Journal of Theoretical and Applied Information Technology
31st October 2016. Vol.92. No.2
© 2005 - 2016 JATIT & LLS. All rights reserved·
ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

PROVIDING A SOLUTION TO IMPROVE PRE-COPY METHOD FOR MIGRATING VIRTUAL MACHINES IN CLOUD INFRASTRUCTURE

[1]PARVIN AHMADI DOVAL AMIRI, [2]SHAYAN ZAMANI RAD,
[3]FARAMARZ SAFI ISFAHANI
[1]Islamic Azad University of Babol, Computer Engineering Department, Babol, Iran
[2]Mazandaran University of Science and Technology, Computer Engineering and IT Department Babol, Iran
[1]Islamic Azad University of Najaf Abad, Computer Engineering Department, Isfahan, Iran
E-mail: [1]p.ahmadi@khuisf.ac.ir, [2]sh.zamani@ustmb.ac.ir
[3]fsafi@iaun.ac.ir

ABSTRACT

Cloud computing can be defined as a new computing model which suggests solutions for providing information technology services analogous to utility services such as power electricity, telephone. Thanks to the virtualization technology, most of Cloud datacenters use this technology for performance improvement. This technology transforms a physical server to several virtual machines. With regard to this property, virtual machines can be transferred from one place to another place that is called migration. Pre-Copy is one of the main methods in migration techniques. An important problem in this method is that when the memory pages are changed (dirty pages) faster than sending pages to a destination, the process of migration will take a long time that is the case in transferring virtual machines. In this paper, a solution is proposed that sets up the speed of virtual machines in terms of CPU frequency and makes a balance between senders and receivers in pre-copy algorithm. The results show that the proposed method has better performance compared to the pre-copy method and decreases the total migration time around 25% and the total amount of transferred data around 47%.

Keywords: *Cloud Computing, Migration, Virtualization, Virtual Machine*

1. INTRODUCTION

Cloud computing suggests solutions for providing information technology services analogous to utility services such as water, power electricity, gas, and telephone. One of the most important constitutive elements of this computational model is virtualization technology [1][3].

Virtualization makes datacenters flexible and powerful and also it enables them to utilize their physical resources in an efficient way. Furthermore, hypervisor is an integral part of virtualization technology. It emulates physical resources for virtual machines and plays as an intermediate layer between resources and virtual machines, which are running on the physical server. By doing so, virtual machines can use emulated and isolated resources. As each of these virtual machines may provide a database server, mail server, or even a web server, most of the giant cloud pioneers and providers are using virtualization technology to improve their efficiency and capacity.

In other words, virtualization technology converts a physical machine into a logical file, so, it could be easy for that machine called virtual machine, to move from one place to another one, and this is known as virtual machine migration technique.

Nowadays, datacenters have to handle a large amount of requests and workloads, and migration technique helps them to cope with this issue. Therefore, virtual machine migration is an undeniable and remarkable technique for delivering quality of service (QoS). In fact, migration techniques are used to achieve a wide variety of goals such as load balancing, fault tolerance, power management, response time reduction, server enhancement, etc.

Pre-copy is known as the most common migrating method, which has been used as a live technique by most of the hypervisors. Although this method has too many advantages and benefits, substantial amount of workloads that change memory pages in a very high rate may downgrade the performance of this method. In this case, pre-copy sends memory pages to the destination in each round but without any strictness. It means that because of the high rate of changes, memory pages are modified in each round and the hypervisor resend them again, an endless job. To state it more clearly, when the change rate of a bunch of memory pages is higher than the network bandwidth, iterative copy phase will completely downgrade the whole system. However, this phase continues for 30 rounds. Then, the hypervisor turns virtual machine off and migrates it by non-live method. As a result, all the TCP connections will be disconnected and all running services will stop, so, that virtual machine will be inaccessible for a short period of time (*downtime period*). It comes to a major issue, when that virtual machine has many users or customers and this disruption bring financial losses for that cloud service provider. It means that not only the provider may lose the trust of customers, but also has to reimburse to them.

This paper suggests that there exist a direct link between decreasing the virtual machine's CPU frequency and amount of sending pages in network. Therefore, the hypothesis of this research is that to reduce the CPU frequency until the network bandwidth become greater than the change rate of pages. By using our method, memory pages are transferred to the destination hypervisor in a short time. As a result, the amount of data sending and total migration time are decreased significantly.

The test results show that the proposed method decreases the total migration time to around 25% and transferred data to around 47% in comparison to the basic pre-copy approach.

Here is an outline of the covered topics: Section 2 gives and overview of the related works, Section 3 introduces migration techniques, and pre-copy method. Section 4, proposes the alternative and new method. Section 5, describes the experiments and justifies the results of evaluation.

