World Academy of Science, Engineering and Technology
International Journal of Computer and Information Engineering
Vol:18, No:5, 2024

# A Dirty Page Migration Method in Process of Memory Migration Based on Pre-copy Technology

Kang Zijian, Zhang Tingyu, Burra Venkata Durga Kumar

***Abstract*—**This article investigates the challenges in memory migration during the live migration of virtual machines. We found three challenges probably existing in pre-copy technology. One of the main challenges is the challenge of downtime migration. Decreasing the downtime could promise the normal work for a virtual machine. Although pre-copy technology is greatly decreasing the downtime, we still need to shut down the machine in order to finish the last round of data transfer. This paper provides an optimization scheme for the problems existing in pro-copy technology, mainly the optimization of the dirty page migration mechanism. The typical pre-copy technology copies n-1th's dirty pages in nth turn. However, our idea is to create a double iteration method to solve this problem.

***Keywords*—**Virtual machine, pre-copy technology, memory migration process, downtime, dirty pages migration method.

## I. Introduction

VIRTUAL machine migration usually refers to the migration of a virtual machine running on one host (the source host) to another host (the destination host). The main purpose of virtual machine migration is to improve system availability, flexibility and resource utilization. When the resource utilization of one physical host is high or exceeds its carrying capacity, migrating some virtual machines to other physical hosts can balance the load and prevent some hosts from being overloaded. And this effect also improves overall system performance and responsiveness in the same time. Moreover, in the event of a physical host failure, virtual machines can be quickly migrated to other healthy hosts to maintain application availability and continuity. Memory migration of a Virtual machine (VM) can be divided into (1) Push Phase, (2) Stop-and-copy Phase, and (3) Pull Phase [1]. Currently, there are many different migration mechanisms for memory migration of virtual machines, such as post-copy and pre-copy. Taking Xen live migration as an example, it uses a pre-migration mechanism to cycle memory pages to the destination host while the virtual machine is running and to record memory dirty pages at the same time. Each round of loops only needs to transfer the dirty pages generated during the previous cycle. The virtual machine will be paused when most of the memory is synchronized. Then the CPU state and remaining unsynchronized memory pages will be synchronized to the destination host, and the virtual machine can resume operation on the destination host [2].

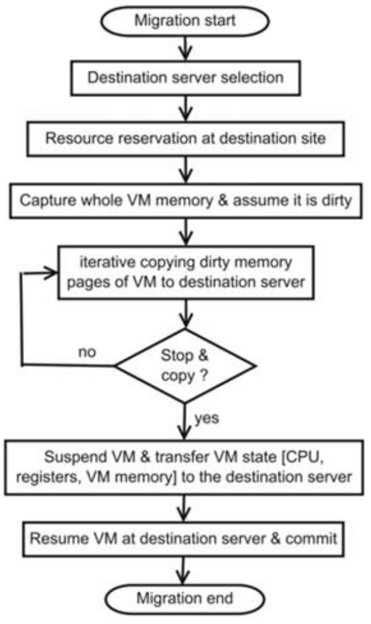

Migration start
Destination server selection
Resource reservation at destination site
Capture whole VM memory & assume it is dirty
iterative copying dirty memory pages of VM to destination server
Stop & copy ?
no
yes
Suspend VM & transfer VM state [CPU, registers, VM memory] to the destination server
Resume VM at destination server & commit
Migration end

Fig. 1 Steps of VM migration [3]

Unlike pre-copy, post-copy first stops execution of the virtual machine on the source host. The virtual machine is then restored by initiating data erasure and copying to the target host. Various methods, including proactive pushing, pre-paging, and on-demand fetching, can be used to transfer the final in-memory data. The overall migration time is predictable since every memory page is copied only once after it has already been transferred [4]. As we just mentioned, the pre-copy technique uses iteration during the push phase. Some memory pages that are updated/modified during iteration are called dirty pages. Throughout the migration phase, dirty pages on the source server are produced again. In subsequent iterations, these filthy pages are transmitted again to the target host. Additionally, some frequently visited memory pages are delivered more than once as a result, making the migration time-consuming [3].

