VOL. 12, NO. 6, MARCH 2017 ISSN 1819-6608
ARPN Journal of Engineering and Applied Sciences

www.arpnjournals.com

DEDUPLICATION OF VM MEMORY PAGES USING MAPREDUCE IN LIVE MIGRATION

TYJ Naga Malleswari[1] and Vadivu G[2]
[1]Department of CSE, Sri Ramaswami Memorial University, Chennai, India
[2]Department of Information Technology, Sri Ramaswami Memorial University, Chennai, India
E-Mail: ramuyamu@gmail.com

ABSTRACT

Virtual Machine (VM) migration is one of the main features of virtualization. Fault tolerance, load balancing, power management and system maintenance are the great advantages of VM migration. In Live VM Migration, pre-copy approach algorithm is widely used where the memory pages are iteratively copied to the destination host from source host without disconnecting the VM being migrated. But a lot of duplicate memory pages are transferred which takes longer migration time and down time. This paper proposes an algorithm DedupMR, (Deduplication using MapReduce in Live Migration) which performs deduplication of memory pages that are being migrated in parallel using Map and Reduce phases in many iterations of pre-copy approach. In DedupMR memory pages are chunked into fixed size and identification of similar chunks done by calculating fingerprints. Then compressed deduplicated pages transferred to destination VM. The entire process is parallelized by using MapReduce. Due to this deduplication done in short time thus achieves less migration time and down time. MapReduce is a technique that can be customized to process the independent data in parallel. In this paper duplicated pages are reduced by maximum of 29% and minimum of 7%. This can be done in parallel by MapReduce with minimum time.

Keywords: virtualization, deduplication, live migration, chunking, rabin karp rolling hash, map reduce.

1. INTRODUCTION

Due to the rapid growth of cloud computing Virtual Machines (VMs) [1, 2] are created in cloud data centers through virtualization [3] for resource management with minimal effort. The same physical infrastructure shared by all the applications running on VM. To achieve consistency, reliability, for disaster recovery data is duplicated by running VMs. Due to this storage is a challenge and with the increase of cloud data centers there are millions of VMs being created by which load balancing is a critical issue. Migration service which is provided in virtualization is good to resolve these types of challenges. There are two methods in the migration process. They are Non-live migration and live migration. Migrating a powered off virtual machine from one host to another is called non-live migration. In this method, VM status is lost, and service is interrupted to the user [15] and is a drawback. Transferring a running virtual machine from one physical host to another is known as live migration [1]. Through this process load balancing, power management, system maintenance, and fault tolerance objectives are achieved successfully [4]. In live migration, virtual CPU context, storage, network connectivity, memory image are moved from source host to destination host which collectively a large size.

In Live Migration pre-copy approach is widely used where downtime is less. In this approach upon receiving the request of migration from the hypervisor the destination host is checked for the resource availability. If available then the all the memory pages of source VM are transferred to destination. In the mean while some pages may be dirtied. Again the dirtied pages are transferred to destination. This process is continued in iterations until the dirty page rate is less than the transferring rate. Then VM on source host is suspended and further traffic is redirected to VM on destination host once it is resumed. The flow chart of this process is given in Figure-1.

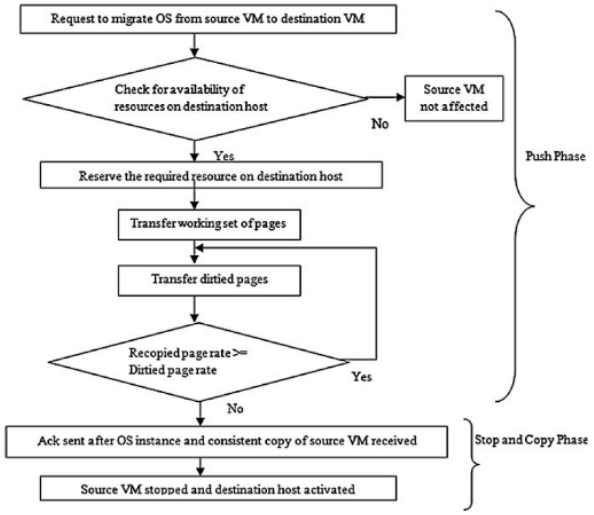

Figure-1. Pre-copy approach [1].

