The Journal of Systems and Software 000 (2016) 1–13

Contents lists available at ScienceDirect

The Journal of Systems and Software

journal homepage: www.elsevier.com/locate/jss

Clustering-based acceleration for virtual machine image deduplication in the cloud environment☆

Jiwei Xu[a,b,c], Wenbo Zhang[a,*], Zhenyu Zhang[a,b], Tao Wang[a], Tao Huang[a,b]

[a] Institute of Software, Chinese Academy of Sciences, Beijing 100190, China
[b] State Key Laboratory of Computer Science, Beijing 100190, China
[c] University of Chinese Academy of Sciences, Beijing 100190, China

ARTICLE INFO

Article history:
Received 20 December 2014
Revised 8 January 2016
Accepted 20 February 2016
Available online xxx

Keywords:
Cloud computing
Virtualization
VM image
Deduplication

ABSTRACT

More and more virtual machine (VM) images are continuously created in datacenters. Duplicated data segments may exist in such VM images, and it leads to a waste of storage resource. As a result, VM image deduplication is a common daily activity in datacenters. Our previous work Crab is such a product and it is on duty regularly in our datacenter.

The size of VM images is large and the amount of VM images is huge, and it is inefficient and impractical to load massive VM image fingerprints into memory for a fast comparison to recognize duplicated segments. To address this issue, we in this paper propose a clustering-based acceleration method. It uses an improved *k*-means clustering to find images having high chances to contain duplicated segments. With such a candidate selection phase, only limited VM image candidate fingerprints are loaded into memory.

We empirically evaluate the effectiveness, robustness, and complexity of the proposed system. Experimental results show that it significantly reduces the performance interference to hosting virtual machine with an acceptable increase in disk space usage, compared with existing deduplication methods.

1. Introduction

Cloud computing is an on-demand and self-service computing paradigm. A main enabling technology for cloud computing is virtualization. Virtualization can provide several servers on a single physical host in forms of virtual machine (VM). For each virtual machine, all its disk contents (including operating system, application software, data, and so on) are encapsulated to form a whole virtual machine image. This has brought obvious convenience to VM image backup and it is well known that data backup is of great significance to disaster recovery. As the prevalence of cloud computing, more and more data centers are adopting virtualization technology as server management solution. A standard example is the famous IaaS provider Amazon that allows users to store their virtual machine images or image snapshots to Amazon Simple Storage Service (S3) across regions periodically. However, it is a solution of full backup and gives rise to the duplicate copies of repeating data, which may cause serious storage wastes.

1.1. VM image deduplication and the performance issue

Deduplication techniques (Fu et al., 2011; Bhagwat et al., 2009; Won et al., 2008; Zhang et al., 2013) are extensively employed with the backup operation to remove duplications of VM data segments. In practice, Content Addressable Storage (CAS) technologies (Tolia et al., 2003), such as Venti (Quinlan and Dorward, 2002), is one of the most common deduplication methods. CAS exposes a digest generated by a cryptographic hash function (such as Rivest or SHA-1 Eastlake and Jones) from the data block. The digest, also referred to as fingerprint, is treated as the address of the data block content. CAS system solely saves a single data block copy, but shares the data block among different backup files by checking the digests of data blocks.

Typically, a deduplication process consists of three steps: data chunking and fingerprinting (Quinlan and Dorward, 2002; Policroniades and Pratt, 2004; Hunt et al., 1998; Muthitacharoen et al., 2001), index lookup (Min et al., 2011; Lillibridge et al., 2009; Zhu et al., 2008) and chunk store (Mao et al., 2014).

Index lookup is the key step of deduplication because it determines whether a chunk is duplicated. However, with the dramatic growth of stored data, the fingerprint index table becomes huge and cannot be stored in memory, causing index lookup a performance bottleneck (Min et al., 2011). According to our previous

☆ The conference version of the paper in Xu et al. (2014) is published in the IEEE 38th Annual Computer Software and Applications Conference (COMPSAC 2014).

* Corresponding author. Tel.: +86 10 62661583 630.
E-mail address: zhangwenbo@otcaix.iscas.ac.cn (W. Zhang).

http://dx.doi.org/10.1016/j.jss.2016.02.021
0164-1212/

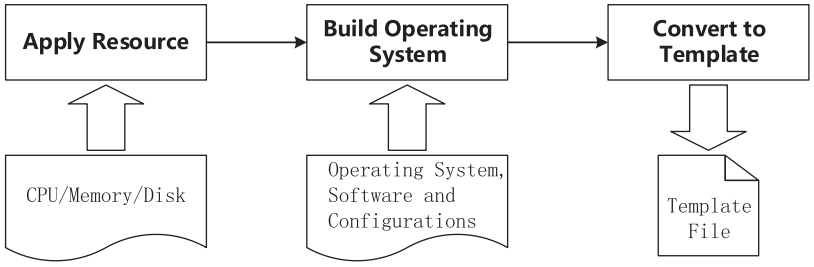


Fig. 1. Image standard installation process.

experiences, about half of the deduplication time is wasted due to the frequent swap of the fingerprint table between memory and disk.

In cloud environment, this issue can be more serious. The severely inadequate performance is due to the nature of cloud computing and VM images. Using the mechanism of VM encapsulation, it is very easy to create, duplicate or backup a VM, according to the scalability and reliability demand of cloud. As a result, a new created or backup image is likely to be similar to an existing image or a small set of common images than the other images in the repository (Jayaram et al., 2011).

However, there is no way to easily know which VM images might have duplications and how much duplications they might have. Conventionally, in both centralized environment Jin et al. (2009) and distributed environment Zhang et al. (2012), all the VM images fingerprints are loaded to the memory to perform the comparison before deduplication. That causes a serious performance bottleneck.

1.2. Our work in the paper

In this paper, we revisit the problem of deduplication, analyze various scenarios of virtual machine image generation, and propose to employ a clustering method to select deduplication candidates to accelerate the index lookup. We proposed an improved *k*-means clustering method, emphasize the initial center selection issue, the trigger timing issue and group merging issue in particular, and introduce a local deduplication method to address the virtual machine image deduplication problem. The process of the clustering based VM image deduplication system is as follows. First, we divide all the images in backup repository into groups according to their similarity. Thus, we assure that images within a group have high similarities, which means that images in one group share large number of identical data blocks. Correspondingly, the fingerprints are divided into groups and each group of fingerprints is a subset of the total fingerprints set. As a result, the fingerprint size of each group is much smaller than the total fingerprint size and such a group can be loaded into memory completely. The number of groups, as a parameter, is determined according to available memory size and total size of block fingerprints. Second, when a new image is requested, we first determine the group it belongs to using a sampling method; and after that load the corresponding fingerprint set into memory to conduct the duplication process. We conduct experiments to validate the proposed clustering method, and evaluate its performance. Empirical results show the proposed clustering method promising.

