I.J. Information Technology and Computer Science, **2015, 10, 8-14**
Published Online September 2015 in MECS (http://www.mecs-press.org/)
DOI: 10.5815/ijitcs.2015.10.02

Two Fold Optimization of Precopy Based Virtual Machine Live Migration

Sangeeta Sharma
Department of Computer Science & Engineering, MANIT, Bhopal, 462003, India
E-mail: sanjsharma29@gmail.com

Meenu Chawla
Department of Computer Science & Engineering, MANIT, Bhopal, 462003, India
E-mail: chawlam@manit.ac.in

Abstract—Virtualization is widely adopted by the data centers, in order to fulfill the high demand for resources and for their proper utilization. For system management in these virtualized data centers virtual machine live migration acts as a key method. It provides significant benefit of load-balancing without service disruption. Along with the various benefits virtual machine live migration also imposes performance overhead in terms of computation, space and bandwidth used. This paper analyzes the widely used precopy method for virtual machine live migration and proposes the two fold optimization of precopy method for virtual machine live migration. In the first phase, the proposed two fold precopy method reduces the amount of data sent in first iteration of precopy method. Second phase restricts sending of similar data iteratively in each subsequent iterations of precopy method by identifying frequently updated pages and keeps it till the last stop and copy iteration. In this way it reduces total migration time and total amount of data transferred. The proposed two fold precopy method is compared with precopy method and simulation results show the performance improvement of a virtual machine live migration in terms of total migration time and total amount of data transferred.

Index Terms—Live VM migration, Virtualization, Resource utilization, Total migration time.

I. INTRODUCTION

In present scenario, subscription based services are found beneficial to enhance availability and cost effectiveness of resources and services in case of simultaneous multiple user accesses. Cloud computing [1] provides IT resources or services as a subscription based service. It allows sharing the pool of resources between multiple users simultaneously and pay as per use. For this reason, datacenters need to be more capable in order to serve the exponentially increasing demand for resources such as CPU and storage. With the growth of cloud computing environment, optimal resource utilization is required in order to serve multiple users simultaneously. Virtualization [2] supports sharing of single physical machine by multiple users at the same time. Virtualization is used as an important tool to fulfill the goal of optimum resource utilization.

In the virtualized environment, resources are allocated by means of a virtual machine. Virtual machine provides the abstract view of an underlying hardware. They are the many instances of operating systems run on single physical machine. Virtual machine live migration is an important tool of virtualization. The motivation behind the use of a virtual machine live migration is load balancing, fault-tolerance and maintenance of datacenters. For example, by moving the virtual machine from one host to another host without any service interruptions by means of virtual machine live migration leads to improved resource utilization [3] and service availability. By transferring virtual machine from underutilized hosts and shutting it down, can reduce power consumption. General architecture of a virtual machine live migration from one physical machine to another is shown in Fig. 1.

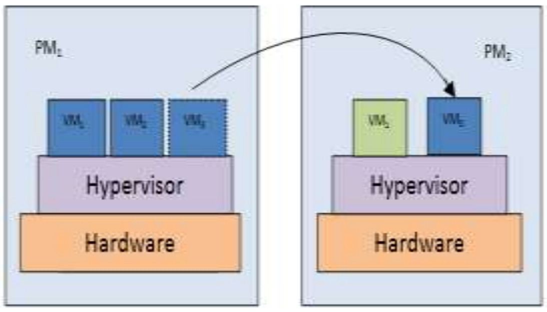

Fig.1. Virtual machine migration in virtualized physical machine

VMware [4], HyperV [5], KVM [6] and XEN [7] are some of the widely available platforms used for implementing virtualization. KVM and XEN are open source and widely used for research work.

Besides many benefits virtual machine live migration leads to performance overhead and long migration time. In order to address different issues imposed due to the migration process, this paper proposes a modified pre-copy method for virtual machine live migration. It utilizes the bitmap arrays in a way that frequently updated pages are found out and kept till the last stop-copy iteration.

This paper models virtual machine live migration performance based on theoretical analysis and empirical studies on CloudSim platform. In this paper performance of proposed two fold precopy method is compared with pre-copy method on various performance parameters such as Total migration time, Downtime, and Total data transferred.

