Research Journal of Applied Sciences 12 (2): 131-138, 2017
ISSN: 1815-932X

On Speeding up Virtual Machine Migration Through Integrated Data De-Duplication Methods

[1]K. Aruna Kumari, [2]K. Raja Sekhara Rao and [3]J.K.R. Sastry
[1]SRKR Engineering College, Bhimavaram, West Godavari, Andhera Pradesh, India
[2]Usha Rama College of Engineering and Technology, Telaprolu, Krishna, Andhera Pradesh, India
[3]KL University, Vaddeswaram, Guntur, Andhra Pradesh, India

Abstract: Virtualization is a concept implemented as a major part of cloud computing infrastructure to offer everything as a service. Hypervisors are supported within the cloud computing platform using which many of the virtual machines can be created as per the demand of the end users. The load on the Virtual Machines (VM) greatly varies while some are over used, some may be under used or average used. Migration of a process, data and sometimes both needs to be done from one VM to other in such a way that the load on the servers is well balanced leading to high performance computing. Load balancing helps in a big way especially in controlling server sprawling and efficient use of the computing resources. Live migration has to be undertaken seamlessly without the initiation of any process from the client application. During migration, movement of data from one server to another has to be undertaken which generally consume lot of time. Data stored on a server by a user is historical and live and enormous amount of duplicated data is stored on the server especially for maintaining different versions of the same data. Elimination of duplicated data before migration of the same from one server to the other will greatly enhance the speed of migration. This study outlines some of the strategies for implementing data de-duplication within migration processes.

Key words: Virtualization, live VM migration, data de-duplication, cloud computing infrastructure, elimination of duplicated

INTRODUCTION

Cloud computing is being used extensively all over the world over for achieving large scale economies through sharing of computing resources. In the world lot of computing power is being un-utilized whereas lot many users are starving for computing resources not being able to afford the same or due to the gestation periods required for creating the infrastructure required for running the desired applications. The evolution of the internet helped to create a backbone that helps in making available the widely distributed computing resources to the clients who could be geographically situated. Cloud computing is built over service oriented architecture which makes everything that includes infrastructure, software and platforms that a client needs as a service.

A user can ask for a machine in terms of OS Software, disk space, etc. and the cloud computing platform will make the required resources as a virtual machine while the real resources required by the user may be situated in one are more of physical machines. Virtualization is the main concept that is used for supporting cloud computing. Everything "as a service" is the premise on which cloud computing is built. Infrastructure as a Service (IaaS) Platform as a Service (PaaS) Software as a Service (SaaS) are the premises on which cloud computing is built. IaaS is made is made possible by creating virtual machines which provide computing power, communication and connectivity and storage collectively as a service to the clients. A virtual machine shall have the software component that maintains details of the infrastructure attached to the VM through a set of files which include configuration file, virtual disk file, file for NVRAM settings, log file. Moving the virtual machine from one physical system to another physical system is known as migration of virtual machine. A single virtual machine or a group of virtual machines concurrently can be migrated. It is rather simple to migrate a VM that is totally contained in a single physical machine. It is however, complicated to migrate a VM when it has the resources allocated drawn from various physical machines. Migration of virtual machine/s will be done for the following reasons:

Load balancing: Distribution of workloads among the available resources.

Corresponding Author: K. Aruna Kumari, SRKR Engineering College, Bhimavaram, West Godavari, Andhera Pradesh, India

VM sprawling: Creation of VMs on a network that cannot be managed by the administrator.

Fault tolerance: When a physical machine is about to fail the VM running on that machine are moved to another machine.

Green computing: If a server is just running few VMs which can be moved to another server so that one server can be switched off so that less energy is consumed. For optimsing the uses of resources connected on to the cloud. A virtual machine can be migrated in three ways.

Cold VM migration: Where the VM is stopped and moved to the destination and restarted.

Warm migration: Virtual machine is suspended on source, RAM and CPU registers copied to the destination and resume it.

Live VM: Migration the applications on the VM continue to work while copying the VM related data to destination and then stop it for a while and start it at the destination. The advantage of live VM migration is that the client is not aware of the migration as the downtime of the VM is not noticeable.

