TYJ and G *Journal of Cloud Computing: Advances, Systems and Applications* (2019) 8:3
https://doi.org/10.1186/s13677-019-0125-z

Journal of Cloud Computing:
Advances, Systems and Applications

RESEARCH Open Access

Adaptive deduplication of virtual machine images using AKKA stream to accelerate live migration process in cloud environment

Naga Malleswari TYJ[1] and Vadivu G[2*]

Abstract

Cloud Computing is a paradigm which provides resources to users from its pool based on demand to satisfy their requirements. During this process, many servers are overloaded and underloaded in the cloud environment. Thus, power consumption and load balancing are the major problems and are resolved by live virtual machine (VM) migration. Load balancing is addressed by moving virtual machines from overloaded host to under loaded host and from under loaded host to any other host which is not overloaded called VM migration. If this process is done without power off (Live) the virtual machines then it is called live VM migration. By this process, the issue of power consumption by physical hosts is also resolved. Migrating virtual machines involves virtualized components like storage disks, memory, CPU and networking, the entire state of VM is captured as a collection of data files. These data files are virtual disk files, configuration files, and log files. The virtual disk files take larger memory and take more migration time and hence the downtime. These disk files include many zero pages, similar and redundant pages. Many techniques such as compression, deduplication is used reduce the size of VM disk image file. Compression techniques are not widely used, due to the disadvantage of compression ratio and decompression time. Many researchers hence used deduplication techniques for reducing the VM disk image file in the live migration process. The significance of the research work is to design an adaptive deduplication mechanism for reducing VM disk image file size by performing fixed length and variable length block-level deduplication processes. The Rabin-Karp rolling hash algorithm is used in variable length block-level deduplication. Akka stream is used for streaming the VM disk image files as it is the bulk volume of live data transfer. To reduce the time of the deduplication process, many researchers used multithreading and multi-core technologies. We use multithreading in Akka framework to run the deduplication process concurrently without OutofMemory errors. The experiment results show that we achieved a maximum of 83% overall reduction in image storage space and 89.76% reduction in total migration time are achieved by adaptive deduplication method. 3% improvement in deduplication rate when compared with the existing image management system. The results are significant because when we apply this in the storage of data centres, there are much space savings. The reduction in size is dependent on the dataset was taken and the applications running on the VM.

Keywords: Virtualization, Cloud computing, Pre-copy technique, Virtual machine migration, Post copy technique, Chunking strategies, Streaming analytics, Akka stream

* Correspondence: vadivu.g@ktr.srmuniv.ac.in
[2]Department of Information Technology, SRMIST, Chennai 603203, India
Full list of author information is available at the end of the article

Introduction

Cloud computing [1] evolved with the advancement of several technologies such as virtualization (hardware), service-oriented architecture (internet), grid computing (grids), utility computing, (business model) and autonomic computing (systems management). It allows us to access huge amounts of computing power by aggregating resources in a fully virtualized manner. By virtualization technology, all the computer system's resources (Memory, CPUs, I/O devices) are virtualized to improve usage and sharing of computer systems and overcome the issues of organizational infrastructure maintenance. Hypervisor or virtual machine monitor (VMM) is a software which creates, manages virtual machines (guest) on a physical computer (or host). Several operating systems run on different guests as shown in Fig. 1(a).

Figure 1(b) shows different types of hardware virtualization. In hardware-assisted virtualization the hypervisor is supported by the hardware without modifying the guest software. In full virtualization, all guest operating systems are abstracted from the hardware and are communicated with VMM using binary translation. Paravirtualization [2] is where the guest OS is modified, and hyper calls are issued to the host OS instead of issuing to hardware.

Virtualization provides and manages the data centre's infrastructure dynamically. By multiplexing many virtual machines (VMs) on a single physical host, the utilization of resources is improved. Based on demand, these VMs scale up and down. Hence some hosts are heavily loaded, and some are under-utilized. So, effective virtual machine management is critical. A strong tool for managing virtual machines is VM Migration. Load balancing, power management, fault tolerance, and system maintenance are the goals of VM migration.

Virtual Machine Migration

The process by which a virtual machine is moved from one storage location or physical host to another physical