The main metrics of migration are Migration time: the total time between the beginning of migration process in source host and the running of migrated VM starts in destination host. Pages transferred: the amount of memory pages transferred during the migration process. Downtime is the amount of time the migrated VM services are not available. In migration process, transferring the VM status takes more bandwidth and consumes more CPU resources which lead more migration time and down time by which the purpose of live migration not achieved. So,

the no. of pages moved should be minimized. There are two ways to reduce the size of memory or storage. One is compression where the redundant bit is identified and eliminated. Earlier, researchers used compression [5] to lessen the size of data to be transferred by removing duplicate memory pages. Though compression reduces the size, it also compresses the redundant pages. Compression algorithm with high compression ratio takes more time to compress and decompress. Due to this overhead migration performance is not improved.

The other one is Deduplication [6] where the duplicated data is identified by finding the hash code called fingerprint using collision resistant hash algorithms like SHA-1[7], MD5[8] and eliminated by storing the pointer to the data. There are two methods of Deduplication.

a) File level deduplication

In this process, files have the same fingerprint are identical and are detected. Fingerprint, original file and links to the references of the original file are stored. As shown in Figure-2 only one copy is maintained in the storage. Thus space is used efficiently. However, in the original file if minor modifications then this method saves the small amount of space.

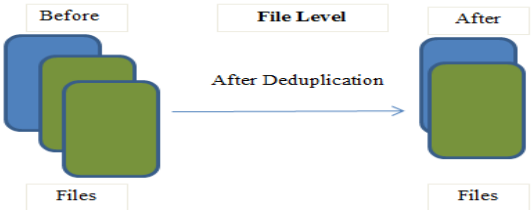

Figure-2. File level deduplication.

b) Block level deduplication

The file was broken into chunks called blocks. For these blocks, a hash algorithm is applied, and duplicate blocks identified. Unique blocks stored and the pointers used for duplicate blocks. Based on block size, there are two types of chunking process. Fixed size chunking, 4KB, 8KB, 16 KB, etc., are the fixed sizes of several blocks. In Variable size chunking, the chunk size is based on content. For example chunk boundaries are identified by Rabin-Karp rolling hash algorithm [9].

Based on the location where deduplication is done there are two types. Client based and Target based. In the former Deduplication is done at client side and the unique data sent whereas in the latter deduplication is done at target side, and individual files written to the disk.

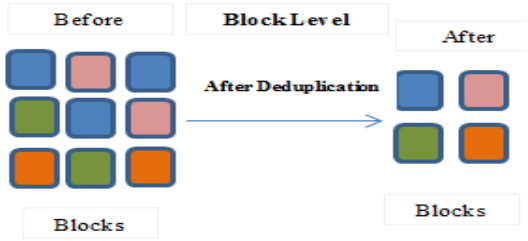

Figure-3. Block-level deduplication.