## II. Background

Live migration, in which logical procedures are nearly identical to offline migration, means that the virtual machine is moved between various physical hosts while maintaining the regular functioning of its services. The distinction is that the migration procedure has very low downtime in order to ensure the availability of virtual machine services during the migration

Kang Zijian, Zhang Tingyu, and Dr. Burra Venkata Durga Kumar are with the School of Computing & Data Science, Xiamen University Malaysia, DULN009(B) Jalan Sunsuria, Bandar Sunsuria, 43900 Sepang, Selangor Darul Ehsan (e-mail: cst2109159@xmu.edu.my, cst2109205@xmu.edu.my, venkata.burra@xmu.edu.my).

process [7]. The time required for a virtual machine to complete the online migration can be divided into two parts: downtime and end-to-end time. The majority of time expenditure stems from the end-to-end process during which machine resources are utilized to execute the migration.

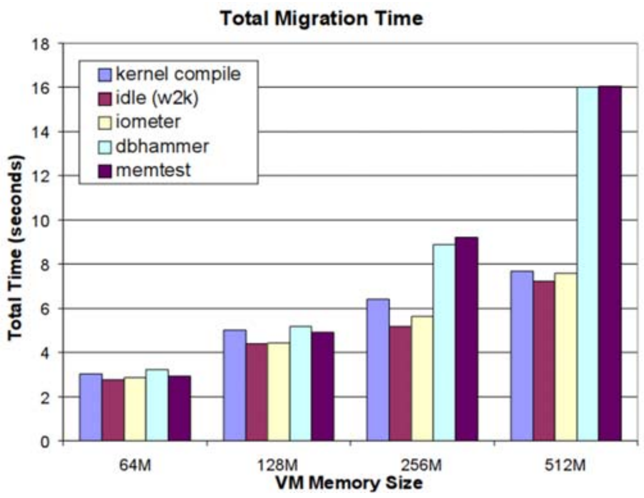


Fig. 2 Total end-to-end time for a migration [5]

Fig. 2 shows that the total end-to-end time depends strongly on the size of the VM's memory, and confirms the need to keep the VM running during most of this time [5]. This is because the virtual machine iterates after the preparation work, the first iteration copies all memory pages, and the second iteration only copies the pages modified during the first iteration, which are dirty pages. By analogy, the nth round copy is the modified interface after the n-1 iteration process [6].

We still take the Xen virtual machine as an example, whose live migration uses a pre-copy approach with simple prediction. It uses Xen's shadow page to record the modified pages, and divides the memory pages into 3 types of dirty bitmap pages:1.to_skip: The page is soiled in this iteration and can be skipped.2.to_send: The page was soiled in the previous iteration and may be migrated in this round. 3.to_fix: During the stop-and-copy phase, pages with a high dirty rate will be migrated [1]. (1) When both to_send and to_skip are 0, the memory pages have not been modified for these two rounds, so the interface is not transmitted; (2) when to_send is 0 but to_skip is 1, the memory page has become dirty in these two iterations. But these pages may continue to get dirty, so the page does not need to be transferred; (3) when to_send is 1 and to_skip is 1, indicating that the process of dirty memory pages in these two iterations has ended and the page needs to be transferred; (4) when both to_send and to_skip are 1, the page gets dirty frequently during both iterations, so the page is not delivered [6]. The rule is shown as Table I.

TABLE I
RULE OF PRE-COPY MIGRATION

| | | | | |
|---|---|---|---|---|
| To_send | 1 | 0 | 1 | 0 |
| To_skip | 0 | 1 | 0 | 1 |
| Send or not | 1 | 0 | 0 | 0 |

## III. ISSUE

There are three issues in the migration time during the memory migration.