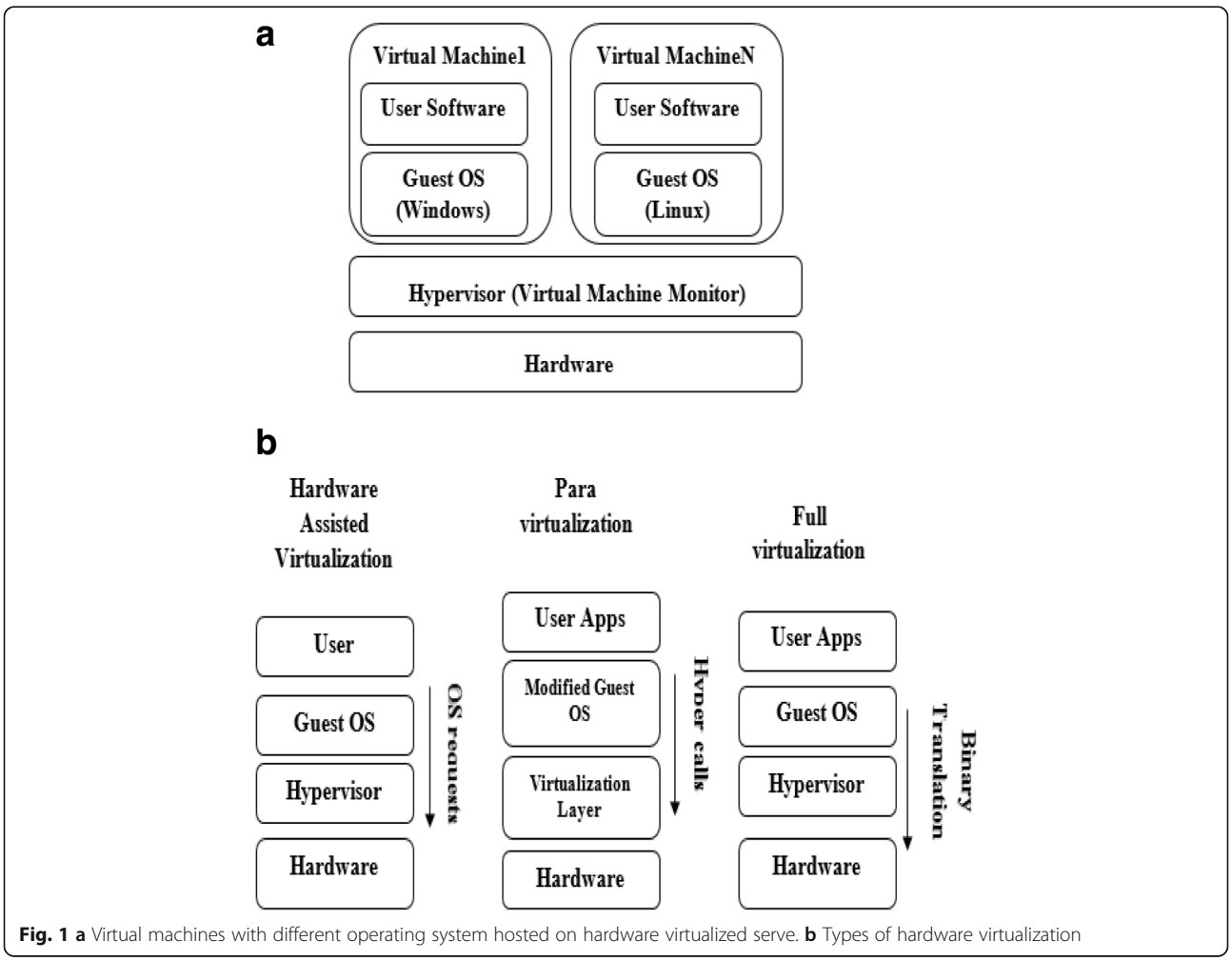

Fig. 1 **a** Virtual machines with different operating system hosted on hardware virtualized serve. **b** Types of hardware virtualization

host is called VM migration [3]. There are mainly two types of VM migration [4]. They are:

a. Non-live migration or cold migration

In this process, VM is switched off, and the entire VM state is transferred to the target host. There are more downtime and service interruption in this method. VM status lost is the major drawback of this technique.

b. Live migration or hot migration

The process of moving the VM from one physical machine to another without switching off the VM is called live migration. There are two methods to live migration.

i) Post copy approach

Post copy approach consists of stop and copy phase where VM is stopped, and the processor state is transferred. Further, based on demand, memory pages are moved to the destination which is called the pull phase as in Fig. 2(a).

ii) Pre-copy approach

In Fig. 2(b) pre-copy approach [5], the minimal state of the processor is transferred to the target and followed by iterative push phase where the dirty (modified) pages are pushed to destination iteratively until the dirty page rate is less than the pages transferred rate. Then a small stop and copy phase is followed. In pre-copy approach, during iterative push phase, many zero pages which contains all zeros, identical pages (80% above similar) and similar pages (60% to 80% similar) are transferred to the target host. These pages are not required for VM to resume [6] at the destination machine.

In the process of VM migration, the entire virtual machine state is to be transferred. It contains connected device states which are of less size and sent to target host easily. Disk state information is not required to transfer as it is provided by network attached storage (NAS), memory state information of size gigabytes need to be transferred. VM's CPU state information is the small amount and not having much effect in live migration time and downtime [7]. Memory state contains guest OS memory state and all information about processes running within the VM. This is the vast amount of information which badly effects migration time and downtime.

Memory state content contains:

i) Dirtied memory

These memory pages are resident in VM memory as they are actively modified through writing to in-memory pages by the applications while running.

ii) VM configured memory

The amount of memory given by the hypervisor to the VM. This is the physical memory to the guest in VM perspective.

iii) Requested memory

The memory requested to the VM's operating system by the applications to run inside VM. These memory pages may not be resident in memory and not currently in use. When VM configured memory is all used, these memory pages may be swapped out to the disk by swapping.

iv) VM used memory

From the perspective of guest VM, these are the memory pages currently used by the guest operating system actively. These pages are resident in VM memory.

v) Allocated Memory

The physical memory of the underlying hardware that is allocated to the guest VM. This is the memory that is actively used by the VM from the hypervisor perspective.

The relationship between these memories according to their sizes is given as in Fig. 2(c).

The configured memory is a considerable size and contains VM image files. VM images consist more than 80% of zero pages and duplicate contents which, not only occupies more storage but also increases pressure on network transmission [8] especially in the live migration process. These pages are not required by the VM to resume at the target host. We can reduce the size of this memory by compression or deduplication techniques. As decompression time is the drawback of many compression algorithms, deduplication is preferable.

Deduplication

Deduplication is the process, which reduces the required storage by determining the redundant chunks