Data de-duplication leads to storage optimization by maintaining a single copy of redundant data and referencing, it wherever required. Incorporating data de-duplication technique on the VM data will reduce the size of VM data quite drastically and as a result the time required to migrate the data would be lessened. The speed of migration will be faster even in the presence of low bandwidths.

Integration of data duplication methods into VM migration algorithms will help in achieving faster VM migration.

Live VM migration: Live VM migration is a feature provided by the hypervisor for relocating virtual machine or virtual machine instances from one physical server to the other at runtime. Figure 1 shows the working principle of VM migration. Software called hypervisor is run on each of the physical server. Hypervisor is responsible for creating and running the virtual machines on each of the physical server. Hypervisor keeps track of the VMs created and being operated on a particular server.

Virtual Infrastructure Manager (VIM) a piece of software that runs on different server or on one of the servers on which the hypervisor resides is responsible for managing the resources resident on the physical server. VIM communicates with each of the

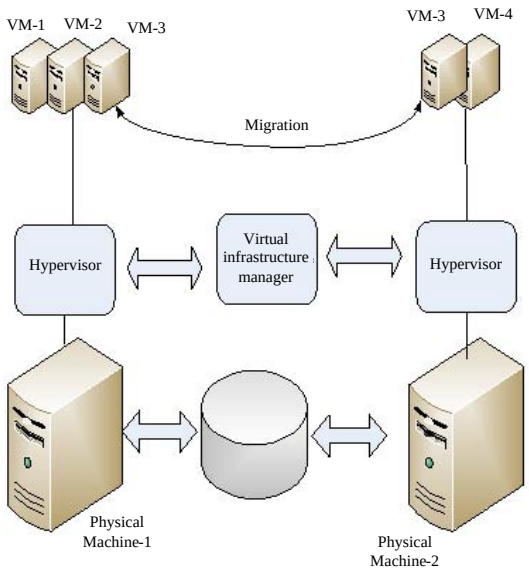

Fig. 1: VM migration flow

hypervisor for knowing and keeping track of the VMs and the resources occupied by the hypervisors. VIM on knowing that the a physical machine is overloaded, migrates some of the VMs on it to some other server which has not been exhausted with the resources due to creation and operation of the VMs on it.

When migration is needed, VIM commands a hypervisor to stop a VM on it and then the VM is migrated to a different server and the operation of the VM is resumed. When VM is migrated, the data used by the VM is migrated. The data to be migrated include processor status, storage status, cache status, memory status, etc. The processes and the state of running the processes on the VM must also be migrated and restored. One can use many strategies for adapting a migration process. A live VM migration can be performed using any of the strategies Which include pre-copy live, post-copy live and hybrid migration.

In the case of pre-copy live migration, the state of the VM is transferred before execution is switched from the source to the destination. Two phases of processing are built into a pre-copy algorithm. In the first phase, the state of resources used by the VM is pushed into the new VM in an iterative manner and in the second phase a minimal stop of the VM is carried and then the state of the processing being undertaken is copied. The transfer of resources being used by the VM must be undertaken iteratively especially for processing the dirty memory pages that get created as the processing on the VM is in progress. The memory pages get updated at a particular speed while the transfer of the memory pages depends on the availability of the bandwidth for transfer. It becomes

necessary to stop processing and then transfer the memory pages when the speed of updating the memory pages is much faster compared to speed of transfer which again is dependent of the availability of bandwidth.

In this case, the VM handles the client requests when it is in migration. In the case of post-copy live migration, execution of the VM is first switched off and the state is then transferred on demand to the newly created VM on a different server.

Post-copy migration defers the memory transfer phase until after the VMs CPU state has been transferred to the target and resumed there. In this case, the client requests during the process of migration are handled by the newly created VM on a different machine.

In the most basic form, post-copy first suspends the migrating VM at the source node, copies minimal processor state to the target node, resumes the virtual machine at the target node and begins fetching memory pages from the source over the network. Variants of post-copy arise in terms of the way the memory pages are fetched.

There are three main ways of handling page fetching in post-copy schemes which include post-copy via demand paging, post-copy via active pushing and post-copy via pre-paging in the case of post-copy via demand paging, VM is resumed at the destination server and then the page faults are serviced by requesting the referenced page over the network from the source node. Usually, this results in unacceptable total migration times and application degradation.

