International Journal of Electrical and Computer Engineering (IJECE)
Vol. 11, No. 1, February 2021, pp. 654~663
ISSN: 2088-8708, DOI: 10.11591/ijece.v11i1.pp654-663

Resumption of virtual machines after adaptive deduplication of virtual machine images in live migration

Naga Malleswari T.Y.J., Senthil Kumar T., Jothi Kumar C.
Computer Science and Engineering, SRM Institute of Technology, India

Article Info

Article history:

Received Jan 23, 2020
Revised Jul 9, 2020
Accepted Aug 7, 2020

Keywords:

Akka stream
Cloud computing
Deduplication
Load balancing
Pre-copy approach
Streaming analytics

ABSTRACT

In cloud computing, load balancing, energy utilization are the critical problems solved by virtual machine (VM) migration. Live migration is the live movement of VMs from an overloaded/underloaded physical machine to a suitable one. During this process, transferring large disk image files take more time, hence more migration and down time. In the proposed adaptive deduplication, based on the image file size, the file undergoes both fixed, variable length deduplication processes. The significance of this paper is resumption of VMs with reunited deduplicated disk image files. The performance measured by calculating the percentage reduction of VM image size after deduplication, the time taken to migrate the deduplicated file and the time taken for each VM to resume after the migration. The results show that 83%, 89.76% reduction overall image size and migration time respectively. For a deduplication ratio of 92%, it takes an overall time of 3.52 minutes, 7% reduction in resumption time, compared with the time taken for the total QCOW2 files with original size. For VMDK files the resumption time reduced by a maximum 17% (7.63 mins) compared with that of for original files.

Corresponding Author:

Naga Malleswari T.Y.J.,
Computer Science and Engineering,
SRM Institute of Technology,
SRM Nagar, Kattankulathur, 603203, Kanchipurm, Chennai, TN, India.
Email: nagamalt@srmist.edu.in

1. INTRODUCTION

The evolution of various technologies, such as grid computing, service-oriented architecture, virtualization, autonomic, and utility computing, leads to cloud computing. Cloud computing services consist of data centres in large scale to host the user's applications through hardware virtualization. In a fully virtualized manner, vast amounts of processing power are accessed by the users by grouping the resources. The goals of cloud computing are a pool of resources, providing elasticity rapidly, broad network access, providing capabilities on-demand and measured service [1]. A hypervisor or virtualization layer or virtual machine monitor (VMM) or play a prominent role in the virtualization concept. It is introduced in the middle of the operating system and the hardware. The hypervisor is a virtualization program [2] which creates VMs and manages VMs on a system. As virtual machines share the physical resources like disk space, network bandwidth, and cores of CPU, resource management, load balancing, and power consumption become critical tasks. Some of these issues can be resolved by dynamic resource management. Live virtual machine migration or live migration is the dynamic VM management activity that resolves the challenges of resource management and power management. During the live migration process, the virtual disk image files which are larger size take more significant time to transfer to the target host and hence more migration time and downtime. Hence

the objective of this paper is to reduce the size of the virtual disk image file that is going to be migrated using the proposed solution adaptive deduplication. The performance of this process measured in terms of deduplication ratio, migration and resumption time.

In this paper, the proposed adaptive deduplication implemented by using both fixed and variable-length block deduplication techniques together. They used to identify and eliminate redundant data in large virtual disk images to achieve better migration performance. As its name, depending on the size of the virtual disk image the type of deduplication was chosen. For smaller (at most 1GB) virtual disk image files fixed-length deduplication technique used with 4 KB as the chunk size. For larger VM disk files, both deduplication techniques, fixed and variable length together applied. The Rabin-Karp rolling hash algorithm found the boundaries of the chunks in variable-length block deduplication. As deduplication process performed double time, the VM disk file size reduced much. The adaptive deduplication technique deployed in the source host and the process done before transferring the VM memory pages to the target system. As the size of the virtual machine memory file to be transferred reduced, total migration time and downtime minimized significantly. As deduplication, an additional operation introduced in live VM migration, and deduplication time might become an overhead. Hence, the deduplication process parallelized using multi-threading in Akka stream. Once the total pages of VM migrated to the target host, these deduplicated pages are taken and reunited to build a single VM disk image file. The VM is resumed back in the target host within no time as the disk image file size reduced using adaptive deduplication. The deduplication ratio, migration time and the resumption time of the live migration process measured and analyzed in this paper. The remaining paper organized as, section 1 gives a brief description of live migration, deduplication and streaming analytics. The work is related to the live migration after deduplication presented in section "Related work". Section 3 discusses the motivation and architecture of the system. Implementation section presents the experimental setup in section 4. Results discussed in section 5. The paper concluded with the conclusion and future work.

Virtual machine migration [3] is the migration of a VM from a given physical server to another, by which balancing the load, server consolidation, fault tolerance, online maintenance [4] and minimization of the utilization of resources and energy. The significant goals [5, 6] of VM migration are i) load balancing where equal load distribution achieved by moving VMs from overloaded host to machines having the minimum load; ii) Server consolidation in which the VMs from minimum load server is migrated to the server which is not overloaded, thus consolidating all VMs in a single server to reduce power consumption; iii) Fault tolerance is the prediction of the failure of a server to avoid performance degradation. There are two categories of VM migration [7]. One of them is non- live Migration where VM is switched off and transferred to the target host. The significant drawbacks are downtime and performance degradation.

