International Journal of Engineering

Journal Homepage: www.ije.ir

# A Versioning Approach to VM Live Migration

M. Tajamolian, M. Ghasemzadeh*

*Computer Engineering Department, Yazd University, Yazd, Iran*

*PAPER INFO*

*Paper history:*
Received 10 July2018
Received in revised form 27 Septemebr 2018
Accepted 26 October 2018

*Keywords*:
*Virtual Machine Live Migration*
*Pre-Copy & Post-Copy*
*QAVNS*
*Quadruple Adaptive Version Numbering Scheme*
*Informational Object*
*Cloud Computing*

*ABSTRACT*

In the context of virtual machines live migration, two strategies called “pre-copy” and “post-copy” have already been presented; but each of these strategies works well only in some circumstances. In this paper, we have a brief presentation of QAVNS and then introduce a new approach which is based on the concept of "informational object", assigning QAVNS-scheme-revision number, and observing the changes. In this approach, the total virtual machine memory is considered as a QAVNS informational object that is constantly changing. In this regard, we have defined some criteria and presented an algorithm by which the hypervisor can detect the current behavior of the virtual machine, and automatically select the virtual machine migration strategy from the two pre-copy and post-copy options. We evaluated the implementation & simulation platforms considering the state of the art available technologies, mostly CloudSim, SimGrid and DartCSim+. Formal analysis shows that applying the proposed scheme and the proposed algorithm can significantly improve the live migration process of virtual machines.

***doi**: 10.5829/ije.2018.31.11b.06*

## 1. INTRODUCTION[1]

Technologies that use virtual machines, especially cloud computing, play the most important role in the currently being used IT infrastructures. In this regard,virtual machine live migration has been one of the significant issues in the related research works.

Various methods have been proposed for live migration of virtual machines, which have strengths and weaknesses relative to each other and will be referred to in the third part of the article. Based on the concept of the "Informational Object" and "the Adaptive Version Numbering Scheme" for tracking the changes in the informational objects which is presented in the fourth part of the article, we have tried to propose a new approach to the issue of choosing the best method for VM live migration and providing criteria for it.

The article is organized as follows: In the following section, some of the basic concepts are briefly cited. In the third section, we will review the research background in the field of virtual migration of virtual machine memory. In section 4, an adaptive numeric mapping scheme is introduced to track changes in information objects. In section 5,our proposed approach is presented in detail. In section 6, we have explained the implementation of our proposed approach and presented an algorithm. In section 7, the analysis of the proposed algorithm is discussed. Finally in section 8, conclusions and suggestions for further research are presented.

## 2. BASIC CONCEPTS

One of the key issues in cloud computing is the ability to maintain a workload balancing on different cloud-based servers. In this regard, various strategies have been proposed, such as "task migration" from one machine to another [1]. But in many cases, only the migration of tasks can not solve the problem of load imbalance and requires the entire machine's migration. Here, the concepts of "virtual machines" and "live migration" have been remedied [2].

Due to the growing importance of technologies that use virtual machines (especially cloud computing), and

*Corresponding Author's Email: *m.ghasemzadeh@yazd.ac.ir* (M. Ghasemzadeh)

Please cite this article as: M. Tajamolian, M. Ghasemzadeh, A Versioning Approach to VM Live Migration, International Journal of Engineering (IJE), IJE TRANSACTIONS B: Applications Vol. 31, No. 11, (November 2018) 1838-1845

to achieve the following benefits, the use of virtual machines has become a defacto standard in information technology infrastructure [3-7]:

- More performance in usage of hardware resources
- Rise in system stability
- Energy saving
- Reduce the amount of physical space consumed in data centers
- Set up experimental environments
- Increase software lifecycle
- Faster delivery of infrastructure and servers
- Decrease dependencies on specific hardware
- Easier management of servers and services, and costs reduction
- Facilitate backup, restore, and snapshots
- Increase the percentage of server active time (continuity of service)
- Improved exposure to critical situations
- Easier migration to cloud computing

"Virtual Machine Migration" means that a snapshot of an operating system running on a virtual machine, along with all processes, system status and memory, network communications, storage devices, and other related items from a hardware platform (known as a host machine) is transferred to another hardware platform and then will continue to run it on the new machine. Migration process is done in a similar manner to the virtual machine suspension process, except that instead of storing the virtual machine status in status image files, this information is transferred via the network to another hardware machine and the task of rebuilding the virtual machine will be performed on the destination machine (instead of the origin machine). The term "Live Migration" means that the virtual machine chosen for migration, without being powered off or even suspended for a long time, at the same time as it is actively engaged in answering received requests, is transferred from one machine to another machine which is a separate hardware.