Literature review: VM migration is implemented for achieving load balancing, fault tolerance, power management and to carry system maintenance. Pre-copy algorithm is quite frequently used for effecting VM migration. The algorithm transfers many duplicate memory pages from source to destination. VM migration through this method takes longer time than expected. Zhang *et al.* (2010) have presented a method that combines VM migration through data de-duplication present in the memory. The method proposed by them considers the self-similarity of run time memory image and hash based finger prints to find the duplicate memory pages and sues run time encoding system for eliminating the duplicate memory pages.

In the case of cloud computing, the need for many physical servers is reduced through creation of many virtual machines which may run several operating systems on the same physical machine. Each of the virtual machine requires disk space in the range of giga bytes. Many methods have been presented in the past for reducing the storage through data de-duplication considering that the entire Storage is Networked (SAN). However, SAN is costly and limited in size and therefore cannot meet the increasing demand of the ever growing demand for virtual machines. Bhuvaneshwari have proposed a scalable de-duplication file system that can be applied on the VM images. The data de duplication system uses comprehensive set of storage features including instant cloning for VM images, on-demand fetching through a network and caching with local disks by copy-on-read techniques and simply identifying zero-filled blocks.

Some of the virtualization mechanisms supported as date requires that disk images are downloaded and the same be used for creating virtual machines. However, there could be common data blocks among different disk images using which the VMs are created as system data is generally common for all the VMs. Wanigasekara and Keppittiyagama (2014) presented a method that de-duplicates common data blocks among the disk images. They have introduced a new file system called buddy FS using which common data blocks are reused to create disk images. They have shown how semantic information can be effectively used for optimizing the data de-duplication process.

Memory state redundancy among several virtual machines that run on the same physical machine can be eliminated by identifying common memory pages shared by the VMs and replacing them with only one copy. Hashing is frequently used technique for identifying the common memory pages. The hash values are generally computed by the hypervisor adding overhead leading to poor performance of the VMs. The performance of the hypervisors can be improved if the hashing and de duplication is implemented outside the scope of the Hypervisor. CPU time is consumed for computing hash values leading the poor performance of VMs. The performance of VMs will be greatly improved if the hashing is done when CPU is idle. Ying have presented a new primitive called Deferrable Aggregate Hyper call (DAH) that allows the computation of hash values during CPU idle time leading to the improvement in the performance of the VMs.

VMs use network storage systems for conducting I/O. Many duplicate data blocks are maintained pertaining to the VMs in the network storage. I/O traffic as such would be high. Feng and Schindler (2013) explored de-duplication of content stored on the network storage systems. They have presented a method that aims at caching the data stored in the storage systems. They have implemented data de duplication techniques while caching the data read from the storage systems. They have shown that the size of the data that should be cached reduces quite drastically when data de-duplication

techniques are used while caching. The cache hit ratio will also increase when larger size of the cache is used.

Virtual machines are created through images which are stored on the network storage systems. To deal with any of the disasters, the network storage is backed up periodically. The storage as such runs into terra or pico bytes. Large amount of duplicate data related to VMs is stored in the network storage systems. The existing data de-duplication techniques are not quite suitable for removing the duplication that exists in VM images. The performance of the VMs will get greatly affected when VMs are loaded with the issue of data de-duplication. Xu *et al.* (2014) have proposed a local de-duplication method that can speed up the data de-duplication of the VM image. The method is based on an improved k-means clustering algorithm which could classify the metadata of backup image to reduce the search space of index lookup and improve the index lookup performance.

De-duplication is computationally expensive. Roemer *et al.* (2014) have studied the effects of using de-duplication during VM migration. The improvements in the performance of VMs can be seen due to implementation of VM migration and data de-duplication. Much more efficiencies can be achieved by reducing the data de-duplication time. An approach has been presented that deals with grouping virtual machines based on similarity and then it is shown that that the overhead due to de-duplication can be drastically reduced.

Luo *et al.* (2008) have considered migration of the whole system from one physical machine to the other. All the system resources that include CPU state, memory data, local disk storage and VM are migrated during runtime. They have presented a three phase algorithm which minimizes downtime that gets caused when migration is undertaken and also ensure data integrity and consistency. An incremental approach is also presented when reverse migration has to be undertaken. Another algorithm has been presented for reducing the amount of data that must be migrated through use of an algorithm called "Block-bitmap". Block-bitmap tracks all the write accesses to the local disk while undertaking migration of the same. The Block-bitmap is used for synchronizing the local disk with the disk to which the migration is undertaken.