2. RELATED WORK

As Virtual machine disk images need more storage in physical servers, fixed, variable size chunking strategies are used to chunk the files, zero filled blocks [10] are identified, and deduplication is done. It has been observed that fixed size chunking is better and have the same deduplication ratio as the variable size chunking. At the beginning phase, few VM disk images took more storage, and later on, virtual disk images of different operating system versions had many redundant data and eliminated which occupies less storage. In Live virtual machine migration, from the source host to destination host many duplicated memory data image is transferred which takes more migration and downtime. To avoid this uses MDD, migration with data deduplication where during migration process similar memory pages are identified using fingerprints generated by the hash algorithm and run length encode used to eliminate duplicate memory pages. As virtual machine images need more storage in cloud data centers, LiveDFS [11], is a deduplication file system where with the limited number of fingerprints deduplicated VM images are stored. 40% of storage saved when compared to an ordinary file system. [12] Used deduplication at two levels as backing up of VM snapshots take much storage in the cloud and achieves high deduplication ratio with limited resources. A copy of VM's image file is maintained in DFS at physical host. Consistency ensured when the log of concurrent disk writes done during the backup of a snapshot. In the first level, deduplication is performed as many data not modified during the period of each backup. The second level deduplication is because of more usage of libraries and software. Liquid [13] proposes, a file system with deduplication where the deduplicated data blocks stored in data servers where the meta server stores the fingerprints of data blocks which can be referred when a data block is accessed. VM-centric backup scheme [14] proposes where all blocks are not treated as equivalent but consider the boundaries of VM by deduplication algorithm for backing up of VM snapshots. Popular chunks shared across the VMs. VM-centric fault isolation techniques used for not affecting the files that share the big block. Resource usage, deduplication efficiency, and fault tolerance were evaluated. A practical study on deduplication [9] considering file level and block level redundancy elimination for storage of back up images and live file systems and achieved 87% of savings.

3. ARCHITECTURE

The proposed work is to accelerate live VM migration by introducing data deduplication into the pre-copy algorithm. Figure-3 shows the steps involved in the proposed work, at the source host the migrated VM's (VM3) memory pages are to be transferred to target host. In pre copy approach modified pages are transferred in iterations. Many duplicate pages are transferred which consume more time for migration.

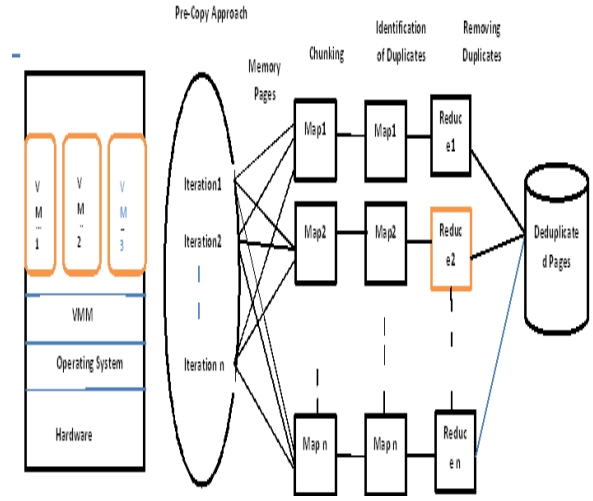

Figure-4. Deduplication of migrated VM (vm3) memory pages using MapReduce at source host.

Hence in the proposed work the memory pages from each iteration are deduplicated by DedupMR algorithm. This algorithm have may two map phases where chunking of memory pages and identification of duplicate pairs are identified. It also contains one reduce phase where all the duplicate pairs are removed. Finally the deduplicated memory pages are further compressed and migrated to the target host. The migrated machine (VM3) will resume working there. As the deduplication process is parallelize using map reduce [18] it takes no longer time which has a direct impact on migration time, down time.

4. ALGORITHM

DedupMR (Memory Pages of Migrated VM)
Step 1: Deduplication using MapReduce
Step 1.1: First Phase
Map tasks for Fixed size Chunking
Step 1.1.1: //Fixed Size Chunking
If (Sizeof(Memory Pages) > Threshold)
Step 1.1.2: Choose less chunk size.
else
Step 1.1.3 Choose more chunk size.
Step 1.2: Second Phase
Map Tasks to find Hash code using MD5
Step 1.2.1 If (hashcode is same for different chunks)
Identify it as duplicate.
Step 1.2.2: Construct Hashtable with chunk name and its reference pointer.
Step 1.3: Reduce Phase
Remove duplicates and combine all chunks.
Step 2: Compression of deduplicated pages.