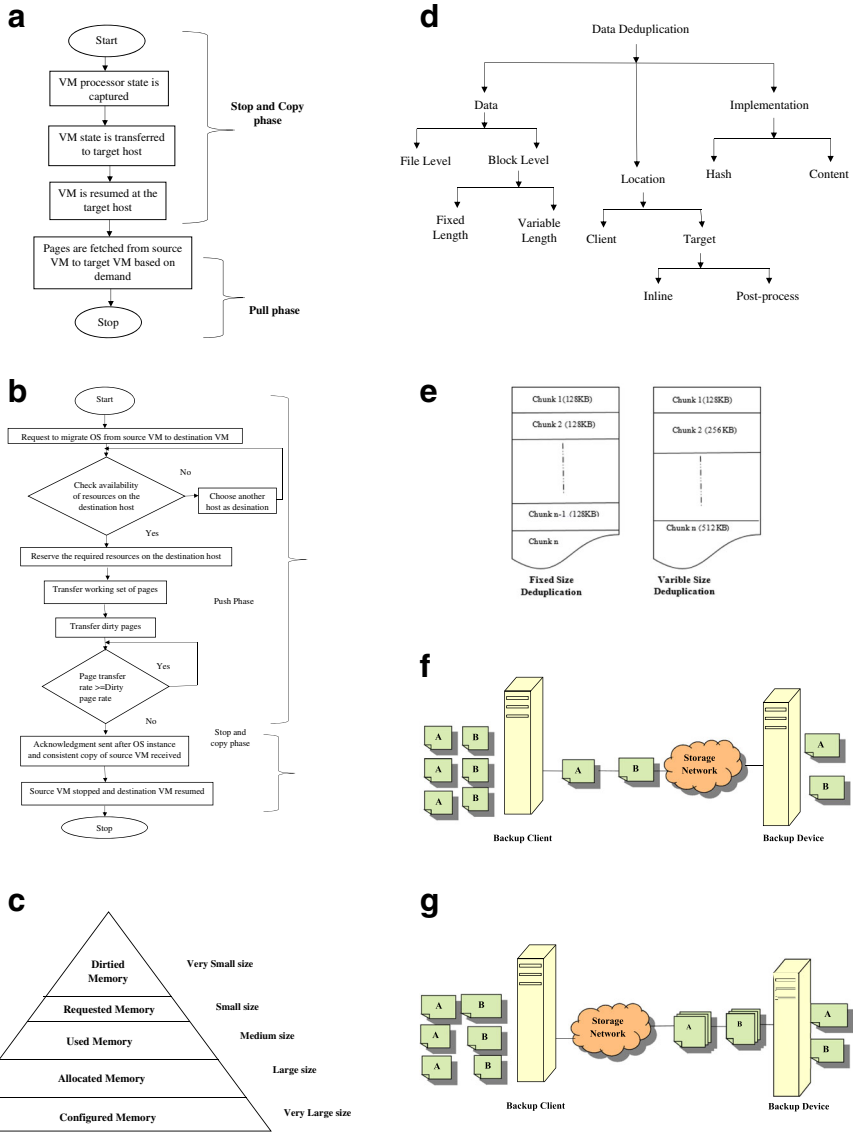

Fig. 2 **a** Post-copy approach with stop, copy and pull phases. **b** Pre-copy approach with iterative push, stop and copy phases. **c** VM Memory content categories. **d** Classification of Data deduplication techniques. **e** Block-Level Deduplication. **f** Client based Deduplication. **g** Target based Deduplication

of data in a set of files and storing only one copy of chunk. For the duplicate chunk, it stores the reference of the chunk without storing the entire chunk, the second time. The classification of deduplication techniques [9] is shown in Fig. 2 (d).

a) Data based deduplication

The unit of data compared in this process is at the file level or sub-file level.

i) File-level deduplication

In file-level deduplication, similar files are identified and are not stored in the file store again. Only the reference to the original file is stored.

ii) Block-level deduplication

This is sub-file level deduplication where the file is divided into chunks called blocks. Based on the size of blocks in the chunking process, block-level deduplication is divided into fixed length block-level deduplication and variable length block-level deduplication.

Fixed length block-level deduplication

In this deduplication, the file is divided into fixed-length blocks. Identical blocks are identified and are removed to minimize the size of the file. It is simple and fast.

Variable length block-level deduplication

In variable-length deduplication, the chunk boundaries are calculated based on the content of the block, and then the similar blocks are identified and removed. It requires more CPU cycles for the identification of block boundaries, but it has the advantage of saving more storage space. Figure 2(e) explains fixed length and variable length block-level deduplication techniques.

b) Location-based Deduplication

Based on where the deduplication process is done, there are two types of deduplication techniques.

- *Client-based deduplication* where redundant data is removed and unique data is transferred to the backup device at the client as in Fig. 2(f).
- *Target-based deduplication*: After receiving all the redundant data at the target, the deduplication process is carried out, and unique data is transferred to the backup device as in Fig. 2(g). If the deduplication is done immediately while receiving, then it is called *in-line deduplication*. After writing to the disk, if deduplication is done, then it is called *post-process deduplication*.

The objective of the work is to reduce the total migration time and downtime in live VM migration process by reducing the amount of VM memory transferred from the source host to destination host during the process of migration.

The significant contributions of the research are as follows:

1) Adaptive deduplication to reduce the VM disk image file size by detecting, and removing the zero, similar and redundant memory copies. Adaptive deduplication uses both fixed and variable size chunking (content defined chunking) strategies.
2) Analysis of deduplication effects on different formats of virtual machine disk images, as these files take more memory in VM and time in the migration process.
3) Akka framework is introduced, by which a high volume of reactive streams (live data) and back pressure mechanism are handled. As the variable length block-level deduplication process takes much time, multi-threading concept is used to accelerate the process.

The remaining paper organized as, the work related to the research discussed in section "Related work". Section 3 explains our motivation for the proposed algorithm. Section "Motivation" explains the research work experimental setup. Results are discussed in section "Experimental setup". The paper is concluded in section "Results and Discussion".

Related work

In the characteristic-based compression (CBC) algorithm [10, 11], where different compression techniques used before the transfer of VM memory pages. Based on the type of pages, different techniques such as Wkdm for high similarity ratio pages, LZO for pages with low similarity were used to reduce their size. An improved algorithm was presented in [12] where memory-to-disk mappings were sent to the target instead of memory pages. From NAS the target host fetches the memory pages after deduplication with the help of NFS fetch queue. MDD (Migration with Data Deduplication) [6] was introduced in live migration for data deduplication of run-time memory image. Zero pages, similar pages were identified using hash-based fingerprints and were eliminated using RLE (Run Length Encode). To cut down the deduplication time multithreading was used by MDD. Extreme binning [13] in which Rabin fingerprints were used to identify the duplicated blocks. MD5 and SHA collision resistant algorithms were used to find the fingerprints. VM scheduling strategies [14] combined with VM replication strategies were introduced to reduce migration latencies associated with live migration of VMs across WAN. Scalability and storage issues of VM images and were resolved by using a liquid distributed file system [15]. Gang Migration using global deduplication (GMGD) [16] was used to detect duplicates in VM memory pages and avoids their retransmission to target host in a cluster. Thus, reducing network overhead while running on several hosts. 42% reduction [17] in migration time was achieved. An optimized Incremental Modulo-K(INC-K) algorithm [18], modulo arithmetic in nature was used for deduplication and achieved 66% better deduplication ratio. In [19], Inner VM and cross-VM duplicates were removed by using a multi-level selective deduplication method. Parallelism was also increased by using local and global deduplication and a high deduplication ratio achieved. Different chunking strategies [20] were applied on various VM disk image dataset and proved that compression rate varied for different operating system versions, software configuration and achieved more savings in storage by identifying zero-filled blocks. In [21] heuristic prediction was used to determine non-duplicates

by using adaptive block skipping, implemented on disk images of size 1 TB which leads to maximum 40% of space savings. Rapid Asymmetric Maximum (RAM) [22], a modified content defined chunking was used, which increases more throughput with less hash-based computations.