The objective of this work can be summarized as follows:

- To reduce number of pages transferred in first iteration as well as in subsequent iterations.
- To efficiently find out frequently updated pages by utilizing features of a bitmap array.
- To reduce total time of migration process by reducing total number of pages transferred.
- Simulation results to validate the effectiveness of the proposed two fold precopy method.

The rest of the paper is organized as follows: Section 2 describes the previous work related to the proposed work. Section 3 briefly presents the background knowledge of the proposed work. Section 4 describes the precopy method and design of the two fold precopy method for virtual machine live migration is presented in Section 5. In Section 6 implementation detail of proposed work is presented. Section 7 concludes the paper.

II. Related Work

This section presents a survey of some existing works on virtual machine live migration. Research work in this field is broadly categorized into two [8] [9] types viz. precopy and postcopy method. Most of the works use precopy method due to its fault tolerance nature over the post copy method. In the precopy method some of the basic techniques [10] are memory ballooning, dynamic rate limiting and rapid page dirtying. Removal of unused memory during a migration process is performed in Memory ballooning method. Dynamic rate limiting adopts the varying bandwidth limit during each precopy iteration. By identifying and sending frequently updated pages at the last iteration, Rapid page dirtying method controls the dirty rate. From several available methods, they can be grouped into different categories based on their techniques. Some of them are compression based, CPU scheduling based and prediction based.

Compression based methods reduce the total data transferred during iterative data transfer phase of migration method. Based on the memory page characteristics Jin et al. [11] presents an adaptive zero-aware memory compression algorithm. Its compression algorithm is dynamic and lossless with small overhead.

Ma et al. [12] uses Run length encoding to transfer only non-empty memory pages in compressed form. Live migration of large virtual machine Svard et al. [13] presents a Delta compression technique. It stores delta pages which is a difference between versions of a page, instead of full page. These delta pages are further compressed using XOR binary run length encoding (XBRLE) to increase migration throughput. It significantly reduces migration downtime along with reduced migration time. Delta compression successfully migrate very large virtual machine that fails while migrating using the traditional method. Svard et al. [14] also presents another dynamic page transfer reordering and compression technique. In this technique, it combines dynamic page transfer reordering with delta compression, in order to reduce both the total migration time and downtime. Dynamic page transfer reordering prioritizes less frequently updated pages over frequently updated pages. In this way, it reduces possibility of resending these pages again.

Compression based methods reduce the amount of data transfer, but the overhead imposed due to compression at source side and decompression at destination is not well evaluated. This also does not consider effect of repeated transmission of a same page during migration.

Performance of migration processes is highly related to the memory dirty rate. For better performance memory dirty rate needs to be kept as low as possible. Towards this goal Jin et al. [15] presents an approach based on CPU scheduling for optimizing the virtual machine live migration method. This method schedules the CPU in such a way that the memory dirty rate can be fixed within the acceptable limit. Liu et al. [16] presents a CPU scheduling based approach combined with the recovering system, Viz. checkpointing/recovery and trace/reply. It schedules a CPU to adjust the log generation rate since it synchronizes the state of log files between source and destination. CPU scheduling based methods are able to reduce memory dirty rate, but impact of CPU scheduling on the application performance running within the virtual machine is ignored.

To reduce the total data transferred some of the methods try to find out the frequently updated pages and keep it till the last stop and copy iteration, which also reduce total migration time and downtime. One of the research works based on this idea is presented by Alamadari et al. [17]. It represents cache replacement policy to trace out the reuse pattern of memory pages. It calculates the reuse distance of a page when the same page is reused again. Frequently updated pages are modified repeatedly and having less reuse distance as compared to less frequently updated pages. Ma et al. [18] presents an improved precopy approach, in which it uses the bitmap which keeps the frequently updated pages till the last iteration of migration. It also bounds the number of iteration in migration method up to five times.