Figure-5. Algorithm for deduplication of migrated VM memory pages using Map Reduce.

5. DISCUSSIONS

Most of the VM memory pages are disk image information hence disk images files are collected from several VMs. Fixed size, variable size chunking techniques are performed on these images in deduplication process using java application. In Fixed size chunking the .vmdk files of Windows, and Linux, are chunked as 1KB, 4KB, 8 KB, 16KB, 32 KB, 64KB. Ubuntu file is chunked with sizes 1 MB,4MB, 8MB, 16MB, 32MB, 64MB, 128MB, 512 MB. Based on the total size of the file the chunk size can be in kilobytes or megabytes and powers of 2. Figures 5 and 7 illustrates the size reduced when the chunk size varies. As the chunk size is reduced, the size of the file to be transferred during migration is reduced. Figures 6 and 7 demonstrates that the deduplication ratio is reduced when the chunk size is varied, which means that the number of pages transferred is reduced. Deduplication ratio is defined as the proportion of deduplicated data size to the total size before deduplication.

$$\text{Deduplication Ratio} = \frac{\text{Size of file after Data Deduplication}}{\text{Size of file before Data Deduplication}}$$

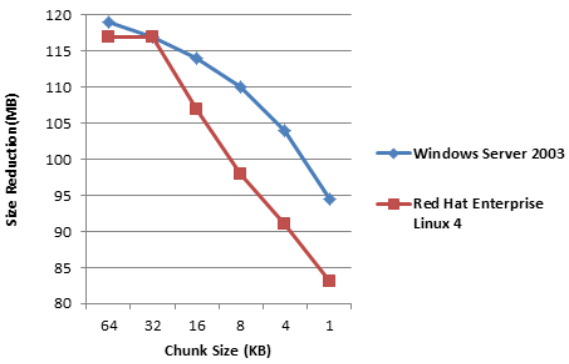

Figure-6. Chunk size vs size reduction in fixed size chunking.

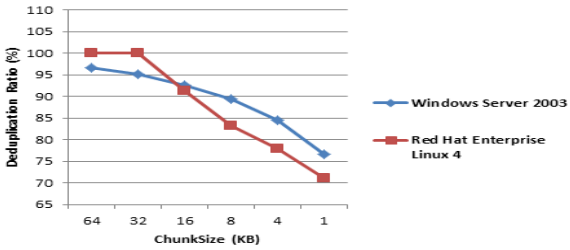

Figure-7. Chunk size vs deduplication ratio (in %) in fixed size chunking.

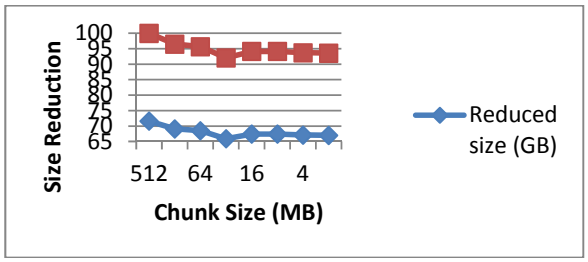

Figure-8. Chunk size vs size reduction and deduplication ratio (in %) in fixed size chunking for Ubuntu vmdk file.

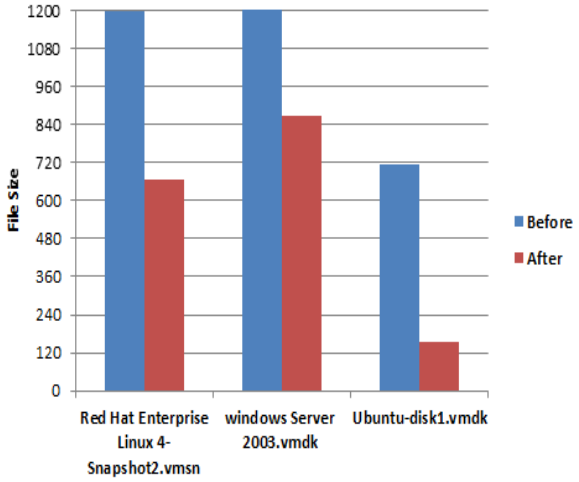

Figure-9. Comparison of file size before and after variable size chunking.