The other one is Live Migration. It is the movement of a VM from a given system to the other without shutting off the power of the VM. In live virtual machine migration, the internal components such as state of memory and processor registers migrated to target. The external state information such as virtual disk information is accessed through network attached storage (NAS) [8] by physical machines as and when required. All the components of a VM such as network connectivity, storage and memory are entirely moved from the source host to the destination host. As the migration process is seamless service, interruption not observed by the customer. Hence, suspended time and downtime of the VM is very less. Memory migration takes a prominent role in VM migration. This process of moving the instance of VM memory from the source server to the target server consists of the following phases [9]:

a. Push phase: At the point when a virtual machine on the source as yet running, the VMM moves the entire pages of memory to the target machine. The pages get dirtied (modified) during transmission. These modified pages transferred again until the copying rate of the pages greater than the rate of pages modified to guarantee consistency.
b. Stop-and- copy phase: The source VM halted, and all the VM pages are moved to the new instance and is resumed back on the target. c) Pull phase: The page fault happens when a page required by the VM and is not available on the target. As a result, the page is fetched from the source virtual machine through the network to the destination. There are two approaches to live virtual machine migration. They are i) Post-copy approach; ii) Pre-copy approach. The pre-copy approach is the predominant one. It contains push phase and stop and copy phases. High reliability of the system is achieved [10] by this approach.
    1) The hypervisor receives a request for migration from source destination. (Pre-Migration).
    2) After checking the available resources on the target, the resources reserved on the destination for VM. If the resources are not available, then the VM on the source machine does not affect. (Reservation).
    3) The working set of pages of VM memory from the origin moved to the destination.
    4) A few memory pages of a virtual machine are modified (dirtied) during the migration process and are transferred again and again iteratively in several rounds until the copying rate greater than the dirtying rate [11] (Iterative Pre-copy).

5) Stop the instance on the source machine, and the new destination receives further pages (Stop and Copy) [12].
6) The destination sends the acknowledgement to the source once the operating system image is received. This is to avoid receiving failure copies (Commitment).
7) On the target machine, VM starts and resumes after successful migration (Activation).

Figure 1 shows the pre-copy approach that contains push phases in which the dirty pages are moved to the destination repeatedly. During this process, many memory pages updated in the source host called dirty pages. The iterative pushing phases stopped when the rate of dirtying the pages is not more than the transferring rate of pages, then the stop and copy phase is executed [13]. In this iterative push phase, many pages with zeros (zero pages), the pages with similar content of above 80 % (identical pages) and the pages which contain 60% to 80% content similar (similar pages) migrated to the destination for VM [14] which are not at all required for instance to resume. In this approach, the entire state of an instance is to be migrated. This state contains little state information of the connected devices and is moved to the destination machine easily. There is no need of transferring the disk state information as NAS is having that. The time for migration and downtime is not affected by transferring the small amount of VMs CPU state. Gigabytes of information of Memory state necessary and hence transferred. Memory state information contains the information about processes running on VM and guest OS memory [15] virtual machine memory content includes dirtied memory, requested memory, VM used memory, Allocated memory, and VM configured memory. Among all, VM configured memory is vast in size, consisting of a virtual machine disk image file.