Many logical sharing mechanisms such as shared library, symbolic links, etc. are used in respect to memory and storage by many of the currently released operating systems. The sharing feature supported by the operating systems cause security and management problem especially when dynamic logical sharing is used. Dynamic sharing is used for effecting the search path replacement attack, global offset table overwrite attack, dependency hell, etc. Suzaki *et al.* (2010) have presented that self contained binaries eliminate the problem of logical sharing. The overhead caused due to migration of self contained binaries is mitigated through memory and storage de-duplication methods. The have investigated the effect of de-duplication through use of kernel same page merging and lookback addressable storage. Many virtual machines are created on the same physical machine for supporting many users to use the same machine in the field of cloud computing. Memory is shared by all the virtual machines as a result lots of duplication of the same data is duplicated in the memory. Thus, it becomes necessary that the memory usage must be optimized through de-duplication of the memory on per page basis. Shaikh *et al.* (2014) have proposed use of single copy of the duplicated pages in physical memory through use of a mechanism called copy-on-write. They have implemented VMDedup which does not require user configuration provides automatic memory de-duplication built within the hypervisor itself which will benefit across the operating system code, data and application binaries. They have implemented VMDedup within Xen hypervisor that supports both para and fully virtualized instances of the operating system.

One of the major issue in cloud computing is energy management. Quite an amount of energy is consumed producing lot of radiation. Consolidating VMs dynamically and efficient use of storages are some of the approaches being used to reduce energy consumption. Storage utilization can be improved by removing duplicate copies existing in memory and in storage through de-duplication. Sachin Sadar and colleagues have presented ACS based VM consolidation along with distributed de-duplication they have presented that underutilized physical machines can be switched of or put into low power mode by migrating the existing VM on that machine. They have presented that the reliability of storage will greatly improve when de-duplication is undertaken in distributed manner.

Live migration of VMs especially when CPU status and memory is migrated, service to the end customer is bound to be effected. When migration is done consuming prolonged durations, it is possible that the down time of the entire system might fall down quite drastically. Significant network resources would be required for effecting migration. When physical servers are situated apart, time required for migration may be more due to enhanced distance between the servers. Svard *et al.* (2011) have presented a novel algorithm that dynamically uses a specific order of memory pages such that re-

transfers of frequently dirtied pages is completely avoided. The total migration time is reduced due to reduction in the number of memory pages that should be transferred. They have combined this technique with a compression technique that reduces the size of the data and thereby reducing the migration time.

Live migration of virtual machines involves many barriers that include high latency, high power consumption, low performance, etc. These barriers can be reduced through use solid state drives which can be used for constructing cache between the memory and the storage systems. While that being the case the process of virtualization involve too many duplicate data blocks in the cache. None of the existing caching algorithms takes into account the issues of existence duplicate data blocks in the cache. Chen *et al*. (2015) has proposed an excellent duplication aware SSD based cache processing architecture. They have also proposed adaptive cache replacement algorithm based on replacement strategy. It has been proved that the algorithms proposed them greatly enhance the cache hit ratio.

Considering the acute time line required for providing the service to the end customers, it became necessary that many VMs are to be migrated simultaneously while migrating one VM after another is the general approach used for effecting the VM migration. The migration of VMs from one set of physical machines to another set of physical machine would be required yet times. Such kind of migration is called GANG migration. GANG migration has the limitation that involves huge network traffic and yet time may lead to overloading core network links.

Deshpande *et al*. 2013 have presented a method called GMGD (Gang Migration Using Global De-Duplication) for reducing the network traffic. GMGD identifies the duplicate pages among a GANG of VMs that run on multiple physical machines that have been formed as a cluster and eliminate the transmission of the same. They have tested the method considering 30 Gbits Ethernet cluster network with 10 Gig network links and found that the method reduce the down time by about 42%.