1) If the memory size is too large, the migration time will greatly increase (refer to Fig. 2). The initial iteration's duration is inversely proportional to the VM memory size, which has an impact on the overall migration time. This is due to the fact that the first pre-copy iteration tries to replicate throughout the RAM that the entire VM has allocated. On average, the total migration time increases linearly with the size of the virtual machine [8].
2) Pre-copy technology cannot avoid downtime. A long offline transfer time often leads to poor performance of the corresponding virtual machine. This is because the virtual machine service shuts down while data are being transferred offline [9].
3) One additional question arises during the memory migration process. The question is, unless a stop condition exists, the iterative pre-copy phase may continue indefinitely. From this vantage point, it is essential to specify the cessation criteria in order to timely conclude this phase. These requirements are often characterized as minimizing VM downtime while reducing the amount of data copied between physical hosts. However, they are typically very dependent on the architecture of the hypervisor and live migration subsystem. Most things have two sides. While achieving a pre-copy phase stop, these stop conditions may have a major impact on migration performance. Even the total migration time and VM downtime encounter a non-linear trend as a result of this [8].

## IV. FLOWCHART OF ISSUE

For a better understanding of the process of iteration in pre-copy migration. Figs. 3 and 4 show how the dirty pages send to a target host.

## V.A DIRTY PAGES MIGRATION METHOD TO SOLVE DOWNTIME PROBLEM

Based on the rule of pre-copy migration and the process of n turns iteration, we imaged and designed a method for dirty pages' migration which we called double iteration. The previous pre-copy iteration sends dirty pages created in n-1th turn to target host when in nth turn. So, when the process arrives stop and copy phase, it still has to progress the nth turn dirty pages. However, the double iteration is to send dirty pages created in nth turn when in nth turn. So, it could decrease the memory that required to be progressed in stop and copy phase which decrease the downtime as well. It uses to_send and to_skip to check whether we should send this page or not.

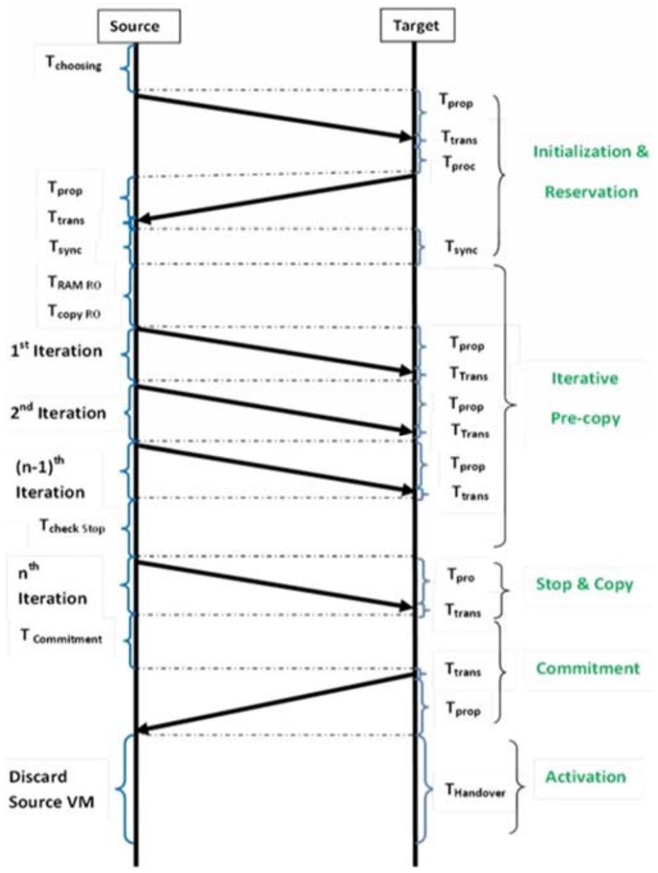


Fig. 3 The process of pre-copy iteration [10]