Our contributions in this paper are as follows:

(1) A clustering-based virtual machine image classification method is given to reduce the fingerprint search space and improve the index lookup performance. This method use local deduplication to replace global, so as to reduce the deduplication operation time and performance interference.

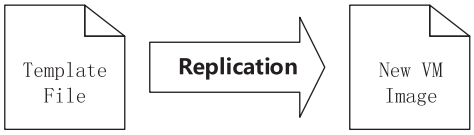


Fig. 2. Templates copy to new virtual machine image.

(2) This is the first work that takes the image content layout into consideration during image deduplication, which can help to classify the images into small groups to reduce the fingerprint search space.
(3) We innovatively propose the method of periodical triggering and small group merging to facilitate virtual machine image deduplication.
(4) We conduct experiment to evaluate the effectiveness, efficiency, and robustness of the proposed method. The empirical results show our method promising.

The paper is organized as follows. Section 2 introduces the background of the work. Section 3 introduces our system architecture and elaborates on the fingerprint clustering approach and sampling method in virtual machine deduplication. Section 4 presents the experimental results and gives the analysis to the results. Sections 5 and 6 review related work and Section 6 draw conclusions, respectively.

2. Background and motivation

In this section, we revisit common scenarios to demonstrate the need and feasibility of virtual machine image deduplication.

2.1. VM image generation

In cloud environment, there are always tens of thousands of virtual machines per cluster which costs a large amount of storage (Zhang et al., 2013). This virtual machine image sprawl Reimer et al. (2008) can lead to a serious storage crisis. Usually, the virtual machine image can be generated as follows:

Standard installation. Fig. 1 illustrates the VM image standard installation process. A cloud service provider or consumer would build some virtual machine images as standardized templates. These templates, such as Amazon Machine Images (AMIs), are some special types of pre-configured operating systems and virtual application software. The templates are used to create a virtual machine within the specific virtualization platform.

Template replication. Fig. 2 illustrates the template replication process. When a virtual machine needs to be created, a template would be copied to form a new virtual machine image. That could go through the trouble of rebuilding the entire software stack.

Specialized configuration. Fig. 3 illustrates the specialized configuration process. The newly generated image needs some specialized configuration to work properly. For example, assigning a new

Please cite this article as: J. Xu et al., Clustering-based acceleration for virtual machine image deduplication in the cloud environment, The Journal of Systems and Software (2016), http://dx.doi.org/10.1016/j.jss.2016.02.021

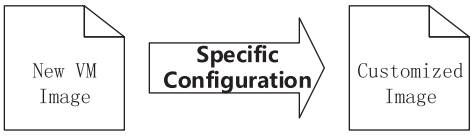


Fig. 3. Virtual machine image customization.

IP address or hostname, rewriting software configurations or installing new applications can be necessary.

2.2. VM image deduplication

As we can see that virtual machines usually inherit from some certain golden images (also called templates), so there would be a large amount of duplicated blocks among these virtual machines. Meanwhile, the frequent backing up operation and periodic virtual machine snapshots also need huge storage. A snapshot is generated in driver level based on copy-on-write technology, so it can be archived with low cost by sharing identical data segments with the original image file. However, when we back up a snapshot, the driver level semantic of snapshot would be broken and a new file would be rebuilt, which will also produce a large amount of duplicate blocks.

Based on the above consideration, we need to deduplicate the replicated blocks to relieve the great storage pressure on the backup of virtual machine image. Different from the general backup data, the virtual machine images often have similar to a small subset characteristic (Jayaram et al., 2011), which can be explained that the "similar" images would have high chance (even more than 90%) to share identical data blocks and the "dissimilar" images would have low chance (less than 1%) to share identical data blocks. Here, the mentioned "similarity" is related to those images with same operating system, applications, and dataset.

2.3. Clustering-based deduplication acceleration

Deduplication can be very time-consuming with the increase of stored data. For example, if the fingerprint repository is twice or more the size of the available memory, the deduplication time would double. In fact, most time usage are ineffective. The extra time are mainly wasted in waiting for the disk I/O of the fingerprint table.

We think of preprocessing the stored VM image and their fingerprints to avoid the disk bottleneck problem during deduplication process. Since, the VM images have similar to a small subset characteristic, we can classify all the VM images into small groups to make sure that each group's fingerprint size would be no larger than the available memory. It seems that such a simple method may work. However, there must be a small number of blocks that are duplicated stored among different groups. Compared to the huge storage gains of the deduplication, this light wasted storage space will be trivial. As a result, it cannot be directly done.

In this paper, we in the so-called pre-deduplication phase, employ a clustering method to serve the purpose. In such a way, we can (1) reduce the search space of index lookup process, (2) avoid the swap of fingerprint table between the memory and disk, (3) reduce the time consume with a slight storage space lost. In the next section, we will present our approaching.

3. Clustering-based deduplication acceleration

In this section, we first introduce our on-duty system Crab (Xu et al., 2014), and based on it propose our clustering method used for deduplication acceleration.

3.1. Preliminaries: the Crab system

We have developed a deduplication backup System (**Crab** for short) (Xu et al., 2014). The Crab system works as illustrated in Fig. 4. It first uses a cluster method ("*F*" in the "*Grouping*" step) to classify all the images into small groups, and then employs a sampling method ("*S*" in the "*Selecting*" step) to select a proper group to perform the deduplication.

We now revisit its deployment architecture and describe its backup strategy.

As illustrated in Fig. 5, we classify the devices into three types: virtual machine host, image storage and backup storage. The virtual machine host provides computing resources (CPU/GPU) and memory resources to the virtual machine. The image storage can be either a shared storage device supporting masses of virtual machine disk images from a different virtual machine host or a local storage device only accessed by local virtual machines. The backup storage, as its name suggests, is used to store the image backups. Our work focuses on the image backup operation under the similar kind of deployment architecture, because this architecture is very popular in private cloud datacenter. Actually, it is similar to the Amazon cloud service architecture, where the virtual machine host is like EC2, the image storage is like EBS, and the backup storage is like S3.