2. RELATED WORK

Figure1 shows taxonomy of migration methods. The basics were proposed in [13], where the authors presented pre-copy for the first time. Live migration method based on adaptive memory compression is presented in [11]. In this method, detection of size of modified pages happens automatically and then a maximum and minimum threshold of pages *similarity ratio* is set by the method. In this case, compression time will be decreased, and consequently the amount of modified pages will be reduced significantly.

In [14], a live migration method based on improving pre-copy presented. This method adds an extra bitmap for marking modified pages.

The results show that this method decreases the total amount of send data to 34% and total migration time to 35%, and also repetition phase (stop and copy) is completed 5 times faster than normal pre-copy.

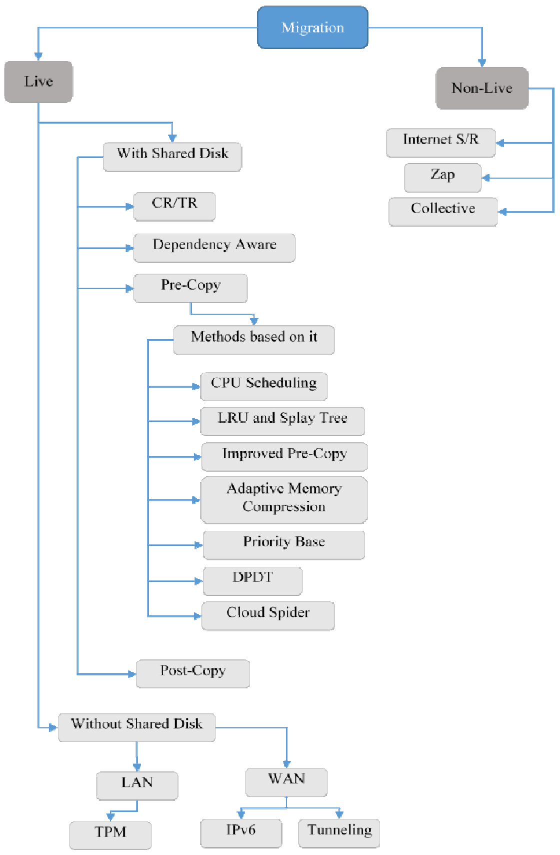

Figure 1: Taxonomy of Migration Methods

Cloud Spider is another method, which presented in [15] for migrating virtual machines in WAN networks. In [12], another new live migration method is proposed which combines the LRU scheduling algorithm with the Splay tree algorithm. The result of experiments show that the method can reduce the amount of send data and total migration time up to 23.67% and 11.45%, respectively. In [18], a new method called DPDT is presented. In this method, modified pages are transmitted by a delay, each of delays are stored in an additional bitmap called *Delay*. In *Delay* bitmap, those pages that are modified in both previous and current round will send later.

Another live migration method based on priority was presented in [16]. In this method, applications, which are sensitive and intolerant to delay, and fraction have more priority than the others. During stop and copy phase, like pre-copy method, the method only sends the high priority modified pages, those are belongs to high priority applications. The results show that this method decreases the disruption of services to 57% compared to the pre-copy.

3. MIGRATION TECHNIQUES

Migration techniques are divided into two general categories. Each migration method is a subcategory of one of these two types: 1) non-live migration, 2) live migration.

In non-live migration a virtual machine completely stops at source host, and then its CPU state, memory pages, and disk data will be transferred to the destination host. Then, destination host starts the VM from the last saved state of it before the transfer.

The most deleterious outcome of this method is the long downtime of the virtual machine. Nevertheless, it has some advantages such as lack of incompatibility due to data transfer, easy implementation and sending memory pages all at once.

In contrast, in the live migration method, while a virtual machine is powering on and responding to users' requests, it starts to move from source to destination host. The usefulness of this technique is in providing services even when the migration process has been started and memory pages and CPU states of VM have been transferring at the same time.

3.1. Pre-Copy Method

In order to transfer virtual machines, the pre-copy method has been used by a majority of hypervisors. Generally, it consists of three phases. However, some extra phases and extensions are added into it. In the first phase, source host sends a virtual machine's information such as memory pages and CPU states to the destination host, iteratively. Furthermore, during this operation, the source host records all changed memory pages in each round of copying.

After passing several rounds, the second phase is started, immediately. In this step, virtual machine is suspended at the source host and then it is started to move from the source to the destination. However, the other pages are changed consistently during the first phase. Obviously, it is impossible to transfer them completely due to the continuous changes. CPU states, which are an integral part of launching or restoring virtual machine on the destination host, are also moved. Finally, in the third phase, virtual machine starts at the destination host from its last state before suspension.

It is important to mention that less rate of changes in the first phase leads to less downtime, which occurs in the second phase.

4. PROPOSED METHOD

The main problem of the pre-copy method is the lack of appropriate reaction to cope with a large amount of workloads that modify memory pages in a very high rate. In other words, pre-copy is unable to manage a mass of changes that may happen during virtual machine migration process. In such cases, what pre-copy can do is to send memory pages to the destination in each round of migration process. That is to say, pages will be modified in next round and the method will have to resend them again and again iteratively.