The most common way for a virtual machine to be transferred from a hardware platform to another hardware platform without full stopping its guest operating system is a process as follows:

1. Stopping step: Suspend the virtual machine in the source physical machine.
2. Copy step: Transferring memory pages and virtual machine status information, from the physical source machine to the destination machine.
3. Restart step: The virtual machine starts operating in the destination physical machine.

Some researchers have called this process "Naive Method" [8].

As the aforementioned method leads to a sharp decrease in the efficiency of the virtual machine, researchers have always been looking for better ways. In most researches, the virtual machine migration process is generally divided into three stages [9]:

1. Push stage: While the source virtual machine continues to run, its memory pages are copied to the destination machine. Obviously, pages that change during this step should be resent to the destination machine again.
2. Stop and Copy stage: The virtual machine is suspended, and other memory pages and status information are transmitted to the destination machine. Then the virtual machine in destination physical machine begins and resumes the suspended jobs.
3. Pull stage: The new virtual machine in the destination if need to access pages of memory that (for any reason) are not transferred previously, will send appropriate requests for transferring of them to the source, and receive them properly.

Choudhary et al. in their article have a very comprehensive overview of various aspects and concepts of this field of research, such as the various techniques of VM live migration and their comparison, performance issues and security issues [10]. One of the beneficial features in CDCs[2] is "Multiple VM Migration". This means that multiple VMs begin to migrate due to many resons such as load balancing.

## 3. LITERATURE REVIEW

A lot of research has been done on the virtual machines migration. In this section, several basic and recent researches, along with their relationship with the current research are considered.

One of the proposed methods is "Pure demand-migration", which was presented many years ago by Zayas [11]. In his method, important memory pages that are related to the process' data structures are transferred to the destination host machine within a short "stop and copy" stage. After that, the destination machine starts and transfers the pages that are not transferred as soon as first request access to them through the network (pulling). Although this method leads to a shorter "suspension time", but "total migration time" takes much longer. Another critique of this method is its weak performance. The target machine's performance won't reach to the desired level until a high percentage of all memory pages are transferred from the source machine to the destination due to "page fault" requests on the destination machine. Because, for each unsuccessful memory page access request (page fault), one memory page transferring should take place on the network that results in a reduction in overall performance. Of course, Zayas's research focused on the issue of "process

[2]Cloud Data Centers

migration", which somehow shaped the topic of research on "virtual machine live migration" in the next two decades.

The most famous and widely used method is the "pre-copy", which was first suggested by Clark et al. [9]. In this method, the "push stage" is repeated constantly. In other words, in the n-th iteration, those pages of memory that have changed during the (n-1)-th iteration, will be transferred. Of course, always there are pages that change so quickly and continuously that they can not be transferred by iterative pushings. These are named "Writable Working Set" or WWS [9]. The push stage is repeated as long as the WWS volume is reduced to a significant degree, or that the number of iterations of this stage exceeds a predetermined limit. Therefore, the number of iterations that the push stage should be repeated to a large extent depends on WWS, and it is directly proportional to the workload of the source virtual machine.

Hines and colleagues [12] proposed a reversal approach to the "Pre-copy", known as the "Post-copy" approach. In this approach, unlike the pre-copy method, instead of first moving the memory pages from the running source machine to the destination machine, initially operation of source machine is suspended and the processor state information is transferred from the source to the destination. Then immediately virtual machine running will proceeds at the destination just from the point where it was suspended in the source machine. Obviously, the destination virtual machine needs access to the memory pages that are located at the source and therefore, for each unsuccessful access request to the memory pages (called Page Fault), a request will be sent to the source machine through the network, and that page is transferred back to the destination machine.