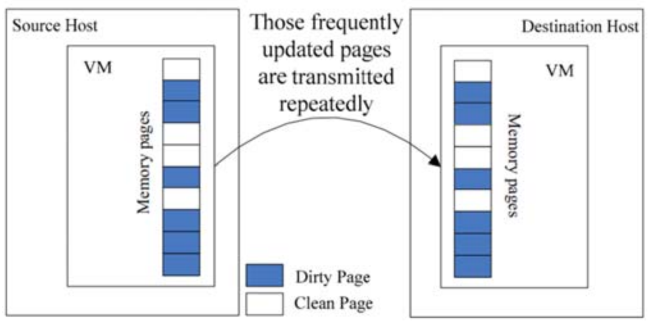
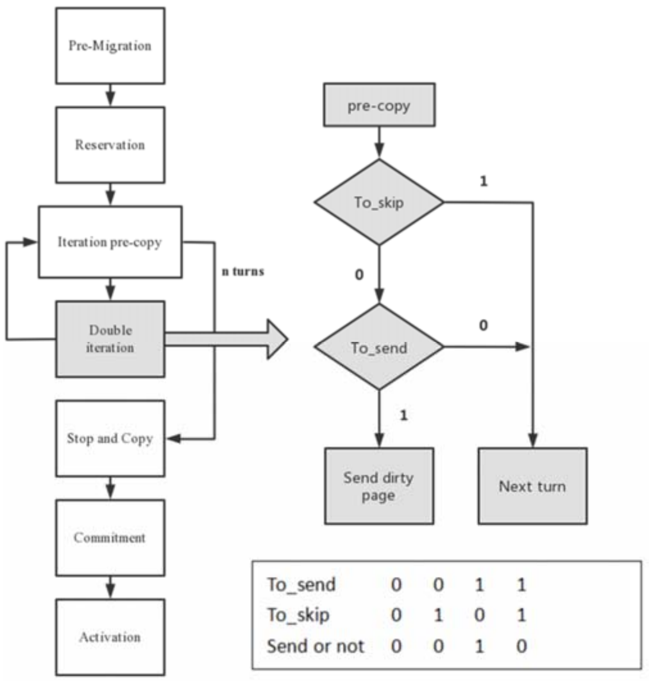


Fig. 4 One iteration in pre-copy live migration [12]

Here is the pseudo-code of a double iteration we provided to decrease the downtime:

```
1: Input: source host, target host
2: createMemorySpace (target host, sizeof source host)
3: mark all pages of source as unmodified
4:     sourcePageTable<-source host.pageTable
5:     targetPageTable<-target host.pageTable
6: for each page in sourcePageTable do
7:  mark pages migration
8:     if (page is modified) then
9:         if (to_skip = 0) then
10:            if (to_send = 1) then
11:                copyPage (source host, target host, dirty page)
12:            end if
13:        end if
14:    end if
15:        targetPageTable[page]<-target host.pages[page]
16: end for
```

Our idea is to detect the function values (i.e., to_send and to_skip) immediately after the modification is done for each memory page in the source host. The dirty pages that meet the criteria are then transferred to the destination host. Because there is only one loop in every iteration, so the time complexity will not change. And the complexity of if statement is O(1). When the problem size is large, the impact of it is negligible. The detail of process is shown as flowchart İn Fig. 5.



| | | | | |
|---|---|---|---|---|
| To_send | 0 | 0 | 1 | 1 |
| To_skip | 0 | 1 | 0 | 1 |
| Send or not | 0 | 0 | 1 | 0 |

Fig. 5 The process of double iteration

Double iteration works based on the pre-copy iteration. There is no need to do iteration again after create dirty page. Actually, the general flow of double iteration is consistent with pre-copy. However, unlike Fig. 1, the system enters the double iteration first after the pre-copy iteration. Next, VM will enter the stop and copy phase.

To better understand how the double iteration run time differs from the original iteration, here are two timelines for pre-copy technology and double iteration method.