It is obvious that when changing rate of memory pages is higher than network bandwidth, the above-mentioned iterations will be absolutely unusable. The iterative copy phase continues up to 30 rounds [12]. After that, the hypervisor turns off virtual machine and transfers it in a non-live way. By doing so, all the TCP connections will be disconnected and all running services will stop and as a consequence, costumers cannot access to the virtual machine's services (e.g. Email, Website, FTP) for a period of time, that is not a good news for them.

4.1. Reducing vCPU Frequency

The main idea of our proposed method is that to adopt CPU speed (or CPU Frequency) during migration time. In other words, we reduce the speed of CPU gradually, and as a result the changing rate of memory pages will be reduced ultimately. In order to implement this idea, we write an algorithm that decreases the frequency of virtual machine's CPU (vCPU) at the time of moving, just for a short period of time.

This declining in the CPU speed will reduce the performance of selected virtual machine, therefore, a reduction on writing operations and modifying pages will happen consequently. The usefulness of this solution emerges in such cases that we have write-intensive workloads on the most of VMs. It should be emphasized that this policy, vCPU frequency reduction, will apply until the migration process finishes.

In the proposed algorithm, the vCPU frequency is decreased until the rate of sending pages through the network (network bandwidth) becomes higher than memory page change rate. After a number of steps, migration process will come to the second phase, stop and copy, and virtual machine becomes suspended. In this stage, if the pre-copy method uses, the migration process takes long time and too many rounds, close to 30. Furthermore, in the worst case, live migration will turn into the non-live, because of the high changing rate.

Pseudo code of the proposed algorithm is presented as follows:

```
1: MM=VM's Memory(Output)
2: R=1
3: T=Threshold(Input)
4: C=Round Cap(default 30)
5: B=Network Bandwidth(Input)
6: CP=VM's vCPU frequency(Input)
7: MPS=MM's size
8: While MPS>T or R<C
9:     Transfer MM to destination host
10:    Mark dirtied pages to bitmap
11:    MM=bitmap index
12:    MPS=MM's size
13:    While I/O rate >= B
14:            TempCP=Reduce CP
15:            vCPU=TempCP
16:    End while
17:    R=R+1
18: End while
19: Pause VM running on source host
20: Transfer CPU state
21: Transfer remain MMs to destination host
22: vCPU=CP
23: Resume VM on destination host
```

According to the 12th line, page memory changing rate and network bandwidth are compared at first step. If changing rate is higher, the migration process will not finish. So, the algorithm goes to 13th line and records the status of vCPU. Then in line 14, it reduces the vCPU frequency to half. In the next line, it checks whether the reduction in the frequency reduces memory page changing rate or not. If nothing happens, it will reduce the frequency to half again. This loop continues until the memory pages changing rate becomes lower than network bandwidth. Finally, after transferring virtual machine to the destination, the main frequency will be adjusted again and all the states will roll back on the machine. In Figure 2, flowchart of the proposed method is shown.

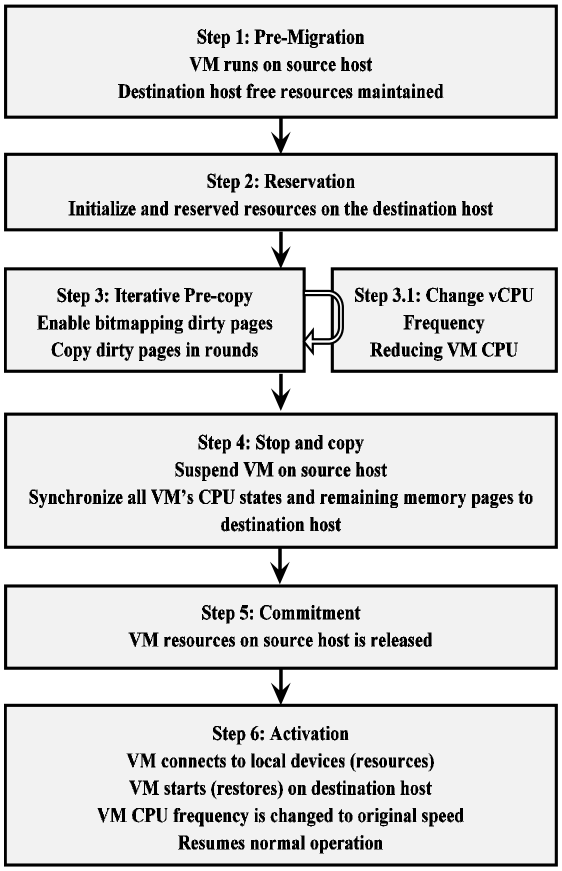
Step 1: Pre-Migration
VM runs on source host
Destination host free resources maintained
Step 2: Reservation
Initialize and reserved resources on the destination host
Step 3: Iterative Pre-copy
Enable bitmapping dirty pages
Copy dirty pages in rounds
Step 3.1: Change vCPU Frequency
Reducing VM CPU
Step 4: Stop and copy
Suspend VM on source host
Synchronize all VM's CPU states and remaining memory pages to destination host
Step 5: Commitment
VM resources on source host is released
Step 6: Activation
VM connects to local devices (resources)
VM starts (restores) on destination host
VM CPU frequency is changed to original speed
Resumes normal operation