To complete the post-copy method, there are four suggested techniques that named: Demand-Paging, Active Push, Prepaging, and Dynamic Self-Ballooning (DSB). These techniques differentiate post-copy from the "Pure demand-migration" approach provided by Zayas [11].

Surprisingly, despite the undeniable similarity of Hines and colleagues' method in their article (named Post-copy) to the one that proposed by Zayas, they have not mentioned anything to him in their article! We think that this can be considered as a kind of plagiarism and should be blamed and criticized.

The post-copy approach seems to have attracted the attention of researchers in this field so far, and many of them have tried to improve it and solve its problems and/or combine it with the pre-copy approach. As the latest advancement in this field, can refer to a combination of pre-copy and post-copy approaches proposed by Sahni and his colleague [13].

Now, the undeniable importance of VM live migration in improving the performance of cloud computing services has proven. In recent years, research trends in this area have led to issues such as accelerating the migration process of virtual machine storage devices [14], reducing the negative effects of VM migration interference on the performance of each other [15], and the automation and scheduling management of VMs live migration [16]. This ongoing research is in the domain of automation of VMs live migration too.

## 4. ADAPTIVE VERSION NUMBERING SCHEME

One of the important issues in live migration of virtual machines is how to monitor the changes of virtual machine memory in the migration process and decide on how to migrate based on it. In this regard, having a scheme that can efficiently displays and tracks virtual machine memory changes as the versions of an informational object, could be useful.

We present a "Quadruple Adaptive Version Numbering Scheme" with multi-purpose usability. "Adaptive" means that this scheme has the capability, without changing its structure, to track the various versions and editions of files, software packages, project output documentation, designs, rules, manuals, style sheets, drawings, graphics, administrative and legal documents, and the other similar things in different environments. So from now on, in this section, in order to emphasize the multipurpose usability, we use the term "InformationalObject" for all of the above. We named this scheme "QAVNS" that stands for Quadruple Adaptive Version Numbering Scheme.

In the proposed scheme, in normal mode, the numbering is carried out using four integers, separated by a dot (".") that respectively left to the right are named: "Release Sequence Number" (or RSN), "Generation Number", "Feature List Number" and "Correction List Number".

If none of the changes made to an object are in the form of adding and/or modifying the functionality or content, but all are corrections and fixings, instead of the feature list number, only the correction list number will increase by one. For example, if after editing 11.0.7.0 a number of changes are made to correct mistakes and fix some problems in the object, the new version would be 12.0.7.1 and not 12.0.8.0 (that indicates the changes are only in features and content). Similarly, if all subsequent changes are corrections and bug fixings, the correction list number will be updated again and the next version will be 13.0.7.2.

If the changes made to the object in relation to the previous version are combination of "adding and/or modifying the functionality or content" and "correcting and fixing bugs/drawbacks", then the feature list number

is incremented by one unit and the correction list number will be reset to "one". So for this case, we will arrive at 28.0.8.1 after the 27.0.7.15 version.

If the changes made to the object are extensive and fundamental, they will change the generation number. For example, from the 235.1.14.772 will reach to 236.2.0.0. It is important to note that if the generation number increases, the feature list number and the correction list number must be reset to zero.

The change in the "Release Sequence Number" is simply one unit increment in each version. This number is in fact a sequential serial number.

So briefly, if in QAVNS we want to show the structure of a version number as a pattern, we will have:
e0RSN.G.FF.CCC
where in:

- e0RSN represents the "Release Sequence Number" using the "extended zero leading" technic.
- G represents the "generation number" and will not be more than one digit in most work environments and applications.
- FF represents the "Features List Number" and will not be more than two digits in most work environments and applications.
- CCC represents the "Correction List Number" and will not exceed three digits in most workplaces and applications.

The QAVNS has the capability to use for tracking changes of any "informational object" in a variety of environments, without altering its structure. Using this scheme, we have presented a new approach for automating the decision making of choosing the appropriate virtual machine live migration method, which is presented in the next section.

## 5. PROPOSED APPROACH AND METHOD

So far, none of the researchers in the field of virtual machine live migration have claimed that their proposed method is optimal for all conditions. It seems that the optimal solution that can be relied upon in different conditions is the combination of the methods proposed so far. This means that depending on the different circumstances of the system, it is decided which of the proposed methods for the virtual machine live migration will be used.