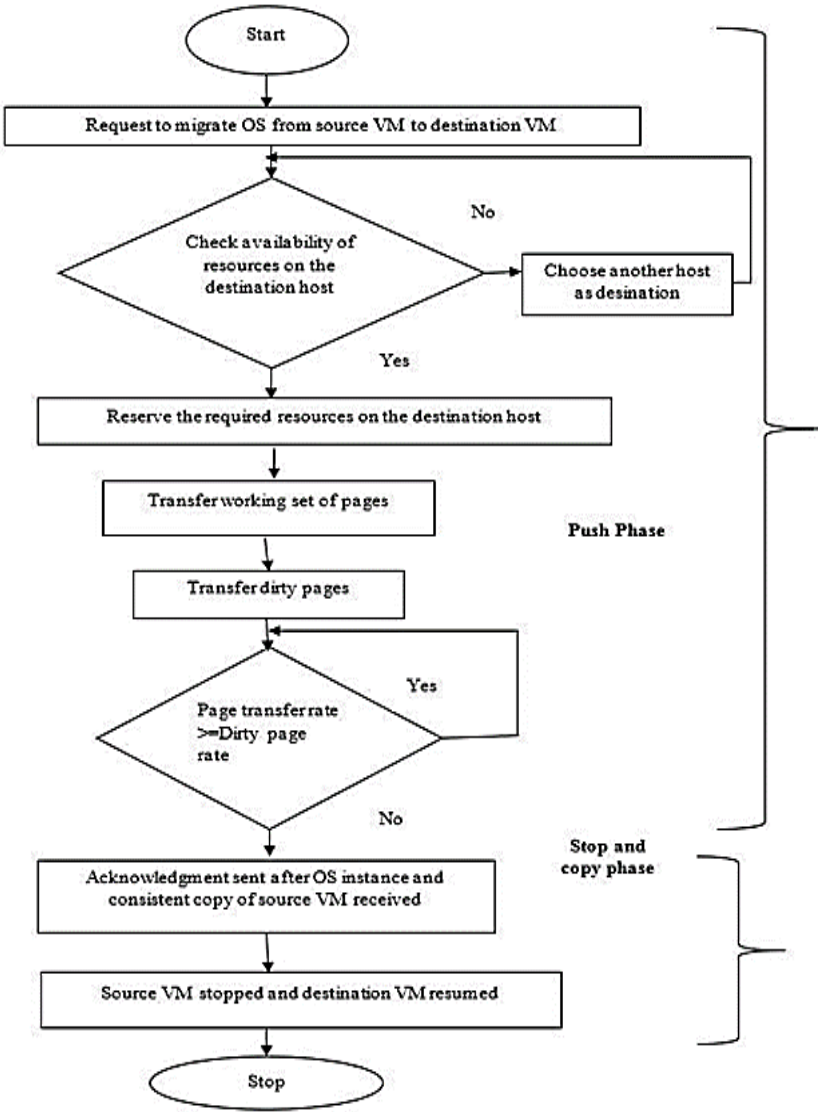

Figure 1. Pre-copy approach with iterative push, stop and copy phases machines

The metrics to measure the live migration performance are:

a. Application Degradation: During migration, the performance of the applications running in the migrated VM degrades.
b. Preparation Time: The time duration when migration initiated, minimal CPU state transferred to the destination. It includes the entire process of iteratively pushing the dirty pages to the destination host [16].
c. Down Time: The duration that a VM suspended (out of service) before it resumes on the target host.
d. Resume Time: The time took by the VM that migrated to resume its operation again at the target machine [17].
e. Total Migration Time: The entire time is taken for an instance on the source machine to migrate and to resume back on the target.
f. Network Traffic Overhead: The additional operations overhead affect the VM migration technique [18].
g. Pages Transferred: The entire set of memory pages of a VM that moved in the migration process [19].

Data deduplication, a technology where the required space reduced by eliminating the duplicate data in a given file set and the unique data is stored. It merely stores the link to the original data, thus saving the space without storing duplicate data again [20]. The classification in Figure 2 shows the types of deduplication techniques [21]. Based on data, the techniques classified as file-level and block level. Further block-level classified into fixed-length and variable-length deduplication techniques. Based on the location where the data duplicated, they are client level and target level. The target level deduplication further categorized as in-line and post-process how the techniques are implemented based on that they classified as hash-based and content-based. In adaptive deduplication, block-level deduplication techniques, with hash computation and at client discussed. In Fixed length block deduplication, the file split into various blocks of fixed size. The blocks with similar content removed to decrease the file size. In Variable-length block deduplication, the boundaries of the blocks are computed depending on the block contents, and if identical blocks found, then they are removed. In Hash-based deduplication in which collision-resistant hash algorithms MD5, SHA are used to identify the redundant data.

Data Deduplication				
1. Data		2. Location		3. Implementation
a. File-Level		a. Client		a. Hash based
b. Block level	i. Fixed length	b. Target	i. Inline	b. Content based
	ii Variable length		ii. Post-process	

Figure 2. Classification of data deduplication techniques

Streaming analytics [22] is the analysis of streams of data while it is on the wire. The analysis consists of understanding data patterns, data classification, or detecting anomalies in the moving data streams. In conventional data analytics data is stored into a database and queries are applied to it, which is a time taking process. Whereas, streaming analytics is where algorithms and queries are applied live and within a short duration of time. Hence, its applications become prominent in the areas such as networking, Internet of things (IoT), cloud computing and big data. Akka streams are only for reactive streams, where a large volume of live data not handled easily. OutOfMemoryErrors slows down the producers when the consumer cannot keep up. One component loses the messages as it is challenging to handle the incoming messages by the component. This is called Backpressure. Akka streams match the speed between producers and consumer messages by avoiding backpressure. Akka streams handle the data movement without resource exhaustion, buffering and losses. Akka is an actor model framework to build fault-tolerant, concurrent, scalable and message-based applications. It is easy and powerful to develop multithreaded applications using this framework. On top of the Akka framework, a high-level API, Akka streams API built.

2. RELATED WORK

Many researchers used a pre-copy approach to live VM migration. Memory compression concept introduced to reduce the volume of the memory to be transferred to the target before initiating the migration process. Later on, the memory pages were decompressed at the destination, after, all memory pages arrived there. Lossless compression algorithm with minimum CPU overhead (compression and decompression time) should be chosen. The compression time directly affects the migration time. Various workloads on VMs had several similarities in data. Choosing one algorithm for compression was not enough for better reduction of similarities. Hence, different algorithms of compression were chosen and implemented according to the type

of pages. Wilson-Kaplan direct-mapped (WKdm) [23] algorithm and Lempel-Ziv-Oberhumer (LZO) were used the type of pages accordingly in characteristic-based compression (CBC) algorithm. Concurrent execution of compression and decompression tasks performed using multithreading. Some researchers implemented the above tasks using the multi-core concept to minimize the time taken for compression. Experiments carried out assuming various workloads and 27.1%, 32% and 68.8% reduction in downtime, migration time and total pages transferred achieved respectively.