Motivation

The main objective of the research work is to reduce the number of pages transferred from source machine to destination machine in live migration process, thus achieving the reduction in migration time and downtime. Hence, in the proposed algorithm, deduplication process on VM disk images applied twice to reduce the duplicates effectively. In the proposed Adaptive deduplication algorithm both fixed size and variable size chunking strategies are implemented to identify the redundant data in VM image files by using Akka stream in the live migration of virtual machines. To accelerate the deduplication process, multithreading in Akka stream framework is used. Akka streams are mainly meant for reactive streams such as live data where handling the non-predetermined volume of data is difficult. For parallel programming, and delay minimization in live data streaming it is better to use actor model. Akka Stream is an actor programming model. In parallel programming, it matches the speed between producers and consumer messages by avoiding avoid back-pressure mechanism. Back pressure means if one component is under stress to handle the incoming messages and subsequently drops the messages. Most of the researchers use the Akka model for Big-data analytics. In this work, Akka is used for streaming of live virtual machine data. As variable length block-level deduplication takes more time Akka with multithreading is used to accelerate the process. In this paper, the results after adaptive deduplication with Akka stream compared with existing deduplication techniques. Akka framework is used here to complete the process without getting OutofMemory Errors. The VM image data set was collected from an image registry of OpenStack, created when launching VMs with the configuration of 2 GB RAM and 10 GB hard disk assuming no applications running on the VM. Each VM is loaded with several guest operating systems like Centos7, Centos 6.9, Ubuntu 12.04 Ubuntu 14.04 and Ubuntu 11.10 versions. Several formats of virtual disk images VDI, QCOW2, VMDK, and VHD of created VMs were taken as input data set to our research work. The proposed adaptive deduplication with Akka stream is as follows.

Algorithm:

// Algorithm for adaptive deduplication using Akka stream.

//Input: VM disk image file in any format.

//Output: VM Disk image without duplicate pages.

Step 1: Start

Step 2: Identify the VM from overloaded/under loaded host in the data centre using minimum utilization policy.

Step 3: Select the target host based on CPU utilization.

Step 4: Reserve resources for VM on the target host.

Step 5: Transfer the necessary CPU state information to the target host.

Step 6: Take VM Configured memory

Step 6.1: If (VM Memory < 1 GB)

Step 6.1.1 Call Fixed Length block deduplication (Chunk size 4KB)

Step 6.2: else

Step 6.2.1 Call Fixed Length block deduplication (Chunk size 1GB)

End if.

Step 7: For each unique chunk obtained from fixed length block deduplication do

Step 7.1: Using Akka framework Call RabinKarpRollingHash (Window size 1 KB) using multithreading.

Step 7.2: Perform live Migration of VM pages.

End for.

Step 8: Resume VM on the target host.

Step 9: Stop.

Figure 3(a) shows the system architecture. Live Migration is the process of moving a virtual machine from source host to destination host, to achieve the goals such as load balancing, fault tolerance, efficient resource usage, and reduction of power consumption. The research work is has three steps. In the first step, if the input file is less than 1GB then fixed length block-level deduplication algorithm is invoked with 4 KB as chunk size, as it is a reasonable chunk size [6] and a considerable number of duplicates are identified and eliminated.