MATERIALS AND METHODS

Post-copy via active pushing method allows the pages to be pushed to the target while the VM is in executing at the target server. The pages that have faults recorded are serviced at higher priority than the other pages. Post-copy via pre-paging methods involves in estimating the spatial locality of the VMs memory access pattern for anticipating the occurrence of major page faults. The pages to be migrated into the new VM changes with the occurrence of a new page fault in real time. As the number of pages to be transferred reduces, the application degradation is minimized.

In the case of hybrid live migration, only the CPU state of the VM is transferred and then the execution state of the VM is switched and the state is then transferred on demand to the destination. The hybrid live migration is a special case of post copy algorithm which is preceded by a limited pre-copy stage processing. The main idea of this approach is to transfer a subset of the most frequently accessed memory pages before the execution of the VM is switched to the destination. The performance degradation of the VM once it is resumed can be reduced due to the need for transferring fewer memory pages which are needed. However, the pre-copy phase can lead to a slightly longer total migration time than for pure post-copy algorithms and it is also challenging to choose the correct set of pages for transfer.

The main issue however is that some of the pages in the memory could be redundant and unused. It is necessary to find the redundant pages, replace them with the pointers and then the can be migrated. This kind of processing reduces drastically the size of the data that must be migrated.

Data de-duplication: Data de-duplication is a technique for reducing the amount of storage space needed to save the data. De-duplication eliminates extra copies by saving just one copy of the data and replacing the other copies with pointers that lead back to the original copy. Data de-duplication can generally operate at the file, block or byte level thus defining minimal data fragment that is checked by the system for redundancy.

A hash algorithm generates a unique identifier-hash number-for each analyzed chunk of data. It is then stored in an index and used for figuring out duplicates, the duplicated fragments have the same hash numbers. Several data de-duplication methods are in use which includes file-level data de-duplication, block level data de-duplication and byte level data de-duplication.

File-level de-duplication also commonly referred to as Single-Instance Storage (SIS). A set of memory pages that must be transferred can be grouped and recognized as a file. File-level data de-duplication compares a file to be backed up or archived with those already stored by checking its attributes against an index. If the file is unique, it is stored and the index is updated; if not only a pointer to the existing file is stored. The result is that only one instance of the file is saved and subsequent copies are replaced with a “stub” that points to the original file.

Block-level de-duplication operates on the sub-file level. As its name implies, the file is typically broken down into segments "chunks or blocks" that are examined for redundancy vs. previously stored information. The most popular approach for determining duplicates is to assign an identifier to a chunk of data, using a hash algorithm for example that generates a unique ID or "fingerprint" for that block. The unique ID is then compared with a central index. If the ID exists then the data segment has been processed and stored before. Therefore, only a pointer to the previously stored data needs to be saved. If the ID is new then the block is unique. The unique ID is added to the index and the unique chunk is stored. De-duplication using block-level mainly comprise of four steps which include chunking, calculating fingerprint, finding the finger print similarity and storing the records. In this case, data are divided into the sequence of bytes called chunks and then the chunks are examined for redundancy. A hash value is calculated for each of the block which is called the finger print. The finger print of one block should be different for the uniqueness of the data. The finger prints are same and the data is duplicate in which the data is replaced with the pointer to the finger print,thus avoiding the data duplication at the block level.

Byte-level data de-duplication performs a byte-by byte comparison of new data streams versus previously stored ones; a higher level of accuracy can be delivered. De-duplication products that use this method have one thing in common: It's likely that the incoming backup data stream has been seen before, so it is reviewed to see if it matches similar data received in the past.

Theoretically, the more detailed analysis is the higher de-duplication rate of storage should be obtained. In actual practice, all three levels have their peculiarities. The file-level de-duplication can be performed most easily. It requires less processing power since file's hash numbers are relatively easy to generate. However, there is the reverse side of this model. If only one-byte of a file is changed, its hash number also changes. As a result both file versions will be saved to storage.

Implementing DD in live VM migration: There are primarily three migration techniques and three data de-duplication techniques which when combined will yield 9 different ways of undertaking the migration. The combinations are shown in Table 1.