Figure 2: Flowchart of Proposed Method

5. CASE STUDY

In order to start the migration process, the administrator should run *migrate* command from *virsh,* the main interface for managing guest domains. This command is shown in the following:

```
virsh migrate --live domain_id destination_node
```

Through running the command, at first, the source hypervisor receives migration command (Figure 3) and finds the VM's information. Then, it tries to make a connection to the destination host and its hypervisor. After establishing the connection, the destination host sends the *ready* command, and then the migration process starts.

The source hypervisor starts to send memory pages iteratively (Figure 4). After some rounds, depending on the memory pages changing rate and network bandwidth, migration stub realizes that sending pages is not useful, because changing rate is much higher.

Accordingly, remote hypervisor asks the source hypervisor to decrease the VM's vCPU frequency until the network bandwidth is lower than the changing rate (Figure 5).

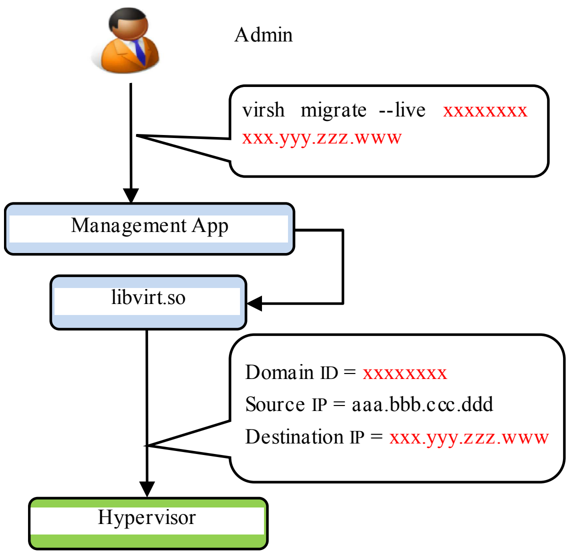
Admin
virsh migrate --live xxxxxxxx xxx.yyy.zzz.www
Management App
libvirt.so
Domain ID = xxxxxxxx
Source IP = aaa.bbb.ccc.ddd
Destination IP = xxx.yyy.zzz.www
Hypervisor

Figure 3: Admin and Hypervisor Interaction

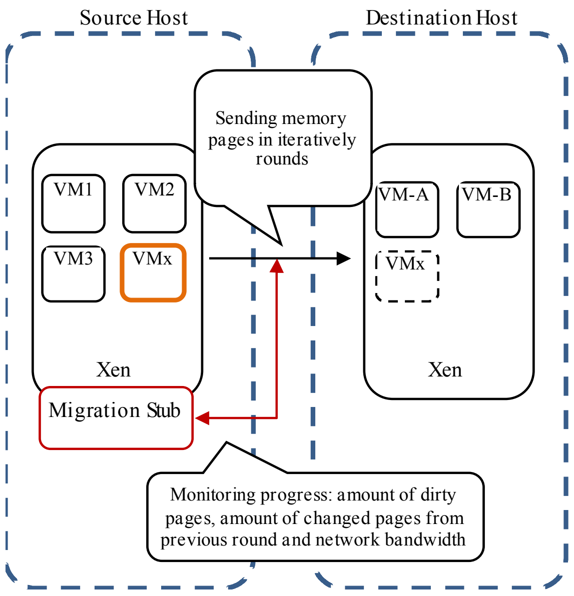
Source Host
Destination Host
Sending memory pages in iteratively rounds
VM1
VM2
VM3
VMx
Xen
Migration Stub
VM-A
VM-B
VMx
Xen
Monitoring progress: amount of dirty pages, amount of changed pages from previous round and network bandwidth

Figure 4: The First Step of Migration Process

After reducing the frequency, source hypervisor continues to send memory pages and at the same time migration stub monitors the migration progress. Again, if the frequency reduction is insufficient, migration stub will ask the source hypervisor to reduce the frequency again. Then, the migration process enters to the second phase, and VM becomes suspended (Figure 6). Finally, after taking few seconds, VM resumes and runs on the destination host.