Some researchers analysed that deduplication had the benefits of dividing the data into tiny chunk sizes and also associated with meta-data overhead. Even though compression not used, fixed-length deduplication with 4KB chunks yields most efficient storage usage and network resources. At this chunk size, storage requirements reduced by approximately 60% when compared Delta encoding techniques. In [24], block-level deduplication techniques are used to reduce the backup storage of images and achieved 87% of space-saving. Salted message digest5 (MD5) hash algorithm used to find the duplicates. The most significant number of unique hashes obtained when the chunk size is 8k. Extreme binning explains data deduplication in backup systems. In [25], Rabin fingerprints calculated using collision-resistant hash algorithms like MD5 and secure hash algorithm (SHA). These fingerprints used to identify the duplicated blocks based on file similarity instead of locality.

Chunk index is split into two tiers to achieve parallelism in the deduplication process. Using this process, throughput increased with low overhead. An optimized Rabin-Karp algorithm used in incremental modulo-K(INC-K) algorithm [26], which is modulo arithmetic in nature used for deduplication with a minimum number of arithmetic operations. Least recently used (LRU)-based index partitioning used for fingerprint storing and lookup. To maintain better chunking speed and deduplication ratio context-aware chunking used. 66% better deduplication ratio achieved on efficient chunking. In [27] several data deduplication techniques like file-based, block-based and byte-based introduced in post-copy, pre-copy and hybrid types of live VM migration to reduce the count of pages that were to be moved from the origin server to target host to achieve load balancing with minimum migration time.

Some researchers did reverse deduplication, RevDedup [28], a deduplication system for optimizing the latest backups reads of VM images. Unlike the traditional deduplication that removes duplicates from new data, RevDedup eliminates duplicates from old data, while keeping the structure of new data as sequential and thereby shifting fragmentation to old data. High deduplication efficiency, high read throughput and high backup throughput achieved by doing the experiments on the real world VM image snapshots of the users. In [29], a secure and efficient data deduplication technique for storing the data received from IoT sensors in the cloud. SecDedoop [30] in which a secure deduplication mechanism used for access control of bigdata over Hadoop Distributed File System. In adaptive deduplication [31], the virtual disk image files taken as input to the algorithm. If the file size was less than or equal to 1GB, a reasonable number of duplicates were found and eliminated using fixed-length block-level deduplication. If the VM disk image file greater than 1 GB, then the file was divided into chunks of size 1 GB each except the last one. For each chunk, that obtained here undergo fixed-length deduplication. The file undergoes fixed-length block deduplication with 4KB as the chunk size. In the first step, the method removes duplicates partially. Then the resultant files were given as input to the variable-length block deduplication, and much more duplicates identified and eliminated. By using this technique, the majority of duplicates found as the Rabin-Karp rolling hash algorithm used to determine the chunk boundaries. No researcher discusses the reunion of VM image pages at destination host and the time taken for the migrated VM to resume. Based on this motivation, the VM pages that migrated after adaptive deduplication reunited into a single disk image file and the time taken for the VM to resume discussed in this paper.

3. MOTIVATION

In the live migration process, there will be service disruption in less than tens of milliseconds: additional network and CPU resources consumed due to iterative tracking, scanning and transfer of VM pages. Hence there will be VM performance degradation that is being migrated. Hosts and network performance that is, all the hosts, VMs or other processes can be affected [32, 33]. This brownout [34] condition avoided by reducing the time that a VM spends in live migration. This condition achieved by eliminating the count of VM pages migrated during the live migration process. Many researchers suggest compression and deduplication techniques to minimize the size of VM pages. Data deduplication [35] is the best way to reduce the memory pages. At the target host, the deduplication process done in a reverse manner, and the migrated VM also resumed within a short time as the size of VM disk image is reduced with several deduplication techniques at the source before migration.

In cloud computing, live migration of VMs plays a vital role to achieve the goals like load balancing, server consolidation, fault tolerance and system maintenance. In this research work, the cloud environment was

built using the OpenStack [36]. The VMs with configuration 2GB of RAM and 10GB of the hard disk was created and launched with different operating systems. In glance component of the OpenStack, image registry maintains VDI, VHD, VMDK, QCOW2 images of instances as shown in Figure 3. These virtual disk images take larger memory and include many zero pages, similar pages and duplicate pages which are not required to transfer to the target host. It also takes more migration time to move these additional pages. Here, an adaptive deduplication algorithm is used to remove the extra pages by applying deduplication process twice. In adaptive deduplication, VM disk pages undergo both fixed-size and variable-size deduplication processes. In the variable-size deduplication process, content defined chunking [37] is done, and many duplicates are removed. Only the unique data transferred to the target host, which optimizes the total time for migration and downtime. The architecture shows how an adaptive deduplication algorithm performed. In adaptive deduplication algorithm, based on the size of the virtual disk image file the type of deduplication process applied. For smaller files (1GB or less), only fixed-length block-level deduplication process with 4KB chunk size applied as it was efficient on smaller files. For larger files (more than 1GB) the file was split into 1GB chunks except the last one. Each chunk undergoes both fixed and variable-length block-level deduplication processes, as explained in the previous sections.