A memory modification probability prediction model is presented by Wu et al. [19]. The model monitors the modification rate of each memory page at present iteration and tries to predict the probability of modification rate in next iteration. In this way, most of the frequently updated pages are found out and kept till last iteration. Hu et al. [20] presents a time series prediction method, in which it uses historical statistics of memory pages to identify the frequently updated pages in a future period. Li et al. [21] also uses historical statistics to identify frequently updated pages. It presents a performance and energy model for the prediction of virtual machine migration cost. Success of the methods based on identification of frequently updated pages highly depends upon the correct prediction process used by it. All above methods discussed here is based on precopy method.

One of the methods based on postcopy method is presented by Hines et al. [22]. In postcopy method, virtual machine is resumed at destination in a first iteration with the transmission of CPU state. Remaining memory pages are transmitted subsequently. It uses the Adaptive prepaging and Dynamic self-ballooning mechanism to remove redundant pages and free memory pages before transmission. It improves the migration method by lowering unnecessary data transfer. Another method based on postcopy method is an advanced virtual machine dynamic consolidation system presented by Hirofuchi et al. [23]. It consolidates the load by migrating the virtual machine with response to every changing resource usage.

Strunk et al. [24] and Voorsluys et al. [25] determine the cost of a virtual machine live migration and discover parameters that affect migration costs.

III. Background Knowledge

This section presents the background knowledge of migration and analyzes the problems which affect the performance of a virtual machine live migration. Virtual machine migration is considered as live migration when the virtual machine moves from one physical machine to another without halting the running virtual machine. The basic need for the virtual machine live migration is to maintain the liveness of virtual migration. This can be done by keeping Downtime (described in 3.1) as minimum as possible. Virtual machine consists of a number of components which need to be transferred during a migration. These components include CPU registers state, Device state, memory and storage state. CPU and device register information are very small in size and have a negligible impact on performance. Virtual machine storage is very large in size and greatly affects the performance of a migration process. But in present scenario it is assumed that datacenters are equipped with NAS (Network attached storage) by which storage can be accessed by all physical machines. So there is no need to transfer storage data. Virtual machine memory size is comparably larger than CPU registers data and updated continuously while virtual machine is running. Transferring continuously changing large size virtual machine memory is the main issue in a virtual machine live migration which highly affects the migration process. Proposed two fold precopy method presented in this paper efficiently transfers the virtual memory with its CPU register data with a minimum service interruption delay.

After knowing the various components which are transferred during migration, next issue to address is the way (method) in which these components will be transferred. There are generally three types of migration methods [9] viz. Stop and copy, Postcopy, and Precopy as shown in Fig. 2.

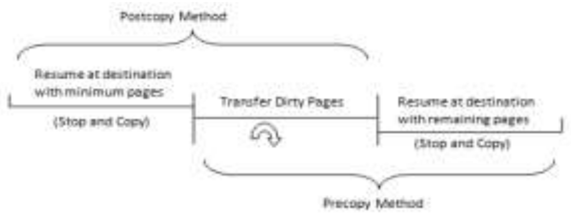
Postcopy Method
Resume at destination with minimum pages
(Stop and Copy)
Transfer Dirty Pages
Resume at destination with remaining pages
(Stop and Copy)
Precopy Method

Fig.2. Types of live virtual machine migration

Type 1: Stop and Copy is simple but it is an offline method. In this method, virtual machine can be stopped at a source and then move to another machine. Service interruption is a major drawback of this method.

Type 2: Post Copy method initially transfers the CPU register with a minimum required memory pages and resumes a virtual machine at the destination machine. Remaining data is transferred subsequently. It provides the low service interruption. One major drawback of post-copy method is that, it cannot recover from failure of the virtual machine at destination side during migration [22]. If the destination side virtual machine is crashed then the running virtual machine will be unsynchronized and all remaining data will be lost.

Type 3: Pre-Copy method is widely adopted method due to its fault proof nature. It also gives the low service interruption by transferring virtual machine in a first iteration then remaining updated data iteratively till both sides have a synchronized copy of a virtual machine. In the final iteration virtual machine resume at a destination machine by sending the remaining data. Proposed two fold precopy method presented in this paper is also based on precopy method and improves its performance.