To achieve this, we provide some criteria for automatically detecting the status of the virtual machine, and hence the automated selection of the virtual machine live migration method/algorithm.

Based on the concept of the "Informational Object" that is presented in the previous section, we look at virtual machine memory as an informational object that is constantly changing, and therefore can have a "VersionIdentifier" based on the QAVNS at any given time.

Assume that the *MST* constant represents the start time of the virtual machine live migration. This will get its value from the public field/variable in the operating system kernel, commonly known as "*jiffies*". The *jiffies* field, in the form of an incremental counter, maintains the number of machine hardware timer interrupts since start of system:

*jiffies* = count (Timer Interrupts) (1)

*MST* = *jiffies* at start of migration task (2)

The mapping of QAVNS fields in the virtual machine memory informational object is suggested as follows:
The *0RSN* field in the QAVNS is known as the "Release Sequence Number". Given the nature of the virtual machine's memory informational object, we consider this field as an order number that has the nature of time and its values are extracted from machine timer interrupts. This field represents the number of moments that have elapsed since the start of the migration operation. Therefore, the *0RSN* field of the QAVNS is mapped here as "Jiffies" with the *JS*acronym and is used with the following definition:

$$JS_i = \begin{pmatrix} 0 & if\ i=1 \\ count(TimerInterrupts) - MST & otherwise \end{pmatrix} \quad (3)$$

The *GG* field in the QAVNS is known as the "Generation Number". Given the nature of the virtual machine memory informational object, we consider this field to be the "Generation Number of the Virtual Machine Operating System". Therefore, the *GG* field of the QAVNS is mapped and used here as "OS Instance Number". By definition, this field is initialized to 1 at the moment the virtual machine live migration operation is started. Each time the guest's operating system is restarted during the virtual machinelive migration, its value increases by one unit:

$$OSIN_i = \begin{pmatrix} 1 & if\ (JS_i - MST) = 0 \\ OSIN_{i-1} + 1 & if\ VM\ OS_{i-1}\ is\ rebooted \\ OSIN_{i-1} & otherwise \end{pmatrix} \quad (4)$$

The *FF* field in the QAVNS is known as the "Feature List Number". Given the nature of the virtual machine's memory informational object, we consider this field to be "indicator of the volume of changes in processes of the virtual machine". Therefore, the *FF* field of the QAVNS is mapped and used here as a "Process Change Number" with the *PCN*acronym. By definition, this field is initialized to zero at the moment the operation of virtual machine live migration starts. With each creation or deletion of a process under the guest's operating system in the virtual machine, the value of itis increased by one unit. This field is reset to zero if the previous field (*OSIN*) is changed:

$$PCN_i = \begin{pmatrix} 0 & if\ (JS_i - MST) = 0 \\ 0 & if\ OSIN_i \neq OSIN_{i-1} \\ PCN_{i-1} + 1 & if\ any\ process\ change \\ PCN_{i-1} & otherwise \end{pmatrix} \quad (5)$$

The *CCC* field in the QAVNS is known as the "Correction List Number". Given the nature of the virtual machine's memory informational object, we consider this field to be "number of the modified memory pages". Therefore, the *CCC* field of the QAVNS is mapped and used here as "Dirty Page Count Number" with the *DPCN*acronym. By definition,this field is initialized to zero at the moment the operation of virtual machine live migration starts. With each increment that occurs in the "number of modified pages" of the virtual machine memory, its value is increased by one unit. This field is reset to zero if the previous field (*PCN*) is changed:

$$DPCN_i=\begin{cases}0 & if\ (JS_i-MST)=0\\ 0 & if\ PCN_i\neq PCN_{i-1}\\ DPCN_{i-1}+1 & if\ any\ change\ \text{in}\ \#\ of\ dirty\ pages\\ DPCN_{i-1} & otherwise\end{cases} \quad (6)$$

Therefore, the Version Identifier Scheme of the virtual machine memory as a QAVNS-based object will be:

$$JS.OSIN.PCN.DPCN \quad (7)$$

We add an initial step called the "Sampling Phase" to the three steps proposed by Clark and colleagues for the virtual machine live migration process (that is mentioned at the end of section 2 of this article). During this initial step, the values of the virtual machine memory version identifier are being updated and monitored. Determining the "Time length of the Sampling phase" (which we call it Ts) is still an open research topic.