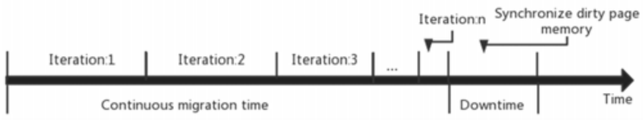


Fig. 6 Pre-copy migration memory synchronization [11]

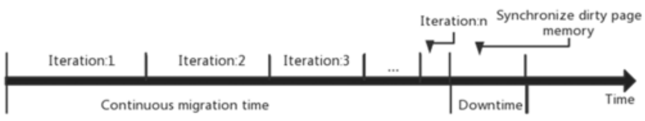


Fig. 7 Double iteration migration memory synchronization

We could see that all the iteration runtime decreases a little except first iteration. This is because in first iteration we still need copy all memory pages. By comparing the two timelines, we can see that the most reduced time is downtime. We believe this method could reduce downtime to keep virtual machines up and running.

## VI. Conclusion

In conclusion, a migration of virtual machine could be divided into two type migration. One is live migration; another is static migration. Static migrations have significant periods of time when the services in the client are unavailable, while dynamic migrations have no significant service downtime [13]. Dynamic migration actually encapsulates the configuration of a virtual machine in a file and then passes it over a high-speed network. Quickly transfer virtual machine configuration and memory running status from one physical machine to another. The virtual machine remains running during this period. The most widely used technology in a memory migration is pre-copy. The pre-copy migration phase involves restarting the destination virtual machine after the hypervisor has copied the memory contents of the source virtual machine to it. If the data are altered in memory during this procedure (known as "dirty pages"), it will be duplicated again until the dirty rate is satisfied. The remaining data are then sent to the destination virtual machine, which is then resumed on the destination host after being suspended on the source virtual machine's physical host [14].

After investigating, we found that when the file memory is large, the migration time tends to be longer. How to stop iterations under the right conditions is also a tricky problem. Sometimes downtime can take several seconds during the stop&copy phase. After studying the detailed process of pre-copy, we proposed a double iteration to reduce downtime in response to the problem of VM performance in stop&copy. This double iteration is established based on the original iteration. It checks the value of to_skip and to_send, then decide next step's action. We have provided pseudocode to verify our assumptions. Theoretically, this double iteration method will be effective and feasible. However, it does not optimize the time cost on dirty page's migration. In the pre-copy method of hot migration, frequently updated pages are repeatedly transferred [12]. Therefore, our future work is to optimize the dirty page prediction mechanism to ensure that those pages that are frequently updated are transmitted only once in the iterative process. That is, the overall migration time will be once again reduced.

## Acknowledgment

We would like to express our sincere gratitude to Dr. Burra Venkata Durga Kumar for his invaluable guidance and mentorship throughout the process of writing this paper. Thanks to Dr. Burra, we could have this valuable opportunity to complete this article. His knowledge of Unix memory management has been crucial in advancing our knowledge and bettering our comprehension of this challenging topic. His ongoing encouragement and assistance have been essential to finishing this project. We also like to express our sincere gratitude to all the authors whose works were referenced in this essay. Our research is built on the solid ground of their important contributions to the field of virtual machine migration. We have been motivated and our comprehension of the topic has been expanded by their tireless work and passion to developing knowledge in this field.

## References

[1] Wu, T., Guizani, N., & Huang, J. (2017). Related Dirty Memory Prediction Mechanism for Live Migration Enhancement in Cloud Computing Environments. Journal of Network and Computer Applications, 90(15 July), 83–89. https://doi.org/http://dx.doi.org/10.1016/j.jnca.2017.03.011

[2] Zhang, B., Luo, Y., Wang, Z., Sun, Y., Chen, H., Xu, Z., & Li, X. (2009). Whole-System Live Migration Mechanism for Virtual Machines. ACTA ELECTRONICA SINICA, 37(4), 894–899.