Variable size chunking (Rabin-Karp Rolling hash) also performed on the VM image files. The file size of VM image files before and after data deduplication using variable size chunking is compared and shown in Figure-8.

In Figure-9 the minimum deduplication ratio achieved for each VM file in fixed size chunking is compared with the deduplication ratio of the same files in variable size chunking. It shows that by using variable size chunking the better deduplication ratio is achieved. For larger size VM files it is much better. The results show that the deduplication ratios of Linux, Windows, and Ubuntu VM disk image files are reduced by 16%, 6% and 70% by using variable size chunking, and it is time-consuming for identifying similar chunks. But Map Reduce does n't applicable for variable size chunking because of interdependency in the process of identifying the chunk boundaries. Thus by using fixed size chunking the number of memory pages transferred during live migration process is minimized. Map Reduce parallelizes the deduplication process and also reduces the time taken for deduplication. This minimizes the overall migration time. The downtime is also reduced with minimum performance degradation.

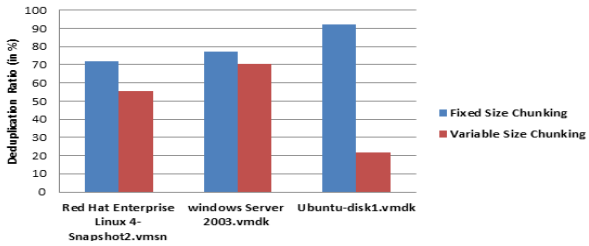

Figure-10. Comparison of deduplication ratio.

6. CONCLUSIONS

The main contribution of the work is the proposed algorithm in which the idea of MapReduce is introduced to parallelize the deduplication process for memory pages that are transferred during live VM migration. The primary objective of the proposed algorithm is to reduce the number of memory pages transferred, migration time and down time. Many map tasks in two phases work parallel for chunking and identifying duplicates. Reduce function in the last phase remove the duplicates and give deduplicated pages which are ready for migration. As a part of proposed work deduplication of .vmdk file is implemented as these pages are the major part of migrated VM. .vmdk file is obtained by using ovf tool which is used by the command used to migrate the VM in the open stack cloud environment. In this paper percentage of size reduction of memory pages, if fixed size chunking with different chunk sizes is used and variable size chunking are compared. The comparison of deduplication ratio for various chunking techniques that involved in deduplication process is presented. The results show that as the size of each chunk decreases the number of duplicates increased which further reduces the whole size and increases deduplication ratio.

It is observed that the variable size chunking give better deduplication ratio over fixed size chunking. Fixed size chunking with less chunk size generates more duplicates and hence storage space required for the hash table is more and is a limitation. As in variable size chunking the memory pages are dependent on identifying the similar chunks and to decide chunk boundaries. Map Reduce is not used for dependent data. Hence in the proposed work fixed size chunking is used in deduplication process using MapReduce to accelerate live migration process.

REFERENCES

[1] NagaMalleswari TYJ, Vadivu.G, Malathi D. 2014. Live Virtual Machine Migration Techniques-a Technical Survey. Advances in Intelligent Systems and Computing. 308, 303-319.

[2] James E.Smith, Ravi Nair. 2005. The Architecture of Virtual Machines, IEEE Computer Society. 38(5): 32-38.

[3] Naga Malleswari TYJ, D. Rajeswari, Dr. V. Jawahar Senthil Kumar. 2012. A Survey of Cloud Computing, Architecture & Services Provided by Various Cloud Service Providers, In Proc. of Int. Conf. On Demand Computing.