For further investigation block based data de duplication method has been tried and memory is generally organized as pages and each page as such could be recognized as block of data. In this study, the hybrid algorithm has been modified to include block-level data de-duplication into it and find the speed with which

Table 1: VM migration using data de-duplication strategies

Comb serials	Types of migration	Type of de-duplication
1	Post copy	File based
2		Block based
3		Byte based
4	Pre-copy	File based
5		Block based
6		Byte based
7	Hybrid	File based
8		Block based
9		Byte based

Table 2: Page table for VM migration

Page No.	Block No.	Start address in decimal notation	Page size	Updated status
1200	1	#0000	512	Y
1207	2	#0513	512	Y
1230	3	#1049	10	N

the VM migration can be achieved. Two algorithms have been developed for undertaking migration with no data de-duplication and with data de-duplication. VIM calls the migration processes whenever it finds that there is a need to move the Virtual machine from one server to the other.

Algorithm with no data de-duplication

Step-1: Get the memory page table related to the VM concerned by making a system call. The details of the page table will be as shown in Table 2.

Step-2: Create a new virtual machine on a physical server which is found to have adequate resources to contain the VM through making a system call to VIM.

Step-3: Transfer the CPU status to new VM.

Step-4: Transfer the memory pages that are frequently updated.

Step-5: Commence the execution of the newly formed VM.

Step-6: Transfer the pages that are either not updated or infrequently updated. This migration process is modified to undertake the following algorithm.

Algorithm with data de-duplication

Step-1: Get the memory page Table related to the VM concerned by making a system call. The details of the page Table will be as shown in Table 3.

Step-2: Calculate the hash value of every page and update the page table with hash value.

Step-3: If computed hash value is same as the existing hash, then update the page row with block number having the same hash for cross referencing and change the block

Table 3: Page table for VM migration through data de-duplication

Page No.	Block No.	Start address in decimal notation	Page size	Hash No.	Cross referenced block No.	Updated status
1200	1	#0000	512	12301		Y
1207	2	#0513	512	12598		Y
1230	3	#1049	10	12301	0001	N

size to 10 (5 bytes for cross reference, 4 bytes for cross referencing and one byte to store the updated status). The leads to a saving 502 bytes which need not be migrated.

Step-4: Create a new virtual machine on a physical server which is found to have adequate resources to contain the VM through making a system call to VIM.

Step-5: Transfer the CPU status to new VM.

Step-6: Transfer the memory pages that are frequently updated and those pages that have no cross references.

Step-7: Commence the execution of the newly formed VM

Step-8: Transfer the cross reference when a page having the cross reference is to be transferred and on the target side create the page row as per the page format. This speeds of the process of migration.

RESULTS AND DISCUSSION

Experimentation and results: To test the above mentioned algorithms which have been coded using JAVA language, cloudsim (cloud simulator software) has been used. Under the tool two physical machines have been configured. On machine-1 three virtual machines, and on machine-2, 1 virtual machine has been created the details of which have been shown in Table 4.

Cloud sim has been fed with the program written in Java containing the code related to algorithm-1 for virtual machine migration using no data de-duplication and the migration requirement has been set. In the Java program the CPU time has been noted in the beginning and at the end of the execution of the JAVA program that effects migration. The above mentioned process has been reported with another JAVA program and the execution times have been displayed and noted. The results obtained are shown in Table 5. From Table 5, it could be seen that the hybrid based VM migration algorithm that implements data de-duplication at the block level has been proved to be quite effective while no extra resources have been used by the migration program that implements the data de-duplication. It could be seen that less time has been taken for completing the migration when data de duplication method has been used.

Table 4: Details of physical and virtual machines created on cloudsim

Types of machine	Configuration parameter	Machine-1	Machine-2
Physical	OS	Windows 8.1	Red Hat
	CPU	Dual core 2.3GHz	Dual core 2.3 GHz
	Storage	300 GB	300 GB
	Memory	8 GB	8 GB
	Data base	SQL 8.0	Oracle 8i
Virtual	VM-1	Windows 8.1	
		Dual core 2.3 GHz	
		100 GB	
		2 GB	
		SQL 8.0	
Virtual	VM-2	Windows 8.1	
		Dual core 2.3 GHz	
		100 GB	
		2 GB	
		SQL 8.0	
Virtual	VM-3	Windows 8.1	
		Dual core 2.3 GHz	
		150 GB	
		4 GB	
		SQL 8.0	
Virtual	VM-4		Red Hot 7.0
			Dual core 2.3 GHz
			150 GB
			4 GB
			Oracle 8i