A. Performance Parameters

There are various performance parameters and factors that affect the migration process and need to handle efficiently. These performance parameters are measured to show the effectiveness of the proposed work as compared to precopy method. They are as follows:

Total migration time (T_{mig}): It is the total time taken by a migration process for its completion. It is defined as $T_{mig} = \sum_{i=0}^{N} T_i$, where T_i is the time taken by the ith iteration of migration and N is the number of iterations. Minimum total migration time leads to fast migration process.

Downtime (T_D): It is the time for which service is interrupted and for higher service availability, it is needed to keep this value minimal. It is defined as time taken in last iteration of migration process or difference of time between Virtual machine resumed at destination side (T_r) and virtual machine stopped at source side (T_s).

Total data (pages) transferred (V_{mig}): It is the amount of data transferred during the migration process. More data transfer directly affects the migration time. It is defined as $V_{mig} = \sum_{i=0}^{N} V_i$, where, V_i is the volume (number of pages) of data transferred in ith iteration and N is the number of iterations. This value should be minimal.

Overhead: It is the additional data transferred during migration which directly affects the time taken for virtual machine migration. To describe overhead trade-off of the migration method, Redundancy ratio R_d is defined as $R_d = \frac{V_{mig}}{V_{mem}} \times B$, Where R_d indicates additional data transferred during migration, V_{mig} is total data transferred during migration, V_{mem} virtual memory size and B is data rate. Bigger the ratio more will be the overhead caused by iteratively dirtied memory during migration. The overhead occurred due to migration process leads to degradation of application performance running on the virtual machine.

Other factor**s** which affect the performance of a virtual machine live migration are as follows:

Memory dirty rate: It shows the memory changing rate by which the memory gets dirty**.** With the lower memory dirty rate, a migration process sooner reaches the stopping condition and results in a reduction of migration time.

Data rate: It is the data transferring rate during the migration process. It is inversely proportional to the migration time which implies higher data transfer rate sends more data in less time.

$$Data\ Rate = \frac{1}{Migration\ Time}$$

IV. Precopy Method

To better understand the proposed work it is necessary to study precopy method. Working of precopy live migration method consists of:

- Complete virtual machine image is transferred in first iteration of precopy migration.
- Data updated (dirty pages) in first iteration is iteratively transferred in subsequent iterations to maintain the consistency. For further iterations dirty pages are iteratively transferred until reached to a predefined threshold value i.e. maximum number of iterations or amount of data remaining to transfer.
- After fulfilling the stopping condition precopy method reaches its last iteration known as stop and copy. In this iteration, virtual machine is terminated at source side and transfers remaining data to the destination. Finally, virtual machine is resumed at destination and migration process ends. Iterations performed in precopy method are shown in Fig. 3.

In the precopy method three types of bitmap [17] called TO_SEND, TO_SKIP and TO_FIX, are used to improve the migration process. These bitmaps are used to find pages which are transferred in current iteration, with the objective that the frequently updated pages are kept till the last iteration. Bitmap TO_SEND holds dirty pages of previous iteration and need to be transferred to the destination. Bitmap TO_SKIP holds frequently updated pages which are not transferred in current iteration. Bitmap TO_FIX holds those pages which are transferred in last stop and copy iteration. In this way, bitmaps help to mark the frequently updated pages in order to reduce the transfer of same pages repeatedly.

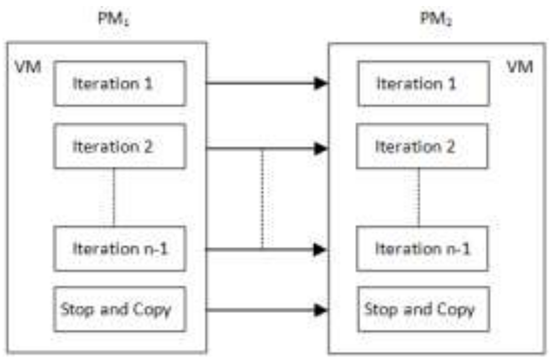

Fig.3. Iterations in Precopy method

H (remaining data) and N (maximum number of iterations) are two thresholds which are used as stopping condition for performing last stop and copy iteration.

Pseudo code for precopy method is described as follows.

Pseudo code: Precopy Method

```
1. D <- Send complete VM                    // first iteration
2. i <- 0
3. while D>H & i<N
4. Send pages TO_SEND ∩ TO_SKIP
5. Perform stop and copy operation to send the pages to
   destination                              // last iteration
6. Resume VM on another host
```

After studying the precopy method, following issues are found:

- Complete virtual machine image is transferred in first iteration of precopy migration which is large in size.
- Data updated in first iteration is transferred in subsequent iterations, which leads to repetitive transfer of same data.

In both cases amount of total data transferred increases, which further increases total migration time.

Above issues can be solved by applying following two way solution. First instead of sending complete virtual machine image in a first iteration of migration method, only unmodified pages are sent. Second for further iterations frequently updated pages are traced by utilizing the bitmaps and historical data (memory pages which are updated in previous iterations). Idea behind this is to keep frequently updated pages till last iteration which reduce repetitive transfer of same pages. Sending only less frequently updated pages in each iteration starting from the first iteration greatly reduce the total data transferred by avoiding transfer of complete virtual machine image in first iteration. In the following section proposed two fold precopy method is discussed.

V. Proposed Work

This section presents proposed two fold precopy method, which reduces the transfer of similar frequently updated pages iteratively and improves the performance of live migration process.

Two fold precopy method works in two phases:

First phase: It simply sends only unmodified memory pages and reduces the data transferred in first iteration. For this it uses the dirty log bitmap shown in Fig. 4. When the memory pages are modified then the corresponding bit in dirty-log bitmap is 1. Any page in dirty-log bitmap, whose corresponding bit is 0 (unmodified page), is sent to a destination in a first iteration. For this task two fold precopy method does not imposes any extra overhead. Finally, in place of complete virtual machine only unmodified pages are transferred which reduces the amount of data transferred in first iteration. Amount of data transferred in first iteration depends on the virtual machine workload type. If the virtual machine is lightly loaded or less write intensive (modification or dirty rate), then the number of unmodified pages are more. Whereas in highly loaded or more write intensive virtual machines, unmodified pages are less. But in both cases amount of data transferred in first iteration is reduced. First phase provides the initial writable working set (WWS [10]), which is further optimized in second phase or in further iterations.

A	B	C	D	E	F	G	H	I	J
1	1	1	1	0	0	0	0	1	1

Fig.4. Dirty log bitmap

Second phase: It further improve the writable working set (WWS) obtained from a first iteration, in order to trace the frequently updated pages. More accurate writable working set (WWS), helps to reduce the iterative transfer of similar memory pages in subsequent iterations. Precopy method compares TO_SEND and TO_SKIP bitmaps and the result of an intersection of these two lists are the pages which are transferred in current iteration. But still there is a possibility that the pages already transferred in current iteration may be modified again in next iteration. It is required to trace the frequently updated pages more accurately. For this purpose along with the TO_SEND and TO_SKIP bitmaps, one more bitmap TO_SEND_h[i] [20] is used which stores historical data of modified pages. TO_SEND_h[i] bitmap stores those pages which are modified in iteration '*i*' and number of times they are modified. Organization of TO_SEND and TO_SEND _h[i] bitmap is shown in Fig. 5.

For each iteration, TO_SEND bitmap is compared with TO_SEND_h[i] bitmap and threshold T_1 (value of higher dirty rate). Pages whose value is more than threshold T_1 is considered as frequently updated pages. Value of threshold T_1 is changed for every iteration and it can be defined as:

$$T_1 = \left\lfloor \frac{\text{Max [page modification rate]} + \text{Min [page modification rate]}}{2} \right\rfloor \quad (1)$$

Only the page which satisfies following condition will be transferred:

$$\text{Number of times page modified} < T_1 \quad (2)$$

It means pages having less modification rate than the threshold value are considered as less frequently updated pages.

TO_SEND
A
B
C
D
E
F
G
H
I
J

TO_SEND_h[i]	
A	5
B	4
C	3
D	2
E	4
F	9
G	5
H	4
I	8
J	7

Fig.5. TO_SEND and TO_SEND_h bitmap

In this way two fold precopy method trace the frequently updated pages in each iteration and send them to a last stop and copy iteration.

Pseudo code for two fold precopy method is described as follows.

Pseudo code: Two Fold Precopy Method

```
1.  D <- Send only Unmodified data                    // first phase
2.  Modified pages are stored into TO_SEND bitmap
3.  i <- 0
4.  while D>H & i<N                                   // Second phase
5.  Compare TO_SEND and TO_SEND_h[i] and Calculate
6.  $T_1 = \left\lfloor \frac{\text{Max [page modification rate]} + \text{Min [page modification rate]}}{2} \right\rfloor$
7.  Send pages P(Number of times page [a] modified < $T_1$)
8.  i = i +1
9.  Perform stop and copy operation to send the pages to
    destination                                       // last iteration
10. Resume VM on another host
```

VI. Performance Evaluation

In order to evaluate the performance of two fold precpoy method, Cloudsim simulator is used. Effectiveness of the two fold precopy method is shown by comparing it with precopy method with respect to metrics, Total migration time, Downtime, and Total data transferred. CloudSim simulator is a framework for modeling and simulating a cloud environment. It is also widely used within the group of researchers and industry-based developers. CloudSim helps to investigate the specific design issues without getting concerned about the low-level details of a cloud environment.

A. Simulation setup and measurement

Experimental environment consist of one datacenter, which contains two hosts. Four virtual machines are created each with different memory sizes. 128M, 256M, 512M and 1024 memory sizes for VM_1, VM_2, VM_3, and VM_4 respectively. All virtual machines migrate from one host to another separately.

Dirty matrix is a data structure used to show memory pages of a virtual machine modified in last iteration called dirty log bit. In this '0' entry corresponding to the pages shows that these pages are not modified and transferred in a first iteration whereas remaining modified pages are stored in TO_SEND bitmap. Here four bitmap arrays are also used such as TO_SEND, TO_SKIP, TO_FIX, and TO_SEND_h. For each iteration pattern is generated by the dirty matrix and based on this, behavior of precopy method and two fold precopy method is checked. Simulation results are shown here in the form of three graphs.

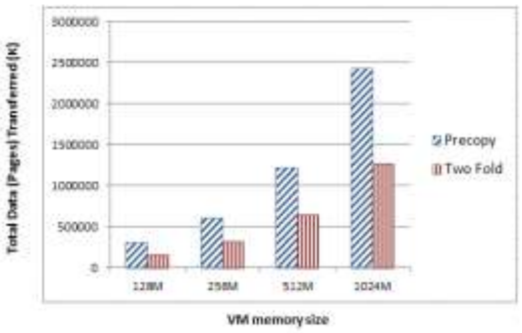

Fig.6. Total data transferred

Total data transferred: It shows the amount of data (pages) transferred during the migration of all four virtual machines. Graph in Fig. 6 clearly shows that amount of data transferred in two fold precopy method is less as compared to precopy method.

Total migration time: Fig. 7 shows the total time taken by the migration of all four virtual machines individually. The two fold precopy method takes less time for the migration as compared to precopy method.

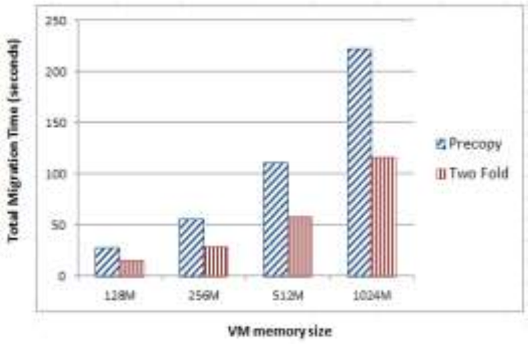

Fig.7. Total migration time

Downtime: Graph in Fig. 8 shows that Downtime is approximately equal to both two fold precopy method and precopy method.

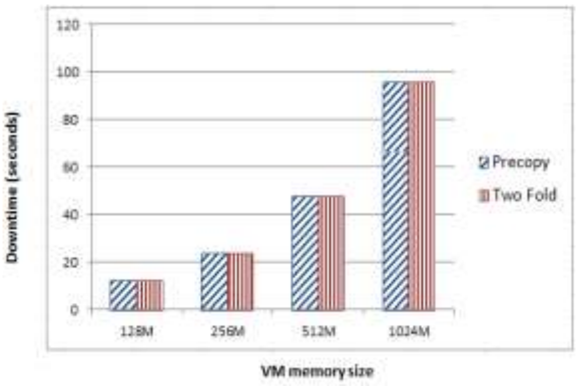

Fig.8. Downtime

Performance evaluation of two fold precopy method shows that it is more effective than precopy method in terms of total migration time, total data transferred and nearly same in terms of downtime.

VII. CONCLUSION

This paper proposes a new two fold precopy method which optimizes the precopy method. The simulation results show that two fold precopy method perform better than precopy method. It reduces the amount of data sent in first iteration by sending only unmodified pages instead of full virtual machine image. It traces out the frequently updated pages for the subsequent iterations, by using the historical statistics of dirty pages. The main feature of the two fold precopy method is that Threshold (high dirty page) value need not to be decided manually. It is calculated based on the historical statistics of dirty pages, which is dynamic in nature. The two fold precopy method does not impose any overhead for the first phase, and for the second phase only one array is used which only stores the statistics of modified pages. The capturing of historical statistics and processing of method is implemented in such a way that it does not affect the performance of a virtual machine.

Two fold precopy method improves the precopy method in terms of total migration time and total data transferred, but downtime still needs improvement. Downtime will be further optimized as a future work. Apart from this effectiveness of two fold precopy method will be tested for real time implementation. Presently, the two fold precopy method works for local area network but further it will be extended for wide area network.

REFERENCES

[1] M. I. Alam, M. Pandey, and S. S. Rautaray, "A Comprehensive Survey on Cloud Computing", *International Journal of Information Technology and Computer Science (IJITCS)*, no. 2, pp. 68-79 (2015).

[2] N. el-Khameesy, and H. A. R. Mohamed, "A Proposed Virtualization Technique to Enhance IT Services."

International Journal of Information Technology and Computer Science (IJITCS) 4, no. 12, pp. 21-30 (2012).

[3] M. Mishra, A. Das, P. Kulkarni, and A. Sahoo, "Dynamic resource management using virtual machine migrations," Communications Magazine, IEEE, vol. 50, no. 9, pp. 34–40, 2012.

[4] "VMware Incorporation," http://www.vmware.com

[5] "Microsoft Corporation," http://www.microsoft.com/en-us/server-cloud/hyper-v-server/

[6] "KVM," http://www.linux-kvm.org

[7] 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," ACM SIGOPS Operating Systems Review, vol. 37, no. 5, pp. 164–177, 2003.

[8] S. Sharma and M. Chawla, "A technical review for efficient virtual machine migration." In Cloud & Ubiquitous Computing & Emerging Technologies (CUBE), 2013 International Conference on IEEE, pp. 20-25, 2013.

[9] S. Sharma, and M. Chawla, "A Review on Efficient Virtual Machine Live Migration: Challenges, requirements and technology of VM migration in cloud" *International Journal of Cloud Computing and Services Science (IJ-CLOSER)* 3, no. 6 (2014).

[10] C. Clark, K. Fraser, S. Hand, J. G. Hansen, E. Jul, C. Limpach, I. Pratt, and A. Warfield, "Live migration of virtual machines," in Proceedings of the 2nd conference on Symposium on Networked Systems Design & Implementation-Volume 2. USENIX Association, pp. 273–286, 2005.

[11] H. Jin, L. Deng, S. Wu, X. Shi, and X. Pan, "Live virtual machine migration with adaptive, memory compression," in Cluster Computing and Workshops, CLUSTER'09. IEEE International Conference on IEEE, pp. 1–10, 2009.

[12] Y. Ma, H. Wang, J. Dong, Y. Li, and S. Cheng, "Me2: Efficient live migration of virtual machine with memory exploration and encoding," in Cluster Computing (CLUSTER), 2012 IEEE International Conference on IEEE, pp. 610–613, 2012.

[13] P. Svärd, B. Hudzia, J. Tordsson, and E. Elmroth. "Evaluation of delta compression techniques for efficient live migration of large virtual machines." *ACM Sigplan Notices* 46, no. 7 (2011): 111-120.

[14] P. Svard, J. Tordsson, B. Hudzia, and E. Elmroth, "High performance live migration through dynamic page transfer reordering and compression," in Cloud Computing Technology and Science (CloudCom), 2011 IEEE Third International Conference on IEEE, pp. 542–548, 2011.

[15] H. Jin, W. Gao, S. Wu, X. Shi, X. Wu, and F. Zhou, "Optimizing the live migration of virtual machine by cpu scheduling," *Journal of Network and Computer Applications*, vol. 34, no. 4, pp. 1088–1096, 2011.

[16] Z. Liu, W. Qu, W. Liu, and K. Li, "Xen Live Migration with Slowdown Scheduling Algorithm", The 11th International Conference on Parallel and Distributed Computing, Applications and Technologies, pp 215-221, 2010.

[17] J. Alamdari and K. Zamanifar, "A reuse distance based precopy approach to improve live migration of virtual machine," in Parallel Distributed and Grid Computing (PDGC), 2012 2nd IEEE International Conference on IEEE, pp. 551–556, 2012.

[18] F. Ma, F. Liu, and Z. Liu, "Virtual Machine Migration Based on Improved Pre-copy Approach," In Proc. IEEE Int'l Conf. Software Engineering and Service Sciences, pp.230-233, 2010.

[19] Y. Wu and M. Zhao, "Performance modeling of virtual machine live migration," in Cloud Computing (CLOUD), 2011 IEEE International Conference on. IEEE, pp. 492–499, 2011.

[20] B. Hu, Z. Lei, Y. Lei, D. Xu, and J. Li, "A Time-Series Based Precopy Approach for Live Migration of Virtual Machines", IEEE 17th International Conference on Parallel and Distributed Systems, pp 947-952, 2011.

[21] H. Liu, C. Xu, H, Jin, J. Gong, X. Liao, "Performance and energy modeling for live migration of virtual Machines", ACM 2011.

[22] M. R. Hines and K. Gopalan, "Post-copy based live virtual machine migration using adaptive pre-paging and dynamic self-ballooning," in Proceedings of the 2009 ACM SIGPLAN/SIGOPS international conference on Virtual execution environments. ACM, pp. 51–60, 2009.

[23] T. Hirofuchi, H. Nakada, S. Itoh, and S. Sekiguchi, "Reactive consolidation of virtual machines enabled by postcopy live migration," in Proceedings of the 5th international workshop on Virtualization technologies in distributed computing. ACM, pp. 11–18, 2011.

[24] A. Strunk, "Costs of virtual machine live migration: A survey," in Services (SERVICES), 2012 IEEE Eighth World Congress on. IEEE, pp. 323–329, 2012.

[25] W. Voorsluys, J. Broberg, S. Venugopal, and R. Buyya, "Cost of virtual machine live migration in clouds: A performance evaluation," in Cloud Computing. Springer, pp. 254–265, 2009.

Authors' Profiles

Sangeeta Sharma: She is currently working towards her PhD degree under Computer Science and Engineering department, at Maulana Azad National Institute of Technology, Bhopal, India. Her main research interest includes virtualization technology for cloud computing.

Meenu Chawla: She received Bachelor of Engineering degree in Computer Science and Engineering from MANIT Bhopal, India, and Master in Technology from IIT Kanpur, India. She was awarded the Ph.D. degree in Computer Science and Engineering from MANIT Bhopal, India. She is currently a Professor in the Department of Computer Science and Engineering, at Maulana Azad National Institute of Technology, Bhopal, India. Her research interests includes adhoc networks, wireless networks etc. She has authored and published over 50 research papers in various International conference and International journals.

How to cite this paper: Sangeeta Sharma, Meenu Chawla,"Two Fold Optimization of Precopy Based Virtual Machine Live Migration", International Journal of Information Technology and Computer Science(IJITCS), vol.7, no.10, pp.8-14, 2015. DOI: 10.5815/ijitcs.2015.10.02