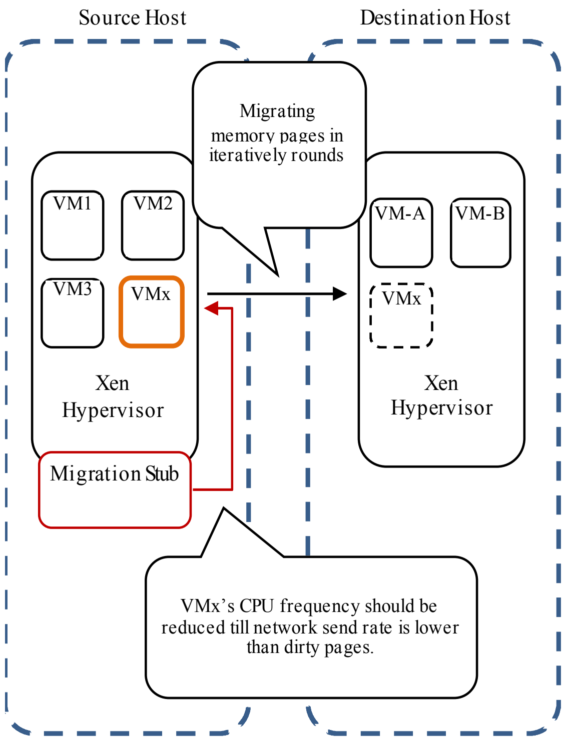
Source Host
Destination Host
Migrating memory pages in iteratively rounds
VM1
VM2
VM3
VMx
Xen Hypervisor
Migration Stub
VM-A
VM-B
VMx
Xen Hypervisor
VMx's CPU frequency should be reduced till network send rate is lower than dirty pages.

Figure 5: vCPU Frequency Reduction During Migration

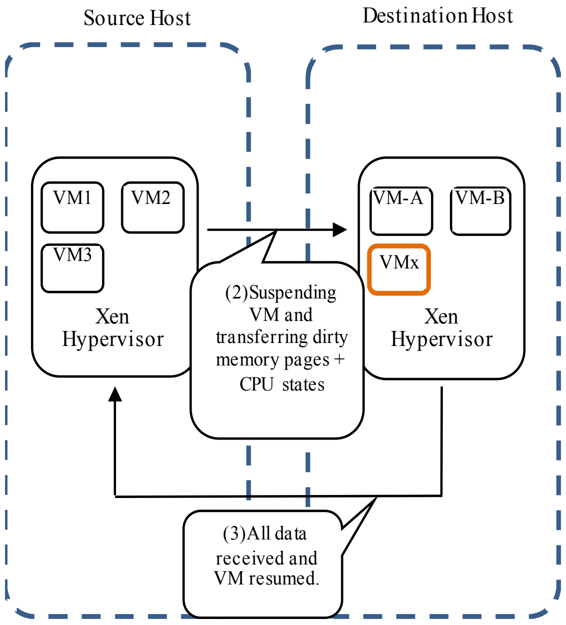
Source Host
Destination Host
VM1
VM2
VM3
Xen Hypervisor
(2)Suspending VM and transferring dirty memory pages + CPU states
VM-A
VM-B
VMx
Xen Hypervisor
(3)All data received and VM resumed.

Figure 6: The Final Phase of Migrating VM

6. IMPLEMENTATION AND EVALUATION

In order to implement a test environment, two machines with AMD Quadro Core 800 MHz processors, disk capacity of 500 GB, and 8GB memory as compute nodes were used. In addition, one another physical machine with Intel Core2Duo 2.66GHz CPU, 320 GB disk capacity and 4 GB of memory was used as a shared node.

These three machines were connected through a LAN network with 100 Mbps bandwidth. The migration operation was performed on a virtual machine with one vCPU, 2GB disk space, and 128 MB memory.

The operating system of two compute nodes was Linux Centos 5.6 and also for virtualization we used Xen 3.4 as hypervisor. Moreover, in order to create a virtual SAN storage, the OpenFiler tool were used on the shared node. Finally, the Stress[22] benchmark version 1.0.2 was used to make changes on memory pages. In addition, a small program was written by C language to simulate a real situation. The evaluation environment is shown in Figure7.

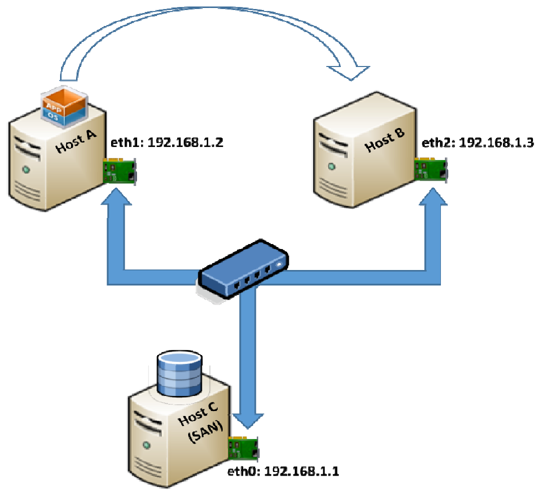

Figure 7: Test Environment

6.1. Test One: Evaluating Response Time per Requests

This test is a simulation of a real-world condition that a virtual machine processes a huge amount of write-intensive workloads. The result of this test is shown in figure 8. The proposed method is analyzed in different conditions to get the optimum reduction in frequency.