[3] Choudhary, A., Govil, M., Singh, G. et al. A critical survey of live virtual machine migration techniques. J Cloud Comp 6, 23 (2017). https://doi.org/10.1186/s13677-017-0092-1

[4] Zhang, F., Liu, G., Fu, X., & Yahyapour, R. (2018, January 17). A Survey on Virtual Machine Migration: Challenges, Techniques and Open Issues. IEEE Xplore. https://ieeexplore.ieee.org/abstract/document/8260891

[5] Nelson, M., Lim, B., & Hutchins, G. (2005, April 10). Fast Transparent Migration for Virtual Machines. ACM DIGITAL LIBRARY. https://dl.acm.org/doi/10.5555/1247360.1247385

[6] Zhang, W., Zhang, X., & Wang, R. (2013). Live Memory Migration for Virtual Machine Based on Dirty Pages Delayed Copy Method. Computer Science, 40(5), 126–131. https://www.jsjkx.com/EN/Y2013/V40/I5/126

[7] Zeeeitch. (2017, June 27). Principles of VM Migration. CSDN. https://blog.csdn.net/zeeeitch/article/details/73800010?ops_request_misc=&request_id=&biz_id=102&utm_term=%E8%99%9A%E6%8B%9F%E6%9C%BAprecopy&utm_medium=distribute.pc_search_result.none-task-blog-2~all~sobaiduweb~default-2-73800010.nonecase&spm=1018.2226.3001.4187

[8] S. Akoush, R. Sohan, A. Rice, A. W. Moore and A. Hopper, "Predicting the Performance of Virtual Machine Migration," 2010 IEEE International Symposium on Modeling, Analysis and Simulation of Computer and Telecommunication Systems, Miami Beach, FL, USA, 2010, pp. 37-46, doi: 10.1109/MASCOTS.2010.13.

[9] N. Tziritas, T. Loukopoulos, S. U. Khan, C. -Z. Xu and A. Y. Zomaya, "Online Live VM Migration Algorithms to Minimize Total Migration Time and Downtime," 2019 IEEE International Parallel and Distributed Processing Symposium (IPDPS), Rio de Janeiro, Brazil, 2019, pp. 406-417, doi: 10.1109/IPDPS.2019.00051.

[10] Elsaid, M.E., Abbas, H.M. & Meinel, C. Virtual machines pre-copy live migration cost modeling and prediction: a survey. Distrib Parallel Databases 40, 441–474 (2022). https://doi.org/10.1007/s10619-021-07387-2

[11] Chen, Y., Huai, J., & Hu, C. (2011). Live Migration of Virtual Machines Based on Hybrid Memory Approach. Chinese Journal of Computer, 34(12), 2278–2291. https://doi.org/10.3724/SP.J.1016.2011.02275

[12] F. Ma, F. Liu and Z. Liu, "Live virtual machine migration based on improved pre-copy approach," 2010 IEEE International Conference on Software Engineering and Service Sciences, Beijing, China, 2010, pp. 230-233, doi: 10.1109/ICSESS.2010.5552416.

[13] Yi, S. (2022, July 7). Migration of Virtual Machines. CSDN. https://blog.csdn.net/beginerToBetter/article/details/125646906?ops_request_misc=%257B%2522request%255Fid%2522%253A%2522168684086816800227472586%2522%252C%2522scm%2522%253A%252220140713.130102334.pc%255Fall.%2522%257D&request_id=168684086816800227472586&biz_id=0&utm_medium=distribute.pc_search_result.none-task-blog-2~all~first_rank_ecpm_v1~rank_v31_ecpm-4-125646906-null-null.142^v88^control,239^v2^insert_chatgpt&utm_term=%E8%99%9A%E6%8B%9F%E6%9C%BA%E5%81%9C%E6%9C%BA%E6%97%B6%E9%97%B4&spm=1018.2226.3001.4187

[14] P. P. Thakre and V. N. Sahare, "VM live migration time reduction using NAS based algorithm during VM live migration," 2017 Third International Conference on Sensing, Signal Processing and Security

(ICSSS), Chennai, India, 2017, pp. 242-246, doi: 10.1109/SSPS.2017.8071599.