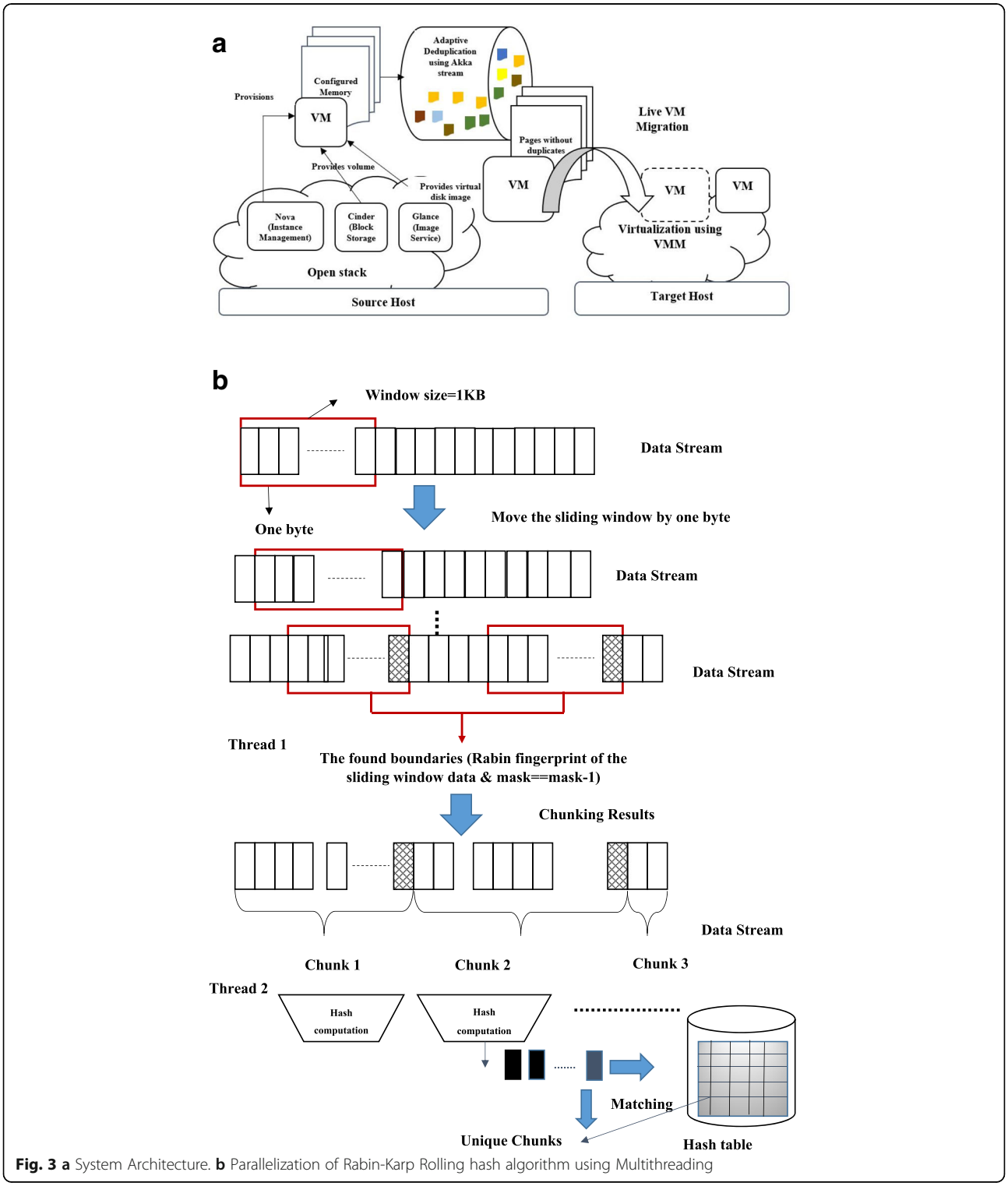

Fig. 3 **a** System Architecture. **b** Parallelization of Rabin-Karp Rolling hash algorithm using Multithreading

Then in the second step, if the virtual disk image file input size is greater than 1GB, split into equal size chunks each of size 1 GB except the last one. Each 1GB split undergoes fixed length block-level deduplication with 4 KB chunk size. The resultant chunks that are obtained after the first stage of duplication undergo variable length block-level deduplication technique and a very reasonable amount of redundant is removed. Thus, it reduces virtual disk image file size efficiently. The Rabin-Karp rolling hash algorithm with 1 KB as the window size is used to find the chunk boundaries in variable length block-level deduplication process. This process takes more time as the boundaries of the chunk are decided based on the content present in the chunk. Hence, this process is parallelized using multithreading in the Akka framework with two threads to improve the speed of deduplication. One thread is for finding chunk boundaries, and other is for computing the hash code as shown in Fig. 3(b). Thus, the size of the virtual disk image file further reduced with a reasonable amount of deduplication time. We used Akka to avoid OutofMemory errors and backpressure mechanism occurred during parallel processing of bulk live data. In the third step, the reduced size of a virtual disk image file is assigned to the VM size parameter in CloudSim. CloudSim setup is explained in section 5. Live migration of VM is performed and the corresponding total migration time is calculated.

Experimental setup

There are some parameters [23] to measure the VM migration performance.

- *Number of pages transferred*: The amount of VM memory or pages transferred during the migration. It also includes zero pages, duplicates, and similar pages.
- *Total Migration time*: The total time required for a VM on source host to migrate and to start on a destination. The sum of all time that is preparation time, page transfer time, down time and resume time.
- *Application degradation*: The performance of the host decreases as the migration is in process.
- *Preparation time*: When migration is initiated, the time for transferring minimal state of the CPU to the destination. In the pre-copy approach, pages become dirty while VM on source host is running. This time includes the entire process of iteratively pushing the dirty pages to the destination host.
- *Resume time*: The time taken by the migrated VM to resume its operations at the target host.
- *Network traffic overhead*: Overhead is the extra operations that are imposed by the virtual machine migration technique. It shows the impact on application performance.
- *Downtime*: The duration of time that a VM is suspended (out of service) before it resumes on the target host.

The experiments are meant to achieve the following goals:

1. To evaluate the deduplication rate of VM image data set using adaptive deduplication techniques and compare with existing fixed length, variable length block deduplication techniques.
2. To carry out the process of live migration of VMs where VMs size dynamically allocated after adaptive deduplication using Akka Streams.
3. To accelerate the adaptive deduplication process by using multithreading.

HARDWARE SPECIFICATIONS:

- Processor: Intel® Core (TM) i5-8250U
- 8th Generation CPU 1.8 GHz DDR4
- OS: Windows 10
- RAM: 8GB
- Memory: 1 TB disk
- System Type: 64-bit OS, × 64-based

SOFTWARE SPECIFICATIONS:

- Software: JDK 1.8 and Oxygen 2
- Simulator: Cloud Sim 3.03

Fixed length and variable length block-level deduplication techniques are implemented in Java and Akka framework. This code returns the size of VM virtual disk image after eliminating the duplicates. CloudSim live migration code is invoked using IqrMu.main(args) which outputs the total migration time.

CloudSim set up

CloudSim configuration and assumed parameters are shown in Fig. 4(a).

Data set

Different formats of VM images like VDI, VMDK, QCOW2, VHD files are provided by glance component of OpenStack image registry. These images have much impact on deduplication rate [24] and are discussed by many researchers. In our work VM Image dataset are taken from OpenStack image registry by creating VMs with a standard configuration of 2GB memory and 10GB hard disk.

a

- Scheduling Interval: 300
- Simulation Limit: 24*60*60
- Cloudlet Length= 2500 * simulation limit
- Cloudlet PEs = 1
- Dynamic Workload
- VM Types: 4
- VM Ram: 870, 1740, 1740, 613
- VM Bandwidth: 100000 (100Mbits)
- VM Size: **Allocated during adaptive deduplication by Akka Stream**
- Host Types: 2 HP Proliant ML110 G4 (1x [Xeon 3040 1860 MHz, 2 Cores], 4GB) HP ProLiant ML110 G5 (1x [Xeon 3075 2660 MHz, 2 Cores], 4GB)
- Host MIPS= 1860, 2660
- RAM= 4096, 4096
- Host Bandwidth= 1 000 000 (1 Gbit/s)
- Storage Capacity= 1 000 000 (1 GB)
- VmAllocationPolicy = iqr, Inter Quartile Range (IQR) VM allocation policy
- vmSelectionPolicy= mu, Minimum Utilization (MU)
- Safety parameter = 1.5, the safety parameter of the IQR policy