Deduplication operation often consumes a lot of resources in both host side and storage side. Taking into account the sequence of two actions, backup and deduplication, there would be three kinds of strategies, deduplication before backup, deduplication

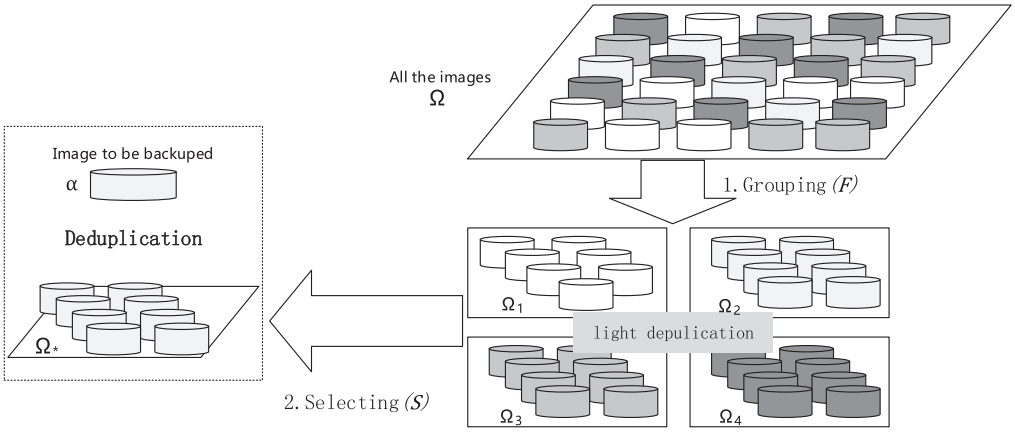


Fig. 4. Clustering-based deduplication diagram.

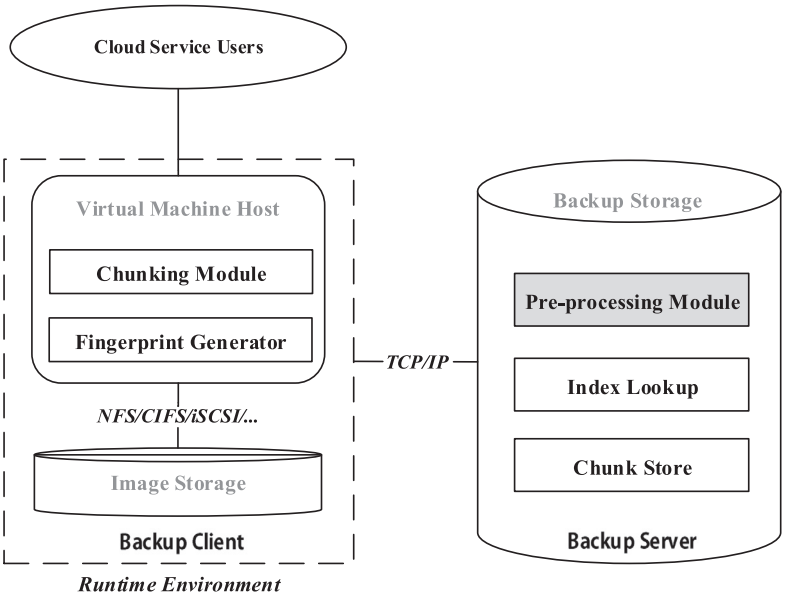


Fig. 5. The Crab system deployment architecture.

after backup, and deduplication during backup. In Crab, we perform the deduplication operation during backup. In this case, the data transmission would be as small as possible (Xu et al., 2014).

In the Crab system, we treat a virtual machine host as the backup client and the backup storage as the backup server. As illustrated in Fig. 5, we put chunking module and fingerprint generator on client side. The other components are put on server side. Such design aims to reduce the network traffics, since it only needs to transmit the changed chunks from image storage to backup storage.

3.2. Our clustering-based acceleration proposal

Based on similarities among images, we employ a clustering method to merge groups. Particularly, we adaptively determine the period to trigger the clustering operation. The details of our proposal are stated in this section.

3.2.1. Similarity based clustering

Similar images have a large chance to contain identical chunks. To calculate the image similarity, we first review the relationship among image backups, fingerprints, and disk storing chunks. Image backup is a logical entity, which is composed by a metadata file, and the corresponding chunks as illustrated in Fig. 6. The metadata file is formed by a set of sequential fingerprints. However, there are large amount of inner duplicated blocks in an image, and the backup metadata may also have a lot of identical fingerprints about these duplicated blocks. Zero-filled block is one such kind of inner duplicated blocks. The number of these valueless zero-filled blocks can be even larger than the number of much more valued blocks.

To calculate the similarity, we first eliminate the inner duplicated blocks. We regard this process as inner deduplication. As the metadata file contains all the indexes of the entire member blocks of an image backup, the similarity of images equals to the similarity of the corresponding metadata. As mentioned above, a metadata can be treated as a vector of fingerprints $M = \langle f_1, f_2, \ldots, f_n \rangle$. We use $M' = (f_1, f_2, \ldots, f_m)(m \leqslant n)$ to represent the trimmed fingerprints set. We further name M' the *feature set* of the corresponding image, and the similarity between two images A and B

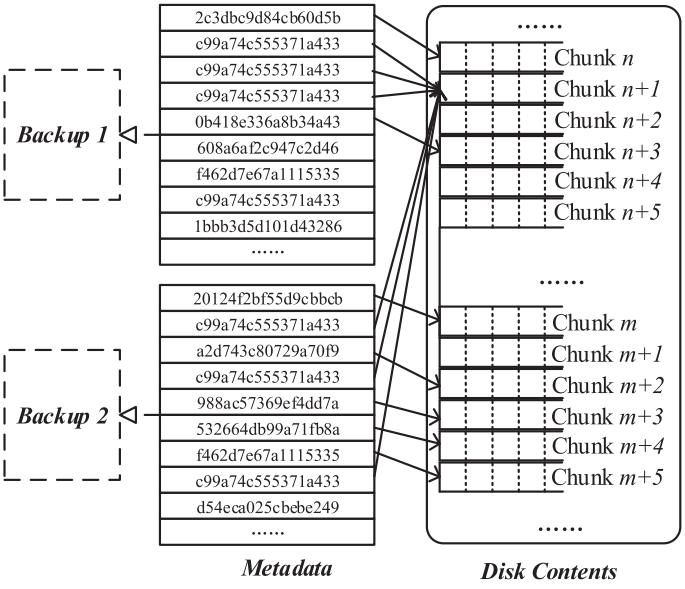


Fig. 6. Relationship among backup, fingerprint and chunks.

is calculated using the following formula Xu et al. (2014).

$$Sim(A, B) = \frac{2 \times |M'_A \cap M'_B|}{|M'_A| + |M'_B|}$$

We improve the *k*-means clustering algorithm according to the specific feature of the virtual machine image and use the improved algorithm to classify the image fingerprints. The image fingerprints data sets are one-dimensional data sets. It has simple structures and is easy to analyze. Meanwhile, *k*-means clustering is simple and common. We can ensure that an entire group of fingerprints index, which is used in the index lookup process, can be put into RAM memory by adjusting the value of *k*.