Table 5: Performance comparison of VM migration algorithms

Measurements	VM without data de-duplication	VM with data de-duplication
CPU start time	10:10:37	12:12:23
CPU end time	10:11:43	12:13:00
Computational time	00:01:06	00:00:37
Memory used	3.7 GB	3.7 GB

CONCLUSION

Every cloud provider is responsible to give services to the customers most effectively and efficiently especially with least response time. Every user is required to be given the machine of his specification. Users are provided with virtual machines of their choice. The physical machines may be overloaded especially by creating too many VMs on those machines. It is the responsibility of the cloud to ensure that all the physical servers are evenly loaded so that most appropriate response time is ensured for the clients. Migrating one VM from another provides the basis for load balancing leading to optimum use of the resources. Employment of data de-duplication methods helps in carrying with the migration most effectively and using least amount of time making the customer un-aware of such migration taking place.

REFERENCES

Chen, X., W. Chen, Z. Lu, P. Long and S. Yang *et al*., 2015. A Duplication-aware SSD-Based cache architecture for primary storage in virtualization environment. IEEE. Syst. J., 1: 1-12.

Deshpande, U., B. Schlinker, E. Adler and K. Gopalan, 2013. Gang migration of virtual machines using cluster-wide deduplication. Proceedings of the 13th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing, May 13-16, 2013, Delft, Netherlands, pp: 394-400.

Feng, J. and J. Schindler, 2013. A reduplication study for host-side caches in virtualized data centre environments. Proceedings of the 2013 IEEE 29th International Symposium on Mass Storage Systems and Technologies (MSST), May 6-10, 2013, IEEE, Sunnyvale, California, ISBN: 978-1-4799-0218-7, pp: 1-6.

Luo, Y., B. Zhang, X. Wang, Z. Wang and Y. Sun *et al*., 2008. Live and incremental whole-system migration of virtual machines using block-bitmap. Proceedings of the 2008 IEEE International Conference on Cluster Computing, September 29-October 1, 2008, IEEE, Beijing, China, ISBN: 978-1-4244-2639-3, pp: 99-106.

Roemer, J., M. Groman, Z. Yang, Y Wang and C.C. Tan *et al*., 2014. Improving virtual machine migration via deduplication. Proceedings of the 2014 IEEE 11th International Conference on Mobile Ad Hoc and Sensor Systems (MASS), October 28-30, 2014, IEEE, Philadelphia, Pennsylvania, ISBN: 978-1-4799-6036-1, pp: 702-707.

Shaikh, F., F. Yao, I. Gupta and R.H. Campbell, 2014. Vmdedup: Memory de-duplication in hypervisor. Proceedings of the 2014 IEEE International Conference on Cloud Engineering (IC2E), March 11-14, 2014, IEEE, Urbana, Illinois, ISBN: 978-1-4799-3766-0, pp: 379-384.

Svard, P., J. Tordsson, B. Hudzia and E. Elmroth, 2011. High performance live migration through dynamic page transfer reordering and compression. Proceedings of the 2011 IEEE 3rd International Conference on Cloud Computing Technology and Science (CloudCom), November 29- December 1, 2011, IEEE, Umea, Sweden, ISBN: 978-1-4673-0090-2, pp: 542-548.

Wanigasekara, N. and C.I. Keppittiyagama, 2014. BuddyFS: A file-system to improve data deduplication in virtualization environments. Proceedings of the 2014 8th International Conference on Complex, Intelligent and Software Intensive Systems (CISIS), July 2-4, 2014, IEEE, Colombo, Sri Lanka, ISBN: 978-1-4799-4325-8, pp: 198-204.

Xu, J., W. Zhang, S. Ye, J. Wei and T. Huang, 2014. A lightweight virtual machine image deduplication backup approach in cloud environment. Proceedings of the 2014 IEEE 38th Annual Conference on Computer Software and Applications Conference (COMPSAC), July 21-25, 2014, IEEE, Beijing, China, ISBN: 978-1-4799-3575-8, pp: 503-508.

Zhang, X., Z. Huo, J. Ma and D. Meng, 2010. Exploiting data deduplication to accelerate live virtual machine migration. Proceedings of the International Conference on Cluster Computing, September 20-24, 2010, Heraklion, Crete, pp: 88-96.