Data Center

- System Architecture= x86
- OS= Linux
- VMM= Xen
- Number of physical hosts=50
- Number of VMs = 60

b

$$\text{Deduplication Rate} = \frac{\text{(Original Image Size-Reduced Image Size)}}{\text{Original Image Size}}$$

Fig. 4 **a** Testbed configuration of CloudSim. **b** Deduplication Rate

The given Table 1. Provides several formats of VM images for various operating systems, of CentOS, Ubuntu different versions. OpenStack [24] was developed by Rackspace for NASA and is a cloud operating system which is used to implement public and private clouds easily. The core components of OpenStack are Horizon, Keystone, Nova, Swift, Glance and many other. Horizon is the dashboard which is the graphical user interface which allows the users to automate the cloud resources. Keystone is the identity service which provides the authentication services. Nova is a compute component which manages the creation of many VMs that handle numerous computing tasks. Swift is in object storage which stores many peta bytes of data. Glance is an image registry service which supports VDI, VMDK, VHD, Raw, qcow2 images of VMs. Deduplication rate is calculated as shown in Fig. 4(b). Deduplication rate is affected by the deduplication algorithm, chunk size, and data sets. Variable size chunking strategy gives better deduplication rate than that of fixed size chunking. In our

Table 1 Virtual Machine disk images in the data set (Bytes)

	CentOS7	CentOS6.9	Ubuntu12.04	Ubuntu14.04	Ubuntu11.10
VMDK	1,242,859	446,922	3,511,238,307	4,882,761,848	1,831,810
VHD	146,448	24,576	3,511,646,874	149,467	477,556
VDI	2,322,209	2,216,387	3,412,228,224	4,856,112,381	2,423,744,118
QCOW2	161,368	131,729	3,462,770,688	4,855,561,190	498,630
Total (bytes)	3,872,884	2,819,614	13,897,884,093	14,594,584,886	2,426,552,114

experiments, first we concentrated to reduce the size of several VM disk images with various guest operating systems. To implement fixed length block-level deduplication the chunk size 4 KB is used as it is the reasonable average chunk size to get better deduplication rate [25]. The smaller the chunk size the better the deduplication rate [26] but a smaller chunk size increases the metadata overhead and the time for the deduplication process also.

Results and discussion

In Fixed length block-level deduplication technique any format of virtual disk image file is taken as input and the file is split into small chunks each of size 4 KB. Using the SHA algorithm, the hash code of each chunk is calculated. The hash code and chunk name are stored in the hash table. If any further in coming chunk having the same hash code, the chunk is not stored in the hash table. Thus, the duplicates are eliminated. As the file size is small, there is no significant impact on live migration performance.

From the Fig. 5(a) it has been understood that minimum 6.61% improvement in the deduplication rate for Ubuntu 11.10 files and 92.66% maximum deduplication rate for Centos 6.9 files are achieved. The small the size of the VM image size the large the deduplication rate. The better the deduplication rate, the better reduction in virtual disk image size reduction, the better the migration time and less downtime. From the above figures for different OS versions of virtual disk images, total migration time is reduced by 86.6% for VHD files of Ubuntu 11.10 which is a lighter version. Migration time is very well reduced by 92.6% for VMDK format of centos 6.9 files. We got 50.4% overall deduplication rate for qcow2 files which is the same as existing.

From Fig. 5(b) and (c) it is observed that Ubuntu OS versions don't have many duplicates except for that lighter versions. 89.87% is the maximum reduction in the size of Ubuntu OS disk image files. 19.77% is the percentage of size reduction for Ubuntu14.04 VMDK format which is the minimum among all available Ubuntu versions. 91.6% reduction in total migration time is achieved for Centos6.9 disk image file. Consequently, the downtime, application performance degradation will be reduced reasonably.

Figure 5(d). shows that adaptive deduplication shows the better deduplication rate for all the input files compared to the fixed and variable length block deduplication techniques. The better the deduplication rate, the more the VM file size is reduced. It means the number of VM pages to be transferred is reduced. We also observed that small VM image files which are lighter versions have many duplicates when compared to large VM image files.

Figure 5(e) and (f) show that adaptive deduplication technique gives better performance when compared with fixed and variable deduplication techniques in terms of deduplication rate and migration time. 92.66% for fixed, 94.35% for variable and 95.53% for adaptive deduplication rate achieved regarding size. 91.4% for fixed, 91.6% for variable and 92.5% better reduction achieved regarding total migration time. Minimum percentage of deduplication rate 6.61%, 19.77%, 56.29% achieved by using fixed length, variable length, and adaptive duplication techniques respectively. 5%, 14.1%, 41.5% total migration time is reduced by using fixed length, variable length, and adaptive deduplication techniques respectively.

Figure 5(g) show the comparison of all techniques regarding storage. By using the proposed technique 6.61% minimum for Ubuntu files and 95.5% maximum deduplication rate is obtained for centos files. The minimum of 5% and 92.5% maximum reduction in total migration time is obtained. IM-Dedup [24] uses static (fixed length) chunking procedure and it achieves 80% reduction in overall image storage. We achieved a 83% reduction in the overall storage of images. 3% improvement is significant because this algorithm is used in data centres for optimal storage of VM disk images. As large VM disk images have peta bytes of size, 3% reduction in storage give a significant memory savings. 89.76% reduction in migration time by using adaptive deduplication. The number of duplicates present in VM disk images is based on the data set taken, the type of disk image, and the applications running on the VM. In this paper, we didn't discuss the time for deduplication process since the adaptive deduplication process if it carried out by high-end servers of cloud, the deduplication was completed within no time and there is no significant impact on migration time and downtime.