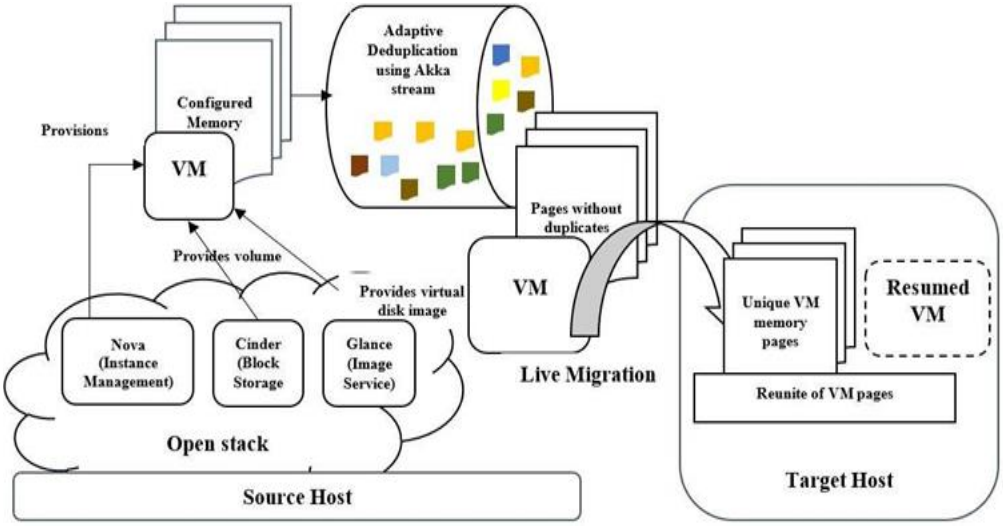

Figure 3. System architecture

The larger virtual disk file was split into multiple splits because, for a single large file, the execution time of the deduplication process was more. Many experiments with different sizes of file splits were carried out and based on the practical experience, the split size decided as 1GB. In adaptive deduplication, multi-threading concept was used in variable-length block-level deduplication as it is computationally expensive [38]. This process implemented using Akka stream with multi-threading to accelerate the deduplication process as content defined chunking (CDC) [39] was challenging to parallelize because of content dependency. In this work, identifying chunk boundaries and hash code computation by SHA-1[40] algorithm performed in parallel by using two threads. The hash code of two chunks compared and if it was unique, then only the chunk was stored in the hash table. The reduced VM disk image file migrated to the destination and after receiving all the unique chunks at target one thread is used to reunite all the pages received into a single virtual machine disk image file with less size. The VM is resumed back on the target within no time and is ready to run its applications. The reunion of deduplicated VM pages at target and resumption of VM implemented. The system architecture is as shown in Figure 3.

4. RESULTS AND DISCUSSIONS

Different formats of VM images such as VDI, QCOW2, VMDK, VHD, files took from OpenStack image registry. The VMs were created with a standard configuration of 2GB memory and 10GB hard disk. The Table 1 shows the size of several VM images of various operating systems such as CentOS, Ubuntu of different formats. This dataset collected from the real private cloud environment that was created by OpenStack with the mentioned. After adaptive deduplication, the deduplicated chunks of VM image disk image files

migrated to the destination host and all the deduplicated chunks are reunited into a single VM disk image file. As the chunks have given the unique reference number while doing the chunking process in adaptive deduplication, these reference numbers ensure the order of pages that are to be reunited. After reuniting all the deduplicated pages at the target host, the size of all the virtual disk image files noted and compared with their original sizes. The results are shown in Figure 4 and Figure 5. They clearly show that adaptive deduplication reduces the size of VM to be migrated in a better way. By using the proposed technique, 6.61% minimum for Ubuntu files and 95.5% maximum deduplication rate obtained for centos files.

To resume the migrated VM at the target host, not only the virtual disk image is required but also the metadata file is required. These are of small size and are migrated separately from source to destination host as the unique pages. These data files do not affect migration time. Virtual disk image file of each format is reunited along with their metadata files and resumed at the target host. The time taken to resume for each VM with reunited disk pages at the target host noted and compared with that of the original virtual disk image file. The results in Figure 6 shows the comparison of the total size of VMDK, VHD, VDI and QCOW2 formats of different operating systems such as CentOS7, CentOS6.9, Ubuntu12.04, Ubuntu 14.04, Ubuntu11.10 before deduplication (original size) and after reuniting the adaptive deduplicated pages. The Figure 6 also shows the time taken for the VMs of different formats to resume in the target host. It clearly shows that as the size of the VM image reduces the resumption time for a VM after migration on target host also reduced. Figure 7 represents the comparison of the deduplication ratio of adaptive deduplication and the percentage of resumption time of all virtual disk images with different formats. It clearly shows that the better the deduplication ratio, the less the time taken for a VM to resume on the target host. For a deduplication ratio of 92%, it takes an overall time of 3.52 minutes, which is a 7% reduction in resumption time when compared with the time taken for the total QCOW2 files with original size. For VMDK files the resumption time is reduced by a maximum 17% (7.63 mins) when compared with that of for original files.