Since similar images often have the same operating system and file system, we first divide the images into several groups. We get the general information about the operating system and file system from the first sector, which called boot sector of the image file,

Please cite this article as: J. Xu et al., Clustering-based acceleration for virtual machine image deduplication in the cloud environment, The Journal of Systems and Software (2016), http://dx.doi.org/10.1016/j.jss.2016.02.021

Table 1
IIS structure and possible values.

Items	Operating system	File system	Disk partition
Possible values	Windows	4—DOS FAT16 < 32M	00: active partition
	Ubuntu	5—EXTEND	80: inactive partition
	CentOS	6—DOS FAT16 > 32M	Others: invalid partition
	SUSE	7—NTFS(OS/2)	
		83—LINUX > 64M	

and then for each group make a further classification through our improved *k*-means clustering algorithm.

The improvement of our algorithm mainly represents in the selection of the first k points, since the selection of the initial k points is very important to the classification result. Different from the random selection strategy, we take into account the feature of virtual machine image, such as disk partition, operating system, file system and so on. We use a specific structure named Image Information Structure (IIS) to represent these features. Table 1 lists out the items and the possible values used in IIS. The operating system criterion has the maximum weight, the other criteria including file system and disk partition also have effects. In the centroid selection step, we group images into different set according to the values of operating system, file system and disk partition. And then we select the newest image from the biggest set as the centroid.

3.2.2. Grouping process and sampling strategy

Different from all the previous work, we embed a pre-process module based on Crab in the deduplication system to accelerate the index lookup progress (Xu et al., 2014). In this module we classify the images into different groups to reduce the index lookup space from a large global space to a small local space.

Totally in-memory index lookup is an important feature of the approach. The basic principle is to divide all images into groups according to image similarity. Generally, images in the same group have high chances to contain identical chunks. With such considerations, we first select a group for the new coming image backup, and then load the index table of that group into memory to perform deduplication. When a new image backup is requested, we get a fingerprint sample set from the image according to certain rule, and calculate the sample hit rate in each group. The sampling hit rate is the statistical indicator of duplication rate between image and group. A high hit rate means high duplication and a low hit rate means low duplication. So the group with maximum hit rate is chosen to carry out image deduplication.

As for the storage progress, we form chunks into a block, which is stored as a file upon the file system on the backup server. The fingerprint index table maintains the map from the fingerprint to the block, which contains the chunk and inner offset of the chunk within the block.

In our work, we use two strategies, *simple random sampling* and *systematic sampling*, to get the sample set Xu et al. (2014).

Simple random sampling (SRS): The image is firstly divided into equal chunks of size 4KB. Then we randomly select n chunks and calculate their corresponding fingerprints to form the sample set S.

Systematic sampling (SS): The image is firstly divided into equal chunks of size 4KB. Then the chunks are divided into m groups. We randomly select n/m chunks from each group and calculate their fingerprints to form the sample set S.

A proper sample size n is estimated using the following formula (Xu et al., 2014).

$$n=\frac{2xp-[\Phi^{-1}(1-\rho)]^2pq+\sqrt{[[\Phi^{-1}(1-\rho)]^2pq-2xp]^2-4p^2x^2}}{2p^2r}$$

where p represents the sample hit rate to the most similarity group, ρ represents the probability of the event that there are at least x samples hits in the similarity group and r represents the inner duplication rate. The parameters should be determined according to the historical experience. For example, if we obverse that the inner duplication rate is 30% and we estimate that the sample hit rate in the most similar group is about 40%, while we claim that the probability of at least 100 sample hits should be 99%. Then we have $\rho = 0.99, p = 0.4, x = 100, r = 0.3$. The derivation process of this formula can be found in our previous work (Xu et al., 2014).

However, a problem within this architecture is that the chunking module and fingerprint generator are CPU sensitive. Since the virtual machine host provides computing resources for all the virtual machines, there would be a resource competition between the virtual machines and the backup operation. It would cause a performance interruption to the virtual machine, and we are aiming to reduce it.

3.2.3. Other acceleration considerations

The above algorithm can effectively separates the images into several groups, however the group size may vary greatly. Some groups may have only a few images and the total feature set size is also very small. That would result in a dramatically increase of comparisons and the performance deteriorates in the sampling phase. Thus, we need to merge the small groups to form a large group. There are two principals we follow in the merging process:

Small group principal: All the merging groups must be small groups. The definition of small group depend on the environment. In our environment, we treat the groups whose total feature sizes are less than $M_s/2$ as small groups. Here, M_s is the available memory size.

Maximum memory principal: The memory requirement of the new merged group must be lower than the threshold. Otherwise, the clustering process and merging would become an endless loop.

Clustering algorithm can be time-consuming and frequent image similarity computation also consumes lots of I/O resources. Fortunately, this work is completed in the pre-process module, and would not affect the deduplication backup process any longer.

Frequently running the algorithm will waste large amounts of resources, while seldom triggering the algorithm may cause the block index table to be out of memory. In our model, we set a right period to trigger the clustering algorithm. We suppose that there are k groups of images and the available memory size is M_s. For each group $i (0 < i < k)$, we set a threshold M_i, $M_i < M_s$. In practice, we evaluate the daily backup increment and find an average value ΔM. Let S_i represent the total feature set size of group i. If $S_i + \Delta M > M_s$, the algorithm will be triggered to divide group i to several sub groups. Such mechanism also ensures that the time complexity of the clustering algorithm is in an acceptable range.

4. Evaluation

In this section, we evaluate the effectiveness, robustness, and complexity of the whole system. We first test the sample hit rate and group similarity to verify the effectiveness of our method. Then, we test the impact of sample size on sample error and stability of the clustering method to verify the robustness of our method. After that, we give a full comparison with existing work. At last, we discuss the threads to validity.

Please cite this article as: J. Xu et al., Clustering-based acceleration for virtual machine image deduplication in the cloud environment, The Journal of Systems and Software (2016), http://dx.doi.org/10.1016/j.jss.2016.02.021

4.1. Experiment setup

The sample hit rate represents the hit rate from sample to different groups, which indicates the effectiveness of sampling method for finding the most similar group. Group similarity is displayed through the statistical indices (including mean value, variance, max value, min value, median) of similarities between the centroid and other images within a group, which indicates the effectiveness of clustering method. To study the impact of sample size on sample error, we iterate different sample size from 100 to 1000, take 100 for each size, and illustrate the average error and the maximum error.