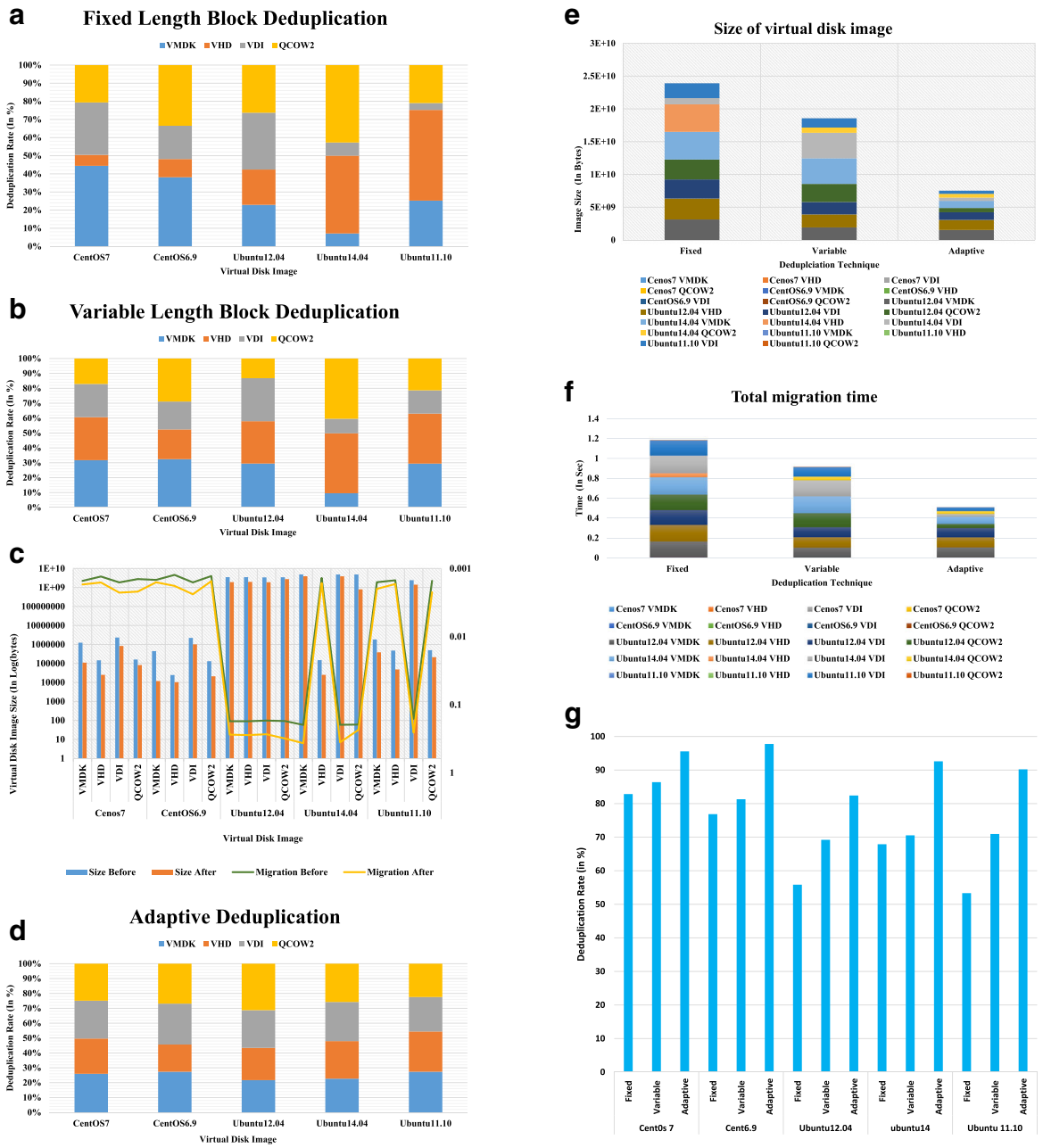

Fig. 5 **a** Deduplication Rate (in %) for Fixed length block deduplication. **b** Deduplication Rate (in %) for Variable length block deduplication. **c** Comparison of virtual disk image size and total migration time before and after variable length block deduplication for the image data set. **d** Deduplication Rate (in %) for Adaptive deduplication. **e** Comparison of size of disk image files after performing fixed length, variable length adaptive deduplication techniques. **f** Comparison of total migration time after performing fixed length, variable length adaptive deduplication techniques. **g** Comparison of Deduplication rate for fixed size, variable size block deduplication and adaptive deduplication

Conclusion

In this research the existing deduplication techniques fixed length, variable length block deduplication techniques are implemented and compared with proposed adaptive deduplication techniques on standard VM images dataset which is taken from open stack image registry when created with each VM configuration of RAM 2GB and hard disk of 10GB assuming no applications running on any VM. The Rabin-Karp rolling hash algorithm is used for variable length block deduplication. IM-Dedup uses static (fixed length) chunking procedure, and it achieves 80% reduction in overall image

storage. We achieved an 83% reduction in the overall storage of images and 89.76% overall reduction in migration time by using adaptive deduplication. 3% improvement in deduplication rate by the proposed method. As the migration time is reduced obviously, downtime and application performance degradation also reduced.

Abbreviations

LZO: Lempel–ziv–oberhumer; NAS: Network attached storage; NFS: Network File System; VM: Virtual machine; VMM: Virtual machine migration

Acknowledgments

We are thankful for the people who gave the technical assistance in getting the data set of VM images.

Funding

Not applicable.

Availability of data and materials

Not applicable.

Authors' contributions

NM did a literature survey on live migration techniques, deduplication techniques. She proposed and implemented adaptive deduplication of virtual machine images by using fixed length and variable length block deduplication techniques. VG guided choosing of parameters for evaluating various techniques. She approved the final version of the paper to be submitted. Both authors read and approved the final manuscript.

Competing interests

TYJNaga Malleswari, G.Vadivu declares that they have no competing interests.

Publisher's Note

Author details

[1]Department of Computer Science and Engineering, SRMIST, Chennai 603203, India. [2]Department of Information Technology, SRMIST, Chennai 603203, India.