Table 1. Dataset of virtual disk images in bytes4

Image Format	CentOS7	CentOS6.9	Ubuntu12.04	Ubuntu14.04	Ubuntu 11.10
VMDK	1242859	446922	3511238307	4882761848	1831810
VHD	146448	24576	3511646874	149467	477556
VDI	2322209	2216387	3412228224	4856112381	2423744118
QCOW2	161368	131729	3462770688	4855561190	498630
Total (in bytes)	3872884	2819614	13897884093	14594584886	2426552114

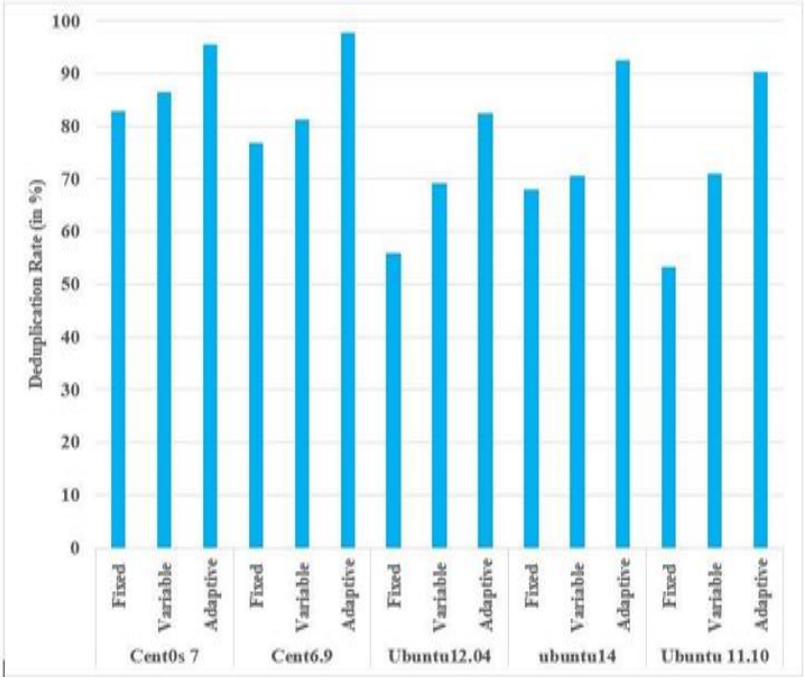

Figure 4. Fixed length vs variable length vs adaptive deduplication technique

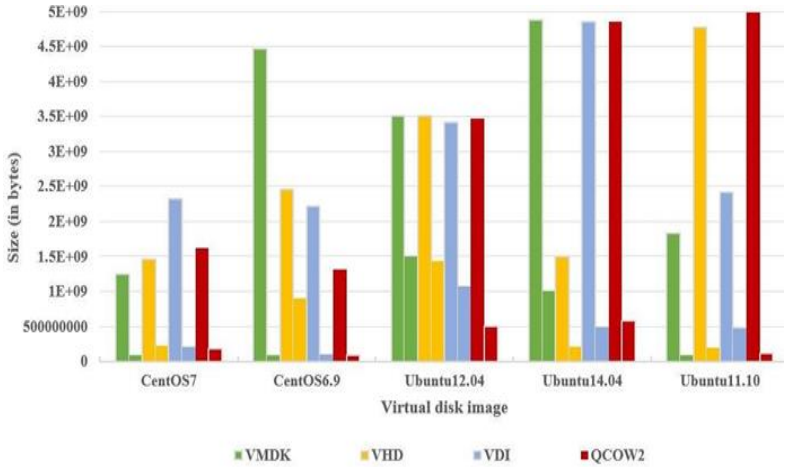

Figure 5. Original size vs after reunion of the deduplicated pages

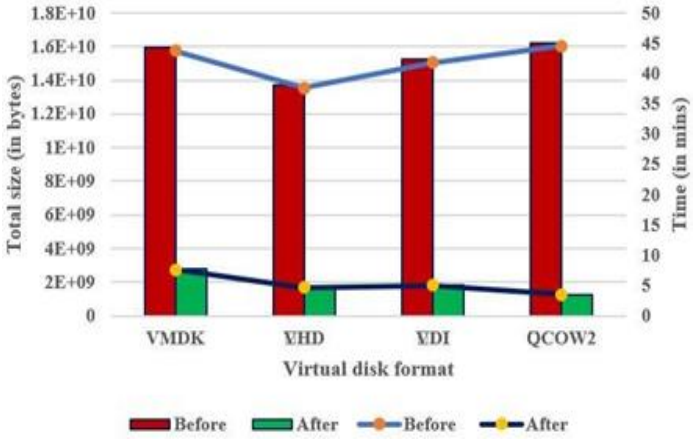

Figure 6. Total size vs resumption time

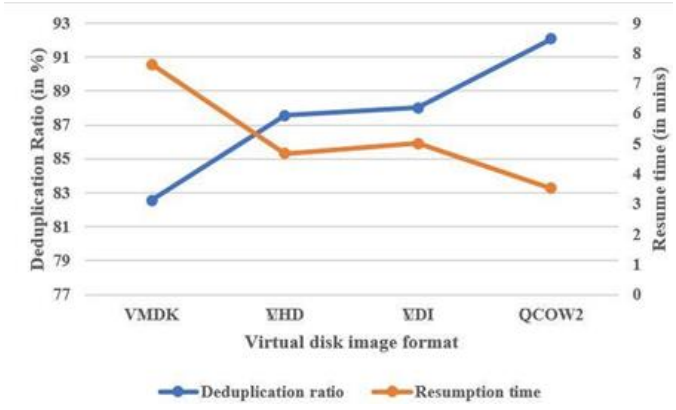

Figure 7. Deduplication ratio vs resumption time

5. CONCLUSION

In this paper, the results show that adaptive deduplication is an efficient deduplication technique to reduce the virtual disk image file when compared to the existing fixed length and variable length deduplication techniques. It shows that 83% of reduction in overall storage achieved. The migrated deduplicated chunks after adaptive deduplication then reunited into a single virtual disk image file at the target host. The metafiles also