Multi-level selective deduplication (short for MSD in this work) (Zhang et al., 2012) are used for VM snapshot deduplication in Aliyun, the largest public cloud of China. MSD classifies images into groups to reduce the memory requirement, but solely according to the operating system factor, while we classify images according to the image similarity. Similarity is affected by but not only by the operating system. The similarity is more flexible and can be used for more fine-grained classification of images. We compare the deduplication performances of Venti, Crab and MSD. Venti is the base work of deduplication and uses the full deduplication technology, so we regarded it as the baseline. The deduplication rate in our evaluation is evaluated by the ratio of the compressed data size to the original data size. The deduplication time metric is defined as the deduplication processing time. Since the clustering operation of Crab belongs to pre-processing stage which can be done in a few seconds and there is no need to execute every time, we do not include this in deduplication time. The RAM usage is measured by recording the space overhead of index-lookup.

We selected 584 different virtual machine images from OnceCloud ISCAS, a cloud platform builds by our institute. Each image size is about 15GB to 20GB. There are 416 raw format images and 168 vhd format images. The total size of these images is 6.68TB. We set the maximum available memory that can be used by index lookup process as 500MB. As we use 128bit MD5 value as the fingerprint and 64bit address as the block index, a record of one chunk need 192bit (24B) storage. If the chunk size is set to 4K, it will need 40.2GB space to store the index of 6.68TB data. Even if the duplicated block has been removed, the total fingerprint table size would be much larger than the given memory size. Obviously, the entire index lookup table cannot be loaded into the memory. Due to the lack of access to source code of Venti and MSD, we have in advance implemented both of them and performed deduplication operations with it.

In our experiments, five blade servers are used as physical host machine. Each server has two Intel Xeon E5645 CPUs, 600GB disk and 32GB RAM. The backup side is a storage cluster with a 10TB storage space. All of these devices are connected via Gigabit LAN.

4.2. Effectiveness

Effectiveness of the system is evaluated in the hit rate of the sampling method and the group similarity of the clustering method.

4.2.1. Hit rate of the sampling method

For new coming images, we use a sampling method to select a group used to perform the deduplication operation. In this experiment, we test the sampling method with both the vhd format image and raw format image. For each kind of image, we separately take samples for ten times and calculate the hit rates for each individual group. At last, a comparison of the average sample hit rate and the practical image duplication rate is given to indicate the effectiveness of the sampling method. For the vhd and raw format image, we set parameters respectively as follows:

$\rho = 0.99, p = 0.4, x = 100, r = 0.4$ and
$\rho = 0.99, p = 0.4, x = 100, r = 0.112$

According to sample size formula in Section 3.1.3, the sample size should be 748 and 2672.

Fig. 7 illustrates the average sample hit rate and the practical image duplication rate in each group. There are two plots in Fig. 7. The top plot represents the sample hit and practice duplication rate of vhd format image in different image group and the bottom plot presents the sample hit rate and duplication rate of raw format image in different image group. The x-axis means the group number and the x-coordinate x means the xth group. The y-coordinate means the hit rate value. There are three legends in this figure. The black rectangle shows the SRS hit rate, the gray rectangle shows the SS hit rate and the white rectangle shows the practice image duplication rate. Let us take the left-most group in the top plot as an example. All the three values are almost zero. It means that the given image is not similar with group and it should not select this group to perform deduplication operation. However, the given image has the highest hit rate (about 76%) within the fourth group. That means we should select the fourth group to perform the deduplication operation for the given image.

Besides, from Fig. 7 we can conclude that for both SRS method and SS method, their results are very close to the practical duplication rate in the group. From the result of vhd format image hit rate in Fig. 7, we can infer that the image used in this experiment should belong to group 4. As for the raw format image, it should belong to group 5. For each set of experiments, their results are much closed to one another. This fully demonstrates that the sampling method is effective for the new image classification.

From the above experiments, we have some interesting observations. First, we can see that there are still other high hit rate groups besides the highest hit rate group. Taking the vhd format image as an example, group 4 has the highest sample hit rate (about 76%). Nevertheless, group 7 also has very high sample hit rate (about 38%). This would result in duplicated blocks among different groups. That is the disadvantage of local deduplication approach. Second, it is generally considered that the systematic sampling is better than the simple random sampling. However, our experiment result shows that both simple random sampling and systematic sampling work well in image classification and this should owe to our reasonable sample size approximation.

4.2.2. Group similarity of the clustering method

Since the result of clustering algorithm is indeterministic, which means that the final results of deduplication depend on realistic environments, its effectiveness has to be tested for the high availability goal. Based on the giving dataset and memory limitation, the algorithm divides all virtual machine images into seven groups.

Let us review the statistics characteristic value of each group. We calculate the similarities between the centroid image and other images within one group. For each group, we calculate their maximum similarity value, minimum similarity value, mean similarity value, and median similarity value. The maximum value and the minimum value represent the maximum and minimum similarities in one group; the standard deviation represents the similarity statistical dispersion; the mean value represents the degree of similarity among all the images in that group; and the median value could help us to find is the distribution skewed.

Fig. 8 shows ten experimental results and each plot represents one experiment result. The implication of Fig. 8 is similar to that of Fig. 7. Take the first plot in Fig. 8 (1) as an example, the mean similarity, max similarity, min similarity and median similarity value are very high (about 93%) and close, while the standard deviation value is about zero. That means the images in this group have very high similarity. From Fig. 8 (1) we can see that the images in the first, the second, and the forth groups have high similarities, since

Please cite this article as: J. Xu et al., Clustering-based acceleration for virtual machine image deduplication in the cloud environment, The Journal of Systems and Software (2016), http://dx.doi.org/10.1016/j.jss.2016.02.021