At first, the algorithm applied 25% decrease in frequency. This amount of reduction in the functionality of VM led to a decrease in the change rate of memory pages. Therefore, the total migration time decreased because the amount of candidate data for sending was reduced significantly.

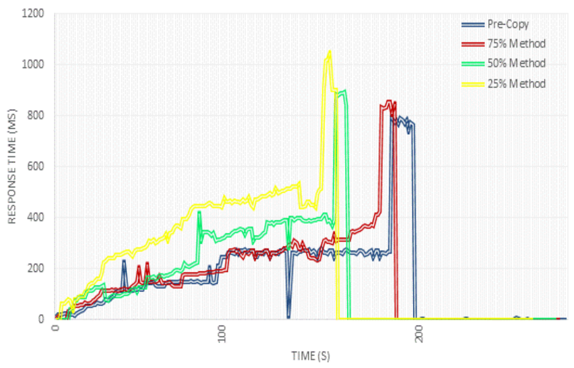

Figure 8: Response Time

In the second try, we reduced the VM's performance to 50%. As could be seen in Figure 8, the total migration time consequently was reduced. However, because of virtual machine suspension the response time rocked up to around 900ms from 140 to 160 seconds of migration process. Furthermore, Table1 shows the response time during migration process in more details.

Take a real-world scenario as an example. If a virtual machine migrates from one host to another one 20 seconds earlier, it will be acceptable only if it continues to respond to the requests with 200ms response time, because no only it occupies network bandwidth less but also users will not be aware of the delay caused by VM replacement. Nevertheless, as far as pre-copy method is concerned, virtual machine migration process takes a long time because total amount of transferred data through the iterative copy phase are huge.

We also tested our method to 75% reduction, it means virtual machine worked with only 25% of its performance capacity. As expected, the total migration time and amount of sent data declined noticeably, however, the response time experienced a disappointing incline. So, this inclining is too high and we can't consider this amount of reduction as a good choice. In other words, if a virtual machine starts to migrate with such a high response time, this movement will absolutely be notable by the users.

Table 1 Response Time During Migration Process (Migration Start Time, # Downtime, ^ Migration End Time)*

Method / (MS) Time(S)	Pre-Copy	25% Decrease	50% Decrease	75% Decrease
20	59	151	125	80
40	120	277	109	148
60	150	* 370	* 194	130
80	213	447	316	191
100	* 244	440	318	192
120	260	483	340	* 266
140	272	# 1015	388	270
160	275	* 0.64	# 841	315
180	266	0.44	^ 0.54	# 853
190	# 769	* 0.74	0.62	^ 0.63
200	^ 1.03	0.70	0.77	0.59

6.2. Test Two: Evaluating Network Throughput

In this test, the virtual machine migrates while the network throughput is being monitored. In order to implement this test, we used Netperf 2.5 benchmark. The result of this test is shown in figure

As we decrease the frequency of vCPU, both total migration time and suspension phase of virtual machine became shorter and shorter; furthermore, the amount of bandwidth consumption is less than what is happening in pre-copy.

As mentioned in the first test, this reduction in frequency makes the VM less functional. The situation of 50% is better than the situation of 25% from the perspective of response time. We insist on this factor because it is completely in a relation with users' happiness. So, it is one of the criteria that involved with quality of service (QoS).

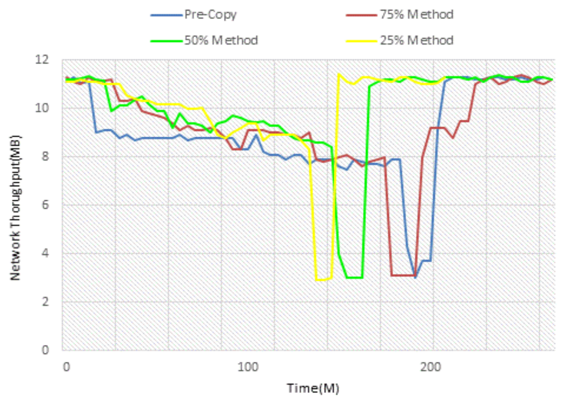

Figure 9: Network Throughput

6.3. Test Three: Evaluating Amount of Data Send

The aim of this test is comparing the proposed method with Pre-Copy from the perspective of transferred data. This test shows that by decreasing the frequency of vCPU, the amount of data, which is sent, reduces. Hence, the migration process will be done in a shorter time and also less network bandwidth will be occupied. The result of this test is shown in figure 10. We used Stress-1.0.2 benchmark to make some minor changes in memory pages. In other words, in this test some parts of memory pages have been changed while the amount of transferred data is being monitored.