[4] Raja Wasim Ahmad, Abdullah Gani, Siti Hafizah, Hamid, Muhammad Shiraz, Abdullah Yousafzai, Feng Xia. 2015. A Survey on virtual machine migration and server consolidation frameworks for cloud data centers. J. of Network and Computer Applications. 52, 11-25.

[5] Hai Jin, Li Deng, Song Wu, Xuanhua Shi, Hanhua Chen, Xiaodong Pan.: MECOM: Live migration of virtual machines by adaptively compressing memory pages. Future Generation Computer Systems, *ACM*, 2014, 38, 23-35.

[6] NagaMalleswari TYJ, Malathi D, Vadivu.G. 2014. Deduplication Techniques: A Technical Survey. International Journal of Innovative Research in Science & Technology. 1(7): 318-325.

[7] Eastlake, 3rd, Us Secure Hash Algorithm 1(sha1), 2001. http://tools.ietf.org/html/rfc3174.

[8] R. Rivest. 1992. The md5 Message-Digest Algorithm. http://tools.ietf.org/html/rfc1321.

[9] Dutch T. Meyer, William J. Bolosky. 2011. A Study of Practical Deduplication, In Proc. USENIX. Conf. On File and Storage Technologies, ACM. 1-1.

[10] Keren Jin, Ethan L.Miller. 2009. The Effectiveness of Deduplication on Virtual Machine Disk Images. In Proc. of SYSTOR 2009: The Israeli Experimental Systems Conference, ACM. Article 7, doi: 10.1145/1534530.1534540.

[11] Chun-Ho Ng, Mingcao Ma, Tsz-Yeung Wong, Patrick P. C. Lee, and John C. S. Lui. 2011. Live Deduplication Storage of Virtual Machine Images in an Open-Source Cloud, In Proc. Int. Conf. On Middleware. 81-100, http://www.cse.cuhk.edu.hk/~cslui/PUBLICATION/middleware11.pdf.

[12] Wei Zhang, Hong Tang, Hao Jiang, Tao Yang, Xiaogang Li, Yue Zeng. 2012. Multi-level Selective level Deduplication for VM Snapshots in Cloud Storage. In Proc. of Int. Conf. on cloud computing. IEEE. 550-557.

[13] Xun Zhao, Yang Zhang, Yongwei Wu; Kang Chen; Jinlei Jiang; Keqin Li. 2014. Liquid: A Scalable Deduplication File System for Virtual Machine Images. IEEE Trans. on Parallel and Distributed Systems. 25(5): 1257-1266.

[14] Zhang, 2015] Wei Zhang, Daniel Agun. Tao Yang, Rich Wolski, Hong Tang. 2015. VM-centric snapshot deduplication for cloud data backup. 31st Symposium on Mass Storage Systems and Technologies (MSST), IEEE. 1-12.

[15] Divya Kapil, Emmanuel S. Pilli, and Ramesh C.Joshi. 2013. Live Virtual Machine Migration Techniques: Survey and Research Challenges. IEEE. 2013. in Proc. Int. Conf on Advance Computing Conference, IEEE, DOI: 10.1109/IAdCC.2013.6514357.

[16] Anala M R, Manjunath Kashyap, Shobha G. 2013. Application performance Analysis during live migration of virtual machines 978-1-4673-4529-3/12. IEEE. 2013. In Proc. Int. Conf on Advance Computing Conference, IEEE, http://ieeexplore.ieee.org/document/6514252.

[17] Meenu Chawla. 2013. A Technical Review for Efficient Virtual Machine Migration, In Proc. of Int. Conf. on Cloud & Ubiquitous Computing & Emerging Technologies. 20 - 25, DOI: 10.1109/CUBE.2013.14

[18] Naga Malleswari TYJ, Vadivu. G. 2016. MapReduce: A Technical Review. Indian Journal of Science & Technology. 9(1).