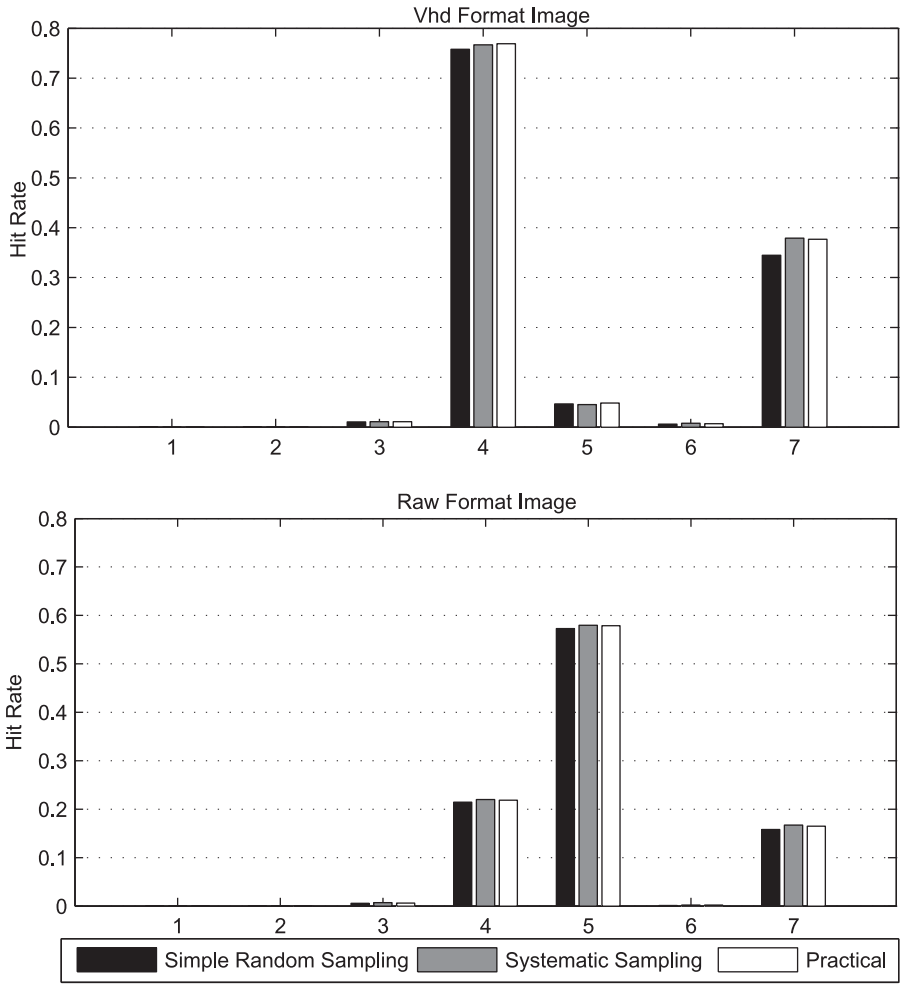


Fig. 7. Sample hit rate vs. practical duplication rate.

their maximum values, minimum values and median values are very high and their standard deviation is low. However, although the minimum value of the third group is very low (less than 5%), and its standard deviation is much higher than the other group, we still believe the whole image similarity of that group is high. That is because that both the mean value and the median value are very high. In the contrary, the images in the last three groups would have relatively lower similarities. Even so, most the image similarities are still larger than 10%. Some are even more than 20%.

The experiment discussed in this section shows that the clustering method we used in our local deduplication approach is acceptably effective.

4.3. Robustness

Robustness of the system is evaluated in the influence of sample size on the sampling method and the stability of the clustering method.

4.3.1. Influence of sample size on the sampling method

The influence of sample size is also studied in our work. We iterate different sample sizes from 100 to 1000 with an interval of 50. Further, we take 100 samples for each sample size tested and calculate its sample error. Then, we calculate an average and maximum error.

The results are shown in Fig. 9. The *x*-axis represents the sample size and the x-coordinate *x* represent that the sample size is *x*. The *y*-coordinate represents the sample error. There are four legends in this figure. The solid line with cross denotes the average error rate of SRS. The solid line with cycle denotes the average error rate of SS. The dotted line with rectangle denotes the maximum error rate of SRS. The dotted line with the star denotes the maximum error rate of SS. From Fig. 9, we can see that with the increase of the sample size, both the average sample error and the maximum sample error drop gradually. And at last, it levels off. The computed result according to the sample size formula in Section 3.1.3 is just in the stable stage. As we know that, the sample size is related to the sample accuracy. The bigger the sample size, the more the accurate approximate in sampling is. Our work gives a guiding opinion to choose the sample size. Sometimes, it may appear much larger in certain situation, especially when the image has a very high similarity to one group and have very low similarities to the others. We argue that all the debate would be wise after the event, because we do not know which group it belongs to.

4.3.2. Stability of the clustering method

Now let us review the content of the statistical characteristic value through examples. As we introduced in Section 4.2.2, there are ten different experimental results in Fig. 8. These experiments are based on the same data set and the same algorithm, but the results are different. The different results between the ten experiments can also validate our conjecture that the result of clustering algorithm is indeterminate. Nevertheless, we have the observation

Please cite this article as: J. Xu et al., Clustering-based acceleration for virtual machine image deduplication in the cloud environment, The Journal of Systems and Software (2016), http://dx.doi.org/10.1016/j.jss.2016.02.021