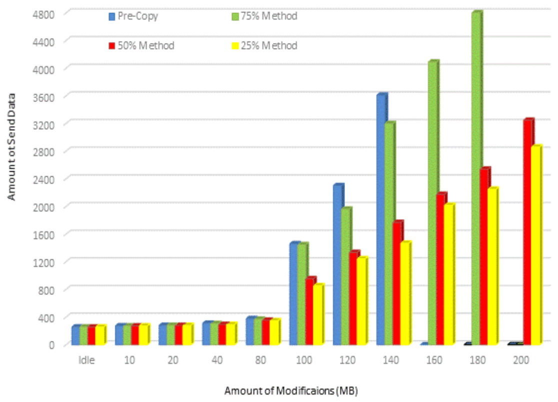

Figure 10: Amount of Transferred Data

As we mentioned before, one of the disadvantages of pre-copy method is to send memory pages several times. We concluded that in our method, by decreasing the frequency of vCPU, migration process especially the first phase, iterative copy becomes shorter. In fact, the main reason is that amount of transferred data will experience a reduction. Table2 shows the amount of transferred data during migration process in more details.

Table 1 Amount of Transferred Data (M1 Method (MB), M2 Modification(MB))

M1 ➔ / M2	Pre-Copy	25% Decrease	50% Decrease	75% Decrease
Idle	257	257	257	257
10	274	273	273	273
20	281	280.72	281.05	281.16
40	310.8	294	295.12	308.03
80	379.47	349	355.5	370
100	1460	855	956.31	1447.59
120	2301.84	1244	1335.22	1962.4
140	3610	1470.33	1770.14	3199.55
160	Stop Operation	2116.9	2176.01	4090.72
180	Stop Operation	2250	2540.34	4802.13
200	Stop Operation	2861.07	3248	Stop Operation

6.4. Test Four: Number of Rounds

The aim of this test is to show the number of rounds that is needed for migrating virtual machine memory pages by using proposed method in comparison with standard pre-copy method. The third test showed that by increasing the frequency, the amount of transferred data would also be increased. In the fourth test, we intend to show that increasing the vCPU frequency is also affect the number of rounds for sending memory pages. It should be emphasized that as the number of rounds increases, the total migration time also faces an increment. The results of this test are shown in Figure 11.

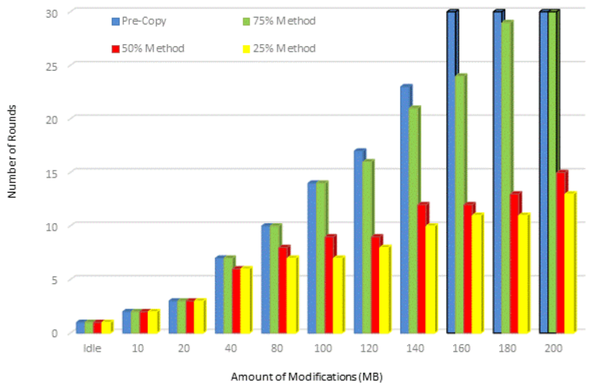

Figure 11: Number of Rounds for Sending Memory Pages

Furthermore, Table 3 shows more information about the number of needed rounds for sending virtual machine memory pages.

Table 3 Number of Rounds for Sending Memory Pages (M1 Method (Rounds), M2 Modification(MB))

M1 ➔ / M2	Pre-Copy	25% Decrease	50% Decrease	75% Decrease
Idle	1	1	1	1
10	2	2	2	2
20	3	3	3	3
40	7	6	6	7
80	10	7	8	10
100	14	7	9	14
120	17	8	9	16
140	23	10	12	21
160	Failed	11	12	24
180	Failed	11	13	29
200	Failed	13	15	Failed

7. CONCLUSION

In this paper, we described how organizations and costumers benefit from the advantages of Cloud Computing. Being pioneer and famous is very important for Cloud service providers, so they try their best to use vary efficient techniques for management, security and computation. One of these techniques for increasing flexibility and comparability of cloud datacenters is virtual machine migration. In this paper, we proposed a method that is more functional than the normal pre-copy method.

We built our method on the theory that decreasing the frequency of virtual machine's CPU has a direct relation with the memory page changing rate and as a consequence sending data over the network.

Therefore, we reduced the vCPU frequency of the virtual machine until the changing rate was higher than sending data through the network. In this way, memory pages were transferred to the destination host quickly. As a result, the amount of sent data and total migration time significantly decreased. The results show that proposed method decreased the total migration time to around 25% and also declines the amount of sent data to 47% in compare to Pre-Copy method.

REFRENCES:

[1] Y. Ruan, Zh. Cao, Y.Wang, "Efficient Live Migration of Virtual Machines with a Novel Data Filter", Procedings of Network and Parallel Computing (NPC 2014), Ilan, Taiwan, Sept, 2014.

[2] M. Forsman, A. Glad, L. Lundberg, D. Ilie, "Algorithms for automated live migration of virtual machines", Journal of Systems and Software, 1 March 2015, Vol.101, pp.110-126.