migrated to the target host. The VM with reunited virtual disk image file along with metadata files resumed at target host and the time noted. This time compared with the VM resumption time with the original disk image file. The results show that the better the deduplication ratio, the less time taken for a VM to resume on the target host. For a deduplication ratio of 92%, it takes an overall time of 3.52 minutes, which is a 7% reduction in resumption time when compared with the time taken for the total QCOW2 files with original size. For VMDK files the resumption time is reduced by a maximum 17% (7.63 mins) when compared with that of with original files.

ABBREVIATIONS

VM: Virtual machine; LZO: Lempelzivoberhume; NAS: Network attached storage; VMM: Virtual machine migration;

REFERENCES

[1] T. Y. J. N. Malleswari, et al., "A survey of cloud computing architecture and services provided by various cloud service providers," *Proceedings of the International conference on demand computing*, 2012.
[2] A. S. A. Alhammadi, et al., "Survey study of virtual machine migration techniques in cloud computing," *International Journal of Computer Applications*, vol. 177, no. 2, pp. 18-22, 2017.
[3] F. Zhang, et al., "A Survey on Virtual Machine Migration: Challenges, Techniques, and Open Issues," *IEEE Communications Surveys and Tutorials* vol. 20, no. 2, pp. 1206-1243, 2018.
[4] M. H. Shirvani, et al., "A survey study on virtual machine migration and server consolidation techniques in dvfs-enabled cloud datacenter: Taxonomy and challenges," *Journal of King Saud University Computer and Information Sciences*, vol. 32, no. 3, pp. 267-286, 2020.
[5] T. Y. J. N. Malleswari and G. Vadivu, "Deduplication of VM Memory Pages using MapReduce in Live Migration," *ARPN Journal of Engineering and Applied Sciences*, vol. 12, no. 3, pp. 1890-1894, 2017.
[6] T. Y. J. N. Malleswari and G. Vadivu, "Implementation and Comparison of various VM Allocation Strategies in Live Virtual Machine Migration using CloudSim," *International Journal of Pure and Applied Mathematics*, vol. 120, no. 6, pp. 1-16, 2018.
[7] T. Y. J. N. Malleswari, et al., "Live virtual machine migration techniques-A Technical Survey," *Intelligent Computing Communication and Devices*, pp. 308-319, 2014.
[8] A. Choudhary, et al., "A critical survey of live virtual machine migration techniques," *Journal of Cloud Computing*, vol. 6, 2017.
[9] H. Alshahrani, et al., "Live migration of virtual machine in cloud: Survey of issues and solutions," in *Proceedings of the International Conference Security and Management*, pp. 280-285, 2016.
[10] Y. Cui, et al., "Traffic-aware virtual machine migration in topology-adaptive dcn," *IEEE/ACM Transactions on Networking*, vol. 25, no. 6, pp. 3427-3440, 2017.
[11] M. R. Anala, et al., "Application performance analysis during live migration of virtual machines," *Proceedings of the 2013 3rd IEEE International Advance Computing Conference (IACC)*, pp. 366-372, 2013.
[12] S. Rathod and Reddy V., "Secure Live VM Migration in Cloud Computing: A Survey," *International Journal of Computer Applications (IJCA)*, vol. 103, pp. 18-22, 2014.
[13] B. Das, et al., "Improving Total Migration Time in Live Virtual Machine Migration," *Proceedings of the Sixth International Conference on Computer and Communication Technology*, pp. 57-61, 2015.
[14] X. Zhang, et al., "Exploiting Data Deduplication to Accelerate Live Virtual Machine Migration," *Proceedings of the 2010 IEEE International Conference on Cluster Computing*, pp. 88-96, 2010.
[15] W. Hu, et al., "A Quantitative Study of Virtual Machine Live Migration," *Proceedings of the 2013 ACM Cloud and Autonomic Computing Conference*, Miami, Florida, USA, pp. 1-10, 2013.
[16] A. Sharma, et al., "A survey on live virtual machine migration," in *Proceedings of the 2017 UKSim-AMSS 19th International Conference on Computer Modelling Simulation (UKSim)*, pp. 187-192, 2017.
[17] G. Singh and P. Gupta, "A review on migration techniques and challenges in live virtual machine migration," *5th International Conference on Reliability, Infocom Technologies and Optimization (Trends and Future Directions)*, pp. 542-546, 2016.
[18] U. Deshpande, et al., "Gang Migration of Virtual Machines Using Cluster-wide Deduplication," *Proceedings of the 2013 13th IEEE/ACM Int. Symposium on Cluster, Cloud and Grid Computing*, pp. 394-40, 2013.
[19] J. Song, et al., "TSMC: A Novel Approach for Live Virtual Machine Migration," *Journal of Applied Mathematics*, vol. 2014, pp. 1-7, 2014.
[20] K. Jin and E. L. Miller, "The Effectiveness of Deduplication on Virtual Machine Disk Images," *Proceedings of SYSTOR 2009: The Israeli Experimental Systems Conference, SYSTOR '09*, Haifa, Israel, pp. 1-12, 2009.
[21] T. Y. J. N. Malleswari, et al., "Deduplication Techniques: A Technical Survey," *Int. Journal for Innovative Research in Science and Technology*, vol. 1, no. 7, pp. 318-325, 2014.
[22] K. N. Abhishika, et al., "Streaming Analytics," *International Journal of Pure and Applied Mathematics*, vol. 118, no. 22, pp. 525-532, 2018.
[23] P. R. Wilson, et al., "The Case for Compressed Caching in Virtual Memory Systems," *Proceedings of the Annual Conference on USENIX, USENIX Association*, Berkeley, CA, USA, 1999.