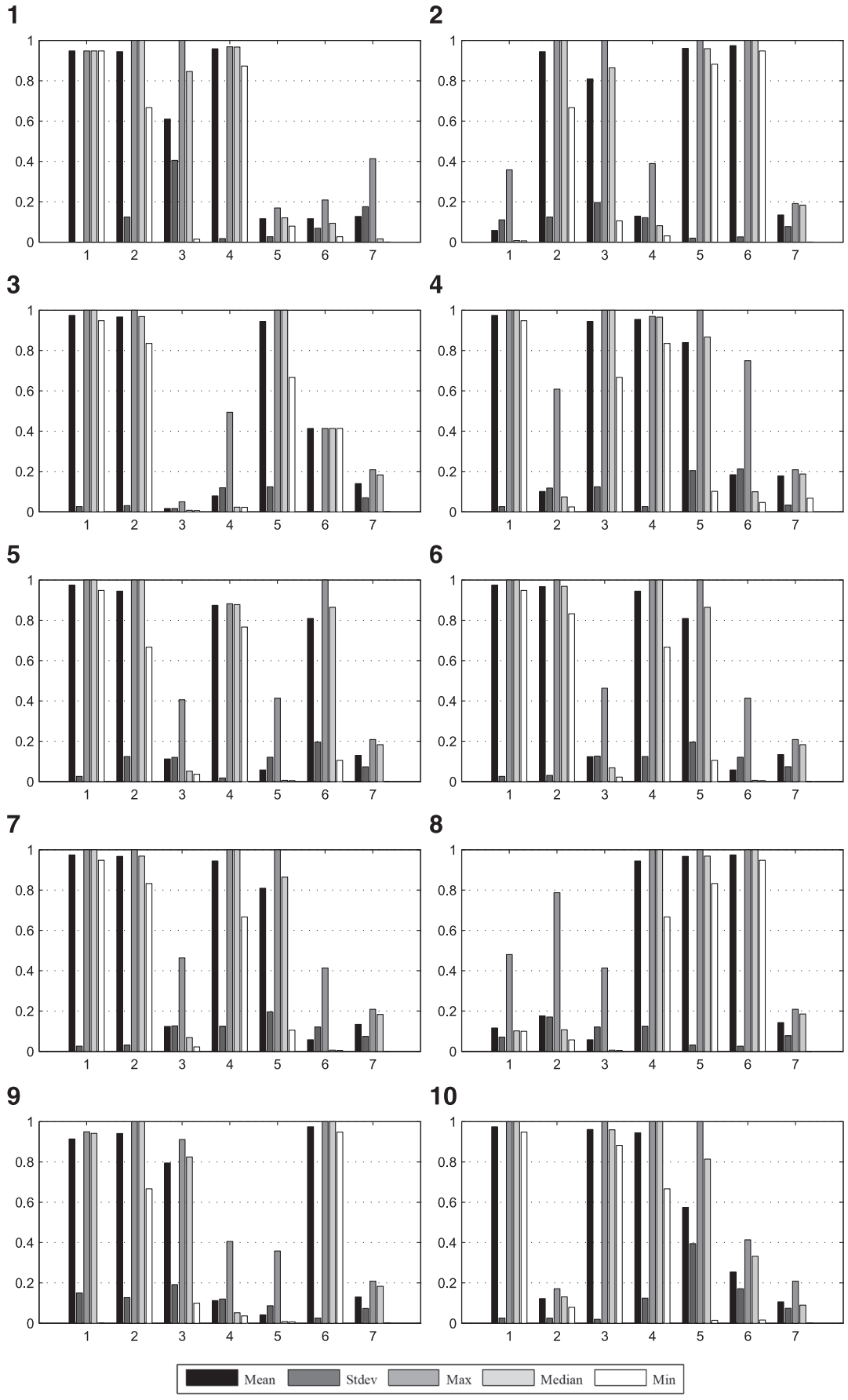


Fig. 8. Each group's statistical characteristic value in ten different experiments.

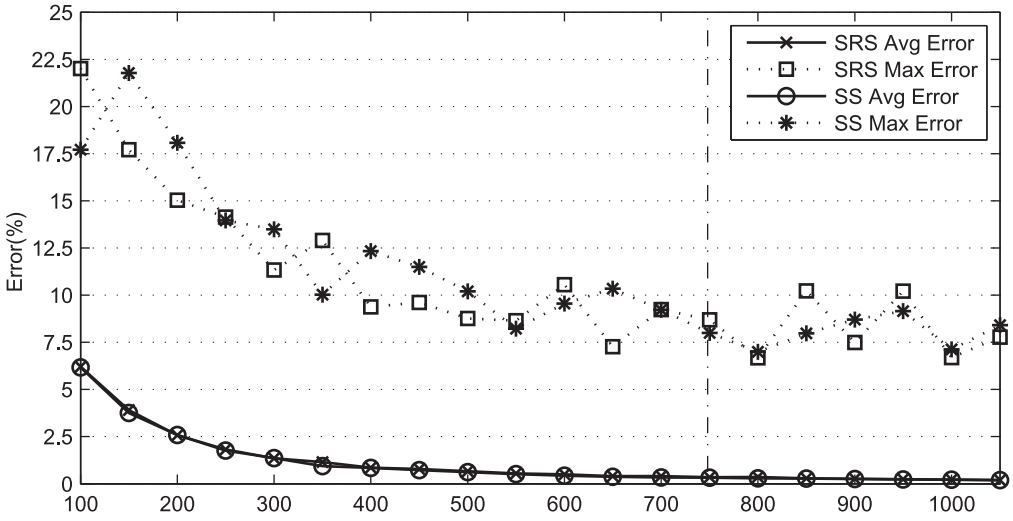


Fig. 9. Sample error with sample size.

that the images within a group also have high similarity according to the statistical characteristic value of each experiment. And the final deduplication rate introduced in Section 4.4.1 can also reflect the same thing.

The experiment discussed in this section shows that the clustering method we used in our local deduplication approach is acceptably robust.

4.4. Comparative evaluation

In VM image deduplication process, deduplication rate and operation time is the most two important factors, since the former has an effect on the storage space and the latter involves backup operation time window. In this section, both the two factors are evaluated. To evaluate the advantage of our approach, we first deduplicate all the virtual machine images without any classification, which is regarded as the global deduplication. Then we use our clustering algorithm to perform deduplication operation, which means to perform a local deduplication. In summary, complexity of the system is evaluated in the storage usage of the deduplication result and the time using deduplication operation.

4.4.1. Deduplication rate

Due to the indeterminacy of the clustering result, we run Crab for ten times under the given data set and calculate average value of the ten results. Then we compare the result with Venti and MSD.

Figs. 10 and 11 show the deduplication rate of our experiment. There are 5 bars in Fig. 10. The “Original” bar denotes the total VM image size. The “Inner Dedup” bar denotes the data size after inner deduplication (L2 deduplication in MSD). The other three bars denote the final data size after Crab, MSD and Venti deduplication.

As illustrated in Fig. 10, we treat the original image size as 100%. When the inner deduplication operation completes, the data set size becomes 24.2%. After that we separately use Crab, MSD, Venti to perform the inter deduplication. As the Venti approach is totally deduplication, it could achieve 9.2% compression rate. Compared with the Venti approach, our Crab deduplication approach based on clustering has marginal gap. And it could achieve 10.2% compression rate. Compare to the 90% compression ratio, the 1% difference is acceptable. In the next experiment, we will see that the 1% wasted space is trade for multiple times time saving. Compared with the MSD approach, the deduplication rate of the Crab approach has 0.1% improvements.

In Fig. 11, the *y*-coordinate represents the compression rate. Legend “Crab *x*” represents the *x*th experiment. From Fig. 11, we can see that the deduplication rates of each experiment are different but very close. The lowest deduplication rate is about 89.4%, and the highest deduplication rate is about 89.9% and the average deduplication rate is 89.74%. The difference between the Venti deduplication rate and the lowest Crab deduplication rate is less than 1.5%.

4.4.2. Deduplication cost

Let us take a look at the deduplication time for the new coming images. After the 584 images have been deduplicated and stored, we backup new images with different formats: the raw and vhd format. For each kind of image we separately use the Crab, MSD and Venti approach to perform backup operation. We run the three kinds of approach under different memory limitation. Fig. 12 illustrates the experimental results. In Fig. 12, the *x*-coordinate represent the available memory size, while the *y*-coordinate represents the backup time.

From both Fig. 12(a) and (b), we can see that the backup time of the Venti and MSD approach are reduced with the increase of the available memory size. However, the backup time of the Crab approach would not change much with the increase of the available memory size. That is because our Crab approach could regroup the image fingerprints to fit the available memory size. During the duplicated block identification process, only one disk access is needed to load the fingerprints. Thus, the Crab approach could achieve total memory index lookup and improve the backup performance. However, if the available memory is large enough (e.g., the available memory size is 1024MB in Fig. 12(a)), the MSD approach could achieve almost the same performance as our Crab approach. Otherwise, compared to the MSD approach and the Venti approach, our Crab approach would save considerable disk seeking time.