## 6. ALGORITHM AND IMPLEMENTATION

Simulation and implementation can be used to provide evidence of the efficiency of the proposed method. To simulate, two general methods can be used: First, use of simulation tools that are appropriate for examining different methods of live migration of virtual machines. Second, utilizing the implementation platforms to create a specific simulation environment for this research. We believe that the simulation tools developed for cloud computing research can be used directly or with minor alterations for this research.

Bahwaireth et al. [17], in their research, have pointed to some of the most promising cloud computing simulators: CloudSim, CloudReports, CloudExp, iCanCloud, CloudAnalyst, and GreenCloud. Of course, in our opinion some other items, such as TeachCloud and GroudSim, can also be added to the list.

In Figure 1, the parts of CloudSim as discussed in the Bahwaireth et al.'s article is illustrated.

Many other tools (including CloudReports) have been developed based on CloudSim. Finally, Bahwaireth and colleagues in their article introduced iCanCloud as the top choice [17].

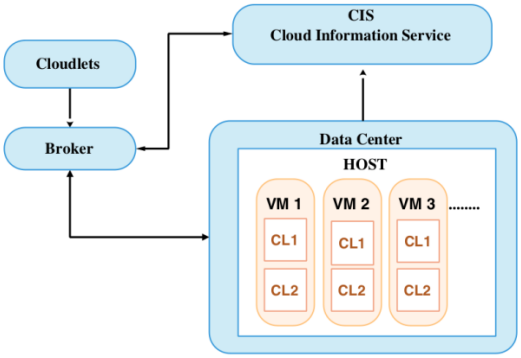


**Figure 1.** CloudSim parts and their relationships [17]

Tian et al. [18] have compared the four open source simulation tools CloudSim, GreenCloud, iCanCloud and CloudSched. Ultimately, the researchers do not list any of them as the best simulation tool among the four software and recommend using a combination of them (depending on the optimization goals).

Additionally, Hirofuchi et al. [8] claimed that the SimGrid toolkit as a general simulation tool for distributed systems research, can be extended to be used for simulation of virtual machines live migration. According to Hirofuchi et al. [8], the simulation results of a live migration using their proposedextension on SimGrid are near to the results of the migration in the real world. Their live migration tests in real world are performed on the Grid5000, a large processing platform and a network for IT research in France. Figure 2 illustrates the Grid5000 geographic structure.

Li and his colleagues, have implemented a toolbased on CloudSim called DartCSim+that have ability to overcome the three CloudSim limitations using the changes they have made [19]. One of these limitations is not considering network overhead on the process of virtual machines live migration. In Figure 3, the flow of data in such a migration, based on the results that are reported by Li and colleagues [19] is depicted.