Received: 6 June 2018 Accepted: 9 January 2019
Published online: 06 February 2019

References

1. Naga Malleswari TYJ, Rajeswari D, Senthil J (2012) A survey of cloud computing architecture and services provided by various cloud service providers. In: Proceedings of 2nd international conference on demand computing, the Oxford College of Engineering, November 15–16, Bangalore
2. Venkatesha, Sharath & Sadhu, Shatrugna & Kintali, Sridhar. (2009). Survey of Virtual Machine Migration Techniques
3. Sun G, Liao D, Vishal A, Dongcheng Z, Yu H (2016) A new technique for efficient live migration of multiple virtual machines. Futur Gener Comput Syst 55:74–86. https://doi.org/10.1016/j.future.2015.09.005
4. Choudhary A, Govil MC, Singh G, Awasthi LK, Pilli ES, Kapil D (2017) A critical survey of live virtual machine migration techniques. J Cloud Comp 6(23):1–41. https://doi.org/10.1186/s13677–017–0092-1
5. Diego Perez-Botero, "A Brief Tutorial on Live Virtual Machine Migration from a Security Perspective"
6. X Zhang, Z Huo, J Ma and D Meng, "Exploiting Data Deduplication to Accelerate Live Virtual Machine Migration," 2010 IEEE international conference on cluster computing, 2010, pp. 88–96. https://doi.org/10.1109/CLUSTER.2010.17
7. S. Sharma and M. Chawla, "A technical review for efficient virtual machine migration," 2013 International Conference on Cloud & Ubiquitous Computing & Emerging Technologies (CUBE), Pune, India, 2014, pp. 20–25. doi:https://doi.org/10.1109/CUBE.2013.14
8. Hu, Wenjin, Hicks, Andrew, Zhang, Long, Dow Eli, Soni, Vinay, Jiang, Hao, Bull, Ronny, Matthews, Jeanna. (2013). A quantitative study of virtual machine live migration. Proceedings of the 2013 ACM Cloud and Autonomic Computing conference. https://doi.org/10.1145/2494621.2494622
9. Naga Malleswari TY, Malathi D, Vadivu G. Deduplication Techniques: A Technical Survey. International J Innovative Res Sci Technol|, 2014, 1, (7) pp. 318–325
10. Jin Hai & Deng, Li & Wu, Song, Shi, Xuanhua, Pan, Xiaodong. (2009). Live virtual machine migration with adaptive, memory compression. Proceedings - IEEE International Conference on Cluster Computing, ICCC. 1–10. 10.1109/CLUSTR.2009.5289170
11. Jin H, Deng L, Wu S, Shi X, Chen H, Pan X (2014) "MECOM: live migration of virtual machines by adaptively compressing memory pages", ACM transactions on. Futur Gener Comput Syst 38:23–25
12. Jo C, Gustafsson E, Son J, Egger B (2013) Efficient live migration of virtual machines using shared storage. ACM SIGPLAN Not 48:41–50. https://doi.org/10.1145/2451512.2451524
13. D. Bhagwat, K. Eshghi, D. D. E. Long and M. Lillibridge, "extreme binning: scalable, parallel deduplication for chunk-based file backup," 2009 IEEE international symposium on modeling, Analysis & Simulation of Computer and Telecommunication Systems, London, 2009, pp. 1–9. doi: https://doi.org/10.1109/MASCOT.2009.5366623
14. Jinhua H, Jianhua G, Sun G, Zhao T (2010) A scheduling strategy on load balancing of virtual machine resources in cloud computing environment, 3rd international symposium on parallel architectures. Algorithms and Programming:89–96
15. X. Zhao, Y. Zhang, Y. Wu, K. Chen, J. Jiang and K. Li, "Liquid: a scalable deduplication file system for virtual machine images," in IEEE Transactions on Parallel and Distributed Systems, vol. 25, no. 5, pp. 1257–1266, 2014.doi: https://doi.org/10.1109/TPDS.2013.173
16. U. Deshpande, B. Schlinker, E. Adler and K. Gopalan, "Gang Migration of Virtual Machines Using Cluster-wide Deduplication," 2013 13th IEEE/ACM International Symposium on Cluster, Cloud, and Grid Computing, Delft, 2013, pp. 394–401.doi: 10.1109/CCGrid.2013.39
17. H. Liu, H. Jin, X. Liao, C. Yu and C. Z. Xu, "Live virtual machine migration via asynchronous replication and state synchronization," in IEEE Transactions on Parallel and Distributed Systems, vol. 22, no. 12, pp. 1986–1999, Dec. 2011. doi: https://doi.org/10.1109/TPDS.2011.86
18. J. Min, D. Yoon and Y. Won, "Efficient deduplication techniques for modern backup operation," in IEEE Transactions on Computers, vol. 60, no. 6, pp. 824–840, 2011. doi: https://doi.org/10.1109/TC.2010.263
19. W. Zhang, H. Tang, H. Jiang, T. Yang, X. Li and Y. Zeng, "Multi-level Selective Deduplication for VM Snapshots in Cloud Storage," 2012 IEEE Fifth International Conference on Cloud Computing, Honolulu, HI, 2012, pp. 550–557. doi: https://doi.org/10.1109/CLOUD.2012.78
20. Jin Keren and Ethan L. Miller. "The effectiveness of deduplication on virtual machine disk images." SYSTOR '09 Proceedings of SYSTOR 2009: The Israeli Experimental Systems Conference SYSTOR (2009). Article 7
21. Zhou B, Wen J (2016) A data deduplication framework of disk images with adaptive block skipping. J Comput Sci Technol 31(4):820–835. https://doi.org/10.1007/S11390-016-1665-Z
22. N.S. Widodo, Ryan & Lim, Hyotaek & Atiquzzaman, Mohammed. (2017). A new content-defined chunking algorithm for data deduplication in cloud storage. Futur Gener Comput Syst 71. https://doi.org/10.1016/j.future.2017.02.013
23. Naga Malleswari TYJ, Vadivu G, Malathi D (2015) Live virtual machine migration techniques-a technical survey. In: 2nd international conference on intelligent computing, communication and devices intelligent computing august 2015, intelligent computing, communication and devices, volume 1. Springer, pp 303–309
24. Jilin Zhang, Shuting Han, Jian Wan, Baojin Zhu, Li Zhou, Yongjian Ren, and Wei Zhang (2013) IM-Dedup: an image management system based on deduplication applied in DWSNs, international journal of distributed sensor networks July 16, 2013, doi.org/https://doi.org/10.1155/2013/625070
25. Kim D., Song S., Choi BY. (2017) Existing deduplication techniques. In: Data Deduplication for Data Optimization for Storage and Network Systems. Springer, Cham
26. TYJ Naga Malleswari, Vadivu G (2017) Deduplication of VM Memory Pages Using Mapreduce In Live Migration, ARPN Journal Of Engineering And Applied Sciences, 2017, Vol 12, No. 6, Pp. 1890–1894