[24] D. T. Meyer and W. J. Bolosky, "A Study of Practical Deduplication," *Transactions on Storage*, vol. 7, no. 4, pp. 1-20, 2012.
[25] D. Bhagwat, et al., "Extreme Binning: Scalable, parallel deduplication for chunk-based file backup," *IEEE International Symposium on Modeling, Analysis Simulation of Computer and Telecommunication Systems*, pp. 1-9, 2009.
[26] J. Min, et al., "Efficient Deduplication Techniques for Modern Backup Operation," *IEEE Transactions on Computers*, vol. 60, no. 6, pp. 824-840, 2011.
[27] K. A. Kumari, et al., "On Speeding up Virtual Ma- chine Migration through Integrated Data De-Duplication Methods," *Research Journal of Applied Sciences*, vol. 12, no. 2, pp. 131-138, 2017.
[28] C. H. Ng and P. Lee, "RevDedup: A Reverse Deduplication Storage System Optimized for Reads to Latest Backups," *The 4th ACM SIGOPS Asia-Pacific Workshop on Systems (APSYS 2013)*, Singapore, 2013.
[29] Y. Gao, et al., "Secure data deduplication for Internet-of-things sensor networks based on threshold dynamic adjustment," *International Journal of Distributed Sensor Networks*, 2020.
[30] P. Ramya and C. Sundar, "SecDedoop: Secure Deduplication with Access Control of Big Data in the HDFS/Hadoop Environment," *Big Data*, vol. 8, no. 2, 2020.
[31] T. Y. J. N. Malleswari and G. Vadivu, "Adaptive deduplication of virtual machine images using AKKA stream to accelerate live migration process in cloud environment," *Journal of Cloud Computing*, vol. 8, 2019.
[32] M. Noshy, et al., "Optimization of live virtual machine migration in cloud computing: A survey and future directions," *Journal of Network and Computer Applications*, vol. 110, pp. 1-10, 2018.
[33] S. Kumar and D. Malhotra, "Live Virtual Machine Migration Techniques, Survey and Research Challenges," *International Journal of Computer Sciences and Engineering*, vol. 06, no. 5, pp. 50-53, 2018.
[34] K, A. K., K, R. S. R., and JKR, S, "On speeding up virtual machine migration through integrated data de-duplication methods," *Research Journal of Applied Sciences*, vol. 12, no. 2, pp. 131-138, 2017.
[35] X. Zheng, et al., "A cloud data deduplication scheme based on certificateless proxy re-encryption," *Journal of Systems Architecture*, vol. 102, p. 101666, 2020.
[36] J. Zhang, et al., "IM-Dedup: an image management system based on deduplication applied in DWSNs," *International journal of distributed sensor networks*, 2016.
[37] T. Gholami T., et al., "CA-Dedupe: content-aware deduplication in SSDs," *Journal of Supercomputing*, 2020.
[38] G. Sun, et al., "A New Technique for Efficient Live Migration of Multiple Virtual Machines," *Future Generation Computer Systems*, vol. 55, pp. 74-86, 2016.
[39] W. Xia, et al., "FastCDC: A Fast and Efficient Content-Defined Chunking Approach for Data Deduplication," *USENIX Annual Techology Conference USENIX ATC'16*, pp. 101-114, 2016.
[40] R. N. Widodo, et al., "A new content-defined chunking algorithm for data deduplication in cloud storage," *Future Generation Computer Systems*, vol. 71, pp. 145-156, 2017.

BIOGRAPHIES OF AUTHORS

T.Y.J. Naga Malleswari pursued doctorate in SRM Institute of Science and Technology and faculty member in the Department of Computer Science & Engineering, Faculty of Engineering & Technology, SRMIST, Chennai, India. Research interests include Cloud Computing, BigData, IOT, and Algorithms.

Mr. T. Senthilkumar completed Ph.D in the field of wireless communication, under the guidance of Dr.S. Prabakaran, Professor, Department of Computer Science & Engineering. He has contributed many scientific research papers.

Jothi Kumar C pursued doctorate degree in the Department of Computer Science and Engineering at SRM Institute of Science and Technology, Kattankulathur, Tamil Nadu, India. His research interests include Wireless Sensor Networks, Adhoc Networks, Mobile Adhoc Networks, Distributed Systems, etc.