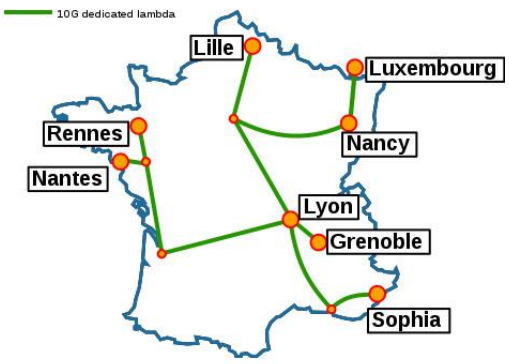


**Figure 2.** Geographic structure of the Grid5000 physical platform in France [8]

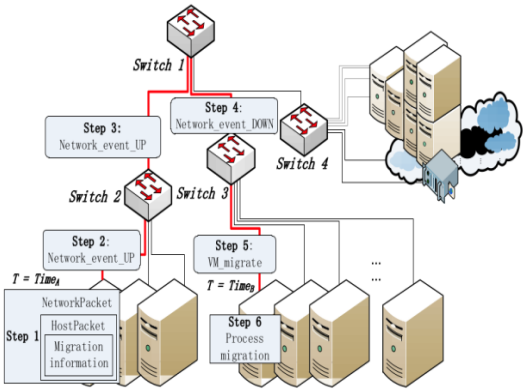


**Figure 3.** Data flow in virtual machine migration including network overhead [19]

The initial results from our studies are not consistent with the results of the research presented by Bahwaireth et al. [17] (i.e.iCanCloud), and so we are not in a position to validate their results. The preliminary results of our studies indicate that CloudSim is superior to the other options in terms of capabilities and can be a good choice for simulating the various VM live migration methods. In addition to the results of Li et al. [19], the results of the research by Stamenov and his colleagues also support the initial results of our studies [20].

We represent the virtual machine memory version identifier at the start of the sampling phase with the index s, and at the end of the sampling phase with the index e. So we will have:
Version ID at the beginning of the sampling phase:

$$JS_s.OSIN_s.PCN_s.DPCN_s \tag{8}$$

Version ID at the beginning of the sampling phase:

$$JS_e.OSIN_e.PCN_e.DPCN_e \tag{9}$$

We claim that the following algorithm can automatically detect the different operating conditions of a virtual machine and choose the appropriate approach for live migration properly. The pseudocode of the proposed algorithm is as follows:

if ($OSIN_e > OSIN_s$) then choose “PostCopy Method”

else if ($\frac{JS_e}{PCN_e - PCN_s} < T_p$) then choose “PostCopy Method” (10)

else if ($\frac{JS_e}{DPCN_e - DPCN_s} < T_d$) then choose “PostCopy Method”

else choose “PreCopy Method"

This is a deterministic algorithm that distinguishes the status of virtual machine, that is the volume of the VM workload plus its nature. Both of them are important, because a light workload, carelessly leads to choose pre-copy method. At the other hand, if the situation is about a heavy memory write-intensive workload, then our algorithm leads the hypervisor to select post-copy method.

To achieve the above mentioned assessment, the proposed algorithm uses two threshold values: $T_p$ and $T_d$. Determining appropriate values of the $T_p$ and $T_d$ threshold constants in the above algorithm is still an open research topic.

## 7. ANALYSIS

Based on researches, it has been prooved that each of the pre-copy and post-copy approaches is more appropriate than the other in special situations. In short, the use of the pre-copy method provides better performance when the processes running on the virtual machine have more "read-intensive" behaviors. Conversely, the use of the post-copy method provides better efficiency when processes running on a virtual machine are more likely to have "write-intensive" behavior [12].

As during the sampling phase, by increasing the value of the *OSIN* field, it is determined that the virtual machine operating system is being restarted, this analysis can be deduced that the changes to the memory pages are repetitive and massive. This volume of change is "level one" (meaning the highest level of changes in memory pages), and yet it is no longer necessary to examine other virtual machine states for choosing the appropriate method. We call this state the "Unstable state of the operating system of the virtual machine" and therefore the appropriate choice in this case is the post-copy method.

If there is no change in the *OSIN* field, then in the next step, it is possible to decide based on the rate of changes in the process set which is managed by the operating system in a time unit (based on changes in the *PCN* field). A logical inference here is that, typically, by creating or deleting any process in a virtual machine, the volume of changes in memory pages is more than the situation that the same process is in its normal operation.

If the change rate of the set of processes managed by the VM operating system, passes through a threshold (which we show it in the previous section with a $T_p$ constant), this volume of changes is "level two" (meaning the mediocre state of the level of changes in the memory pages). In this situation, however, it is no longer necessary to examine other virtual machine status to choose the appropriate method. We call this a "Storm of birth and death of processes in a virtual machine" and therefore the appropriate choice in this case will be the post-copy method.

If none of the above two cases occur, then the decision is based on the third criterion, which is the rate of the number of modified memory pages per unit time

(based on changes in the *DPCN* field). If the rate of increase in the number of modified memory pages passes through a threshold (which is shown in the previous section with a $T_d$ constant), this volume of changes is "third level" (meaning the lowest level of changes in memory pages). This is a simple, yet quite reasonable deduction, in which case it would be a good choice for the post-copy method. The occurrence of this state indicates that while the virtual machine's operating system is in a steady state and also does not see the storm of birth and death of processes, but we are faced with a number of "write-intensive" processes that are constantly changing the memory pages. We call this an "Unruliness of processes" and therefore the appropriate choice in this case will be the post-copy method, too.