[3] M .Atif, P. Strazdins, "Adaptive parallel application resource remapping through the live migration of virtual machines", Future Generation Computer Systems-The International Journal Of Grid Comput, 2014 Jul, Vol.37, pp.148-161.

[4] J. Changyeon, E. Gustafsson, S. Jeongseok, E.Bernhard, "Efficient live migration of virtual machines using shared storage", ACM SIGPLAN Notices, August 2013, Vol.48(7), pp.41-50.

[5] R. Yonghui, C. Zhongsheng, C. Zongmin, "Pre-Filter-Copy: Efficient and Self-Adaptive Live Migration of Virtual Machines", IEEE Systems Journal, 5 November 2014.

[6] E. Baccarelli, D. Amendola, N. Cordeschi, "Minimum-energy bandwidth management for QoS live migration of virtual machines", Computer Networks, 2015 Dec 24, Vol.93, pp.1-22.

[7] K. Changhyeon, J. Changho, L. Wonjoo, Y. Sungil, "A parallel migration scheme for fast virtual machine relocation on a cloud cluster", The Journal of Supercomputing, 2015, Vol.71(12), pp.4623-4645.

[8] L. Kangkang, Zh. Huanyang, W. Jie, D. Xiaojiang, "Virtual machine placement in cloud systems through migration process", International Journal of Parallel, Emergent and Distributed Systems, 03 September 2015, Vol.30(5), p.393-410.

[9] M. Zoltán Ádám, "Allocation of virtual machines in cloud data centers-a survey of problem models and optimization algorithms", ACM Computing Surveys, 1 August 2015, Vol.48(1).

[10] Sh.Z. Rad, M.S. Javan, M.K, Akbari, "Providing a Solution for Live Migration of Virtual Machines in Eucalyptus Cloud Computing Infrastructure without Using a Shared Disk", CLOUD COMPUTING 2012, The Third International Conference on Cloud Computing, GRIDs, and Virtualization, Nice, France, July 2012.

[11] J. Hai, D. Li, W. Song, S. Xuanhua, and P. Xiaodong, "Live virtual machine migration with adaptive, memory compression," in IEEE International Conference on Cluster Computing and Workshops, CLUSTER '09, pp. 1-10.

[12] E. Zaw, N. Lar Thein," Improved Live VM Migration using LRU and Splay Tree Algorithm " International Journal of Computer Science and Telecommunications ,Volume 3, Issue 3, March 2012.

[13] C. Clark, K. Fraser, S. Hand, J. G. Hansen, E. July, C. Limpach, I. Pratt, and A. Warfield, "Live Migration of VMs", Proceedings of the 2nd USENIX Symposium on Networked Systems Design and Implementation, 2005.

[14] P. Padala, K. G. Shin, X. Zhu, M. Uysal, Z. Wang, S. Singhal, A. Merchant, and K. Salem, "Adaptive control of virtualized resources in utility computing environments," in *Proceedings of the 2nd ACM* SIGOPS/EuroSys European Conference on Computer Systems (EuroSys'*07)*, 2007, pp. 289–302.

[15] S. K. Bose, S. Brock, R. Skeoch, and S. Rao, "CloudSpider: Combining replication with scheduling for optimizing live migration of virtual machines across wide area networks," *11th IEEE/ACM International* Symposium on Cluster, Cloud and Grid Computing, CCGrid 2011, May 2011, pp. 13-22.

[16] B. Jiang, J. Wu, X. Zhu, and D. Hu, "Priority-Based Live Migration of Virtual Machine " ,Springer-Verlag Berlin Heidelberg 2013, GPC 2013, LNCS 7861, pp. 376–385, 2013.

[17] Hines, M.R., Gopalan, K.: Post-copy based live virtual machine migration using adaptive pre-paging and dynamic self-ballooning. In: Proceedings of the ACM/Usenix International Conference on Virtual Execution Environments (VEE 2009), pp. 51–60 (2009)

[18] D. Chen,_, H.Yang, Q. Xue, and Y. Zhou," Live Migration of Virtual Machines Based on DPDT" , Springer-Verlag Berlin Heidelberg 2013, CWSN 2012, CCIS 334, pp. 26–33, 2013.

[19] Zhang, X., Huo, Z., Ma, J., Meng, D.: Exploiting Data Deduplication to Accelerate Live Virtual Machine Migration. In: IEEE International Conference on Cluster Computing, Cluster 2010, pp. 88–96 (2010).

[20] Michael, N., Lim, B.H., Greg, H.: Fast transparent migration for virtual machines. In: Proceedings of the USENIX Annual Technical Conference 2005 on USENIX Annual Technical Conference, Anaheim, p. 25 (2005).

[21] Liu, Z.B., Qu, W.Y., Liu, W.J., Li, K.Q.: Xen Live Migration with Slowdown Scheduling Algorithm. In: 2010 International Conference on Parallel and Distributed Computing, Applications and Technologies, PDCAT, pp. 251–221 (2010).

[22] Stress Test http://freecode.com/projects/stress