From the above experiment, we have the observation that our Crab approach can dramatically reduce the virtual machine image deduplication backup time in cost of slight additional storage space usage when the available RAM size is relatively small. It is a typical situation in the cloud environment for the resource competition of virtual machines.

To further evaluate our method, we measured the maximum memory requirement of the Venti, MSD and Crab approaches. The memory threshold of the Crab approach is set to 512MB. Fig. 13 illustrates the maximum memory requirement of the

Please cite this article as: J. Xu et al., Clustering-based acceleration for virtual machine image deduplication in the cloud environment, The Journal of Systems and Software (2016), http://dx.doi.org/10.1016/j.jss.2016.02.021

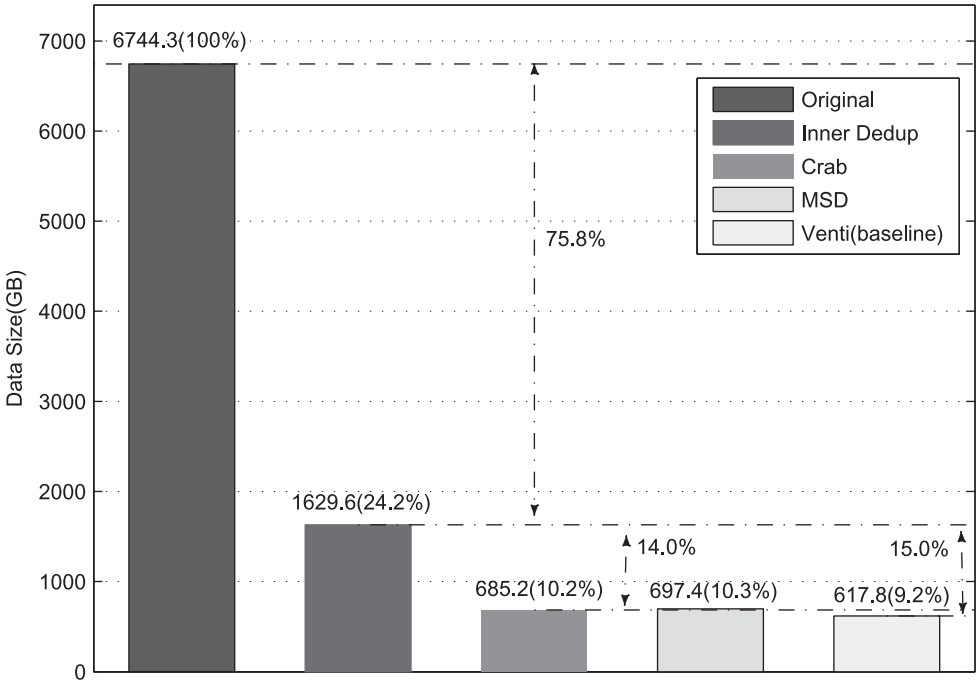


Fig. 10. Data size and deduplication rate.

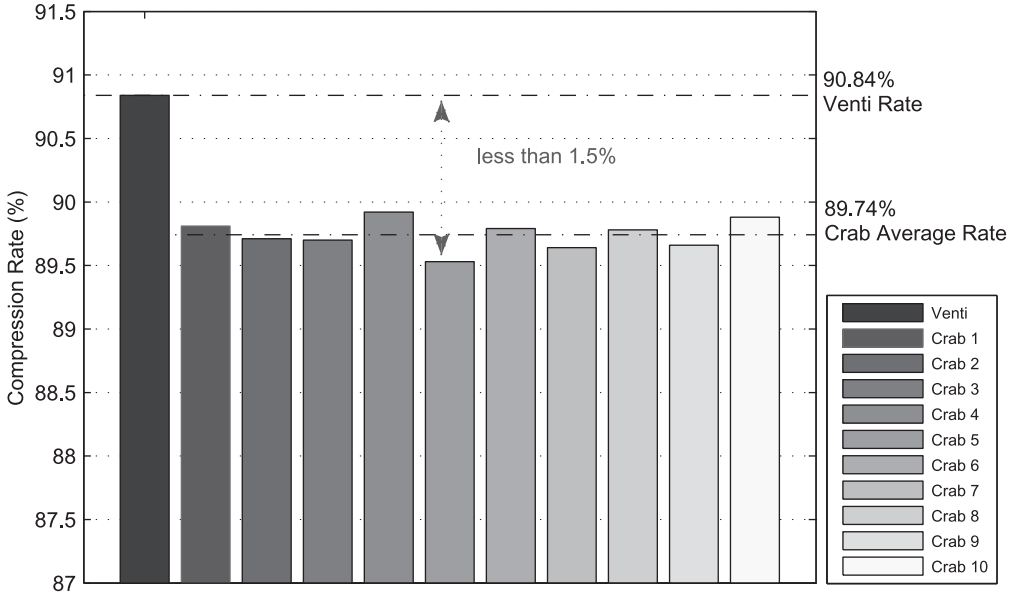
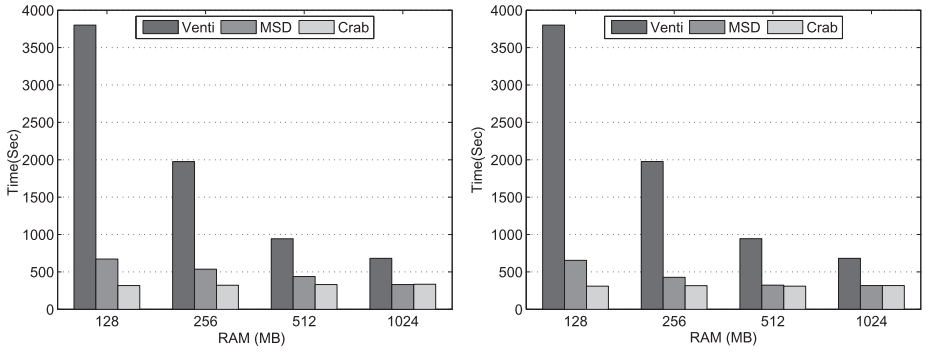


Fig. 11. Data compression rate comparison of ten tests.



(a) raw format (b) vhd format

Fig. 12. Different format image deduplication time.

Please cite this article as: J. Xu et al., Clustering-based acceleration for virtual machine image deduplication in the cloud environment, The Journal of Systems and Software (2016), http://dx.doi.org/10.1016/j.jss.2016.02.021

