machine migration based on improved pre-copy approach”, 2010 IEEE International Conference on Software Engineering and Service Sciences (ICSESS), pp.230-233,16-18 July ,(2010)

[9] Ibrahim, K.Z. , Hofmeyr, S, Iancu, C, Roman, E. ,“Optimized pre-copy live migration for memory intensive applications”, 2011 International Conference for High Performance Computing, Networking, Storage and Analysis (SC), pp. 1-11, 12-18 Nov. (2011)

[10] Wentian Cui ,Meina Song , “Live memory migration with matrix bitmap algorithm”, 2010 IEEE 2nd Symposium on Web Society (SWS),pp.277-281,16-17 Aug. (2010)