Obviously, if none of the above three cases are true for a virtual machine, the amount of changes in memory pages is low. We call this a "Calm virtual machine with noble processes" and so the proper choice in this case will be the pre-copy method.

## 8. CONCLUSIONS AND FUTURE WORKS

In this paper, in addition to a brief presentation of an adaptive version numbering scheme called QAVNS, we present a new approach to automating the process of choosing a suitable strategy for virtual machine live migration. The proposed approach can enable systems based on virtual machine technology and cloud computing to automatically make decision for selecting the best strategy and migration algorithm in terms of workload conditions. This leads to a relative optimum performance in the current situation.

One of the most significant concerns in this research is whether the processing overhead of the proposed method would outweigh its benefits or not. In order to give a convincing answer, it is required to conduct a detailed simulation or real implementations to find the optimal values for the three mentioned parameters: $T_s$, $T_p$, and $T_d$. These investigations can be accomplished as an extension to our research work.

## 9. REFERENCES


1. Singh, H., Kumar, S. and Shukla, A., "Load balancing approaches for web servers: A survey of recent trends", ***International Journal of Engineering***, Vol. 31, No. 2, (2018), 263-269.
2. Rezai, H. and Speily, O., "Energy aware resource management of cloud data centers", ***International Journal of Engineering-Transactions B: Applications***, Vol. 30, No. 11, (2017), 1730-1739.
3. Ahmad, R.W., Gani, A., Hamid, S.H.A., Shiraz, M., Xia, F. and Madani, S.A., "Virtual machine migration in cloud data centers: A review, taxonomy, and open research issues", ***The Journal of Supercomputing***, Vol. 71, No. 7, (2015), 2473-2515.
4. Pop, C.B., Anghel, I., Cioara, T., Salomie, I. and Vartic, I., "A swarm-inspired data center consolidation methodology", in Proceedings of the 2nd International Conference on Web Intelligence, Mining and Semantics, ACM., (2012), 41-48.
5. Beloglazov, A. and Buyya, R., "Energy efficient resource management in virtualized cloud data centers", in Proceedings of the 2010 10th IEEE/ACM international conference on cluster, cloud and grid computing, IEEE Computer Society., (2010), 826-831.
6. Zhou, M., Zhang, R., Zeng, D. and Qian, W., "Services in the cloud computing era: A survey", in Universal Communication Symposium (IUCS), 2010 4th International, IEEE., (2010), 40-46.
7. Mishra, R. and Jaiswal, A., "Ant colony optimization: A solution of load balancing in cloud", ***International Journal of Web & Semantic Technology***, Vol. 3, No. 2, (2012), 33-50.
8. Hirofuchi, T., Lèbre, A. and Pouilloux, L., "Adding a live migration model into simgrid: One more step toward the simulation of infrastructure-as-a-service concerns", in Cloud Computing Technology and Science (CloudCom), 2013 IEEE 5th International Conference on, IEEE. Vol. 1, (2013), 96-103.
9. Clark, C., Fraser, K., Hand, S., Hansen, J.G., Jul, E., Limpach, C., Pratt, I. and Warfield, A., "Live migration of virtual machines", in Proceedings of the 2nd Conference on Symposium on Networked Systems Design & Implementation-Volume 2, USENIX Association., (2005), 273-286.
10. Choudhary, A., Govil, M.C., Singh, G., Awasthi, L.K., Pilli, E.S. and Kapil, D., "A critical survey of live virtual machine migration techniques", ***Journal of Cloud Computing***, Vol. 6, No. 1, (2017), 23-36.
11. Zayas, E., "Attacking the process migration bottleneck", ***ACM SIGOPS Operating Systems Review***, Vol. 21, No. 5, (1987), 13-24.
12. Hines, M.R., Deshpande, U. and Gopalan, K., "Post-copy live migration of virtual machines", ***ACM SIGOPS Operating Systems Review***, Vol. 43, No. 3, (2009), 14-26.
13. Sahni, S. and Varma, V., "A hybrid approach to live migration of virtual machines", in Cloud Computing in Emerging Markets (CCEM), IEEE International Conference on, (2012), 1-5.
14. Yang, Y., Mao, B., Jiang, H., Yang, Y., Luo, H. and Wu, S., "Snapmig: Accelerating vm live storage migration by leveraging the existing vm snapshots in the cloud", ***IEEE Transactions on Parallel and Distributed Systems***, Vol. 29, No. 6, (2018), 1416-1427.
15. Bloch, T., Sridaran, R. and Prashanth, C., Understanding live migration techniques intended for resource interference minimization in virtualized cloud environment, in Big data analytics. 2018, Springer.487-497.
16. Kherbache, V., Madelaine, E. and Hermenier, F., "Scheduling live migration of virtual machines", ***IEEE Transactions on Cloud Computing***, Vol., No. 1, (2017), 1-14.
17. Bahwaireth, K., Benkhelifa, E., Jararweh, Y. and Tawalbeh, M.A., "Experimental comparison of simulation tools for efficient cloud and mobile cloud computing applications", ***EURASIP Journal on Information Security***, Vol. 2016, No. 1, (2016), 15-23.
18. Tian, W., Xu, M., Chen, A., Li, G., Wang, X. and Chen, Y., "Open-source simulators for cloud computing: Comparative study and challenging issues", ***Simulation Modelling Practice and Theory***, Vol. 58, (2015), 239-254.
19. Li, X., Jiang, X., Ye, K. and Huang, P., "Dartcsim+: Enhanced cloudsim with the power and network models integrated", in Cloud Computing (CLOUD), IEEE Sixth International Conference on, IEEE., (2013), 644-651.
20. Stamenov, D. and Kostoska, M., "Virtual machine migration in cloud--techniques, challenges and cloudsim migration simulation", (2017), 103-109.

# A Versioning Approach to VM Live Migration

M. Tajamolian, M. Ghasemzadeh

*Computer Engineering Department, Yazd University, Yazd, Iran*

*P A P E R I N F O*

*Paper history:*
Received 10 July2018
Received in revised form 27 Septemebr 2018
Accepted 26 October 2018

*Keywords*:
*Virtual Machine Live Migration*
*Pre-Copy & Post-Copy*
*QAVNS*
*Quadruple Adaptive Version Numbering Scheme*
*Informational Object*
*Cloud Computing*

**چکیده**

در حوزه مهاجرت زنده ماشین‌های مجازی، دو استراتژی به نام‌های پیش‌کپی و پس‌کپی ارائه شده است؛ اما هر یک از این استراتژی‌ها فقط در برخی شرایط به خوبی عمل می‌کند. در این مقاله، پس از مروری مختصر بر «طرح چهارگانه شماره‌گذاری تطابق پذیر نگارش‌ها» ، یک رویکرد جدید ارائه می‌کنیم که مبتنی بر مفهوم «شیء اطلاعاتی»، اختصاص شماره ویرایش، و پایش تغییرات آن می‌باشد. در این روش، مجموع حافظه ماشین مجازی به‌عنوان یک شیء اطلاعاتی که پیوسته در حال تغییر است، در نظر گرفته می‌شود. در این راستا، معیارهای لازم را معرفی و الگوریتمی را ارائه می‌دهیم که توسط آن آبرناظر می‌تواند رفتار فعلی ماشین مجازی را شناسایی کند و به‌طور خودکار راهبرد مهاجرت ماشین مجازی را از دو گزینه پیش‌کپی و پس‌کپی انتخاب نماید. در این رابطه، بسترهای پیاده‌سازی و شبیه‌سازی را با توجه به آخرین فناوری‌های موجود، بخصوص CloudSim, SimGrid و DartCSim+ ، نیز ارزیابی نموده‌ایم. تحلیل‌های رسمی نشان می‌دهد که استفاده از طرح و الگوریتم پیشنهادی می‌تواند روند مهاجرت زنده ماشین‌های مجازی را به‌طور قابل توجهی بهبود بخشد.

***doi**: 10.5829/ije.2018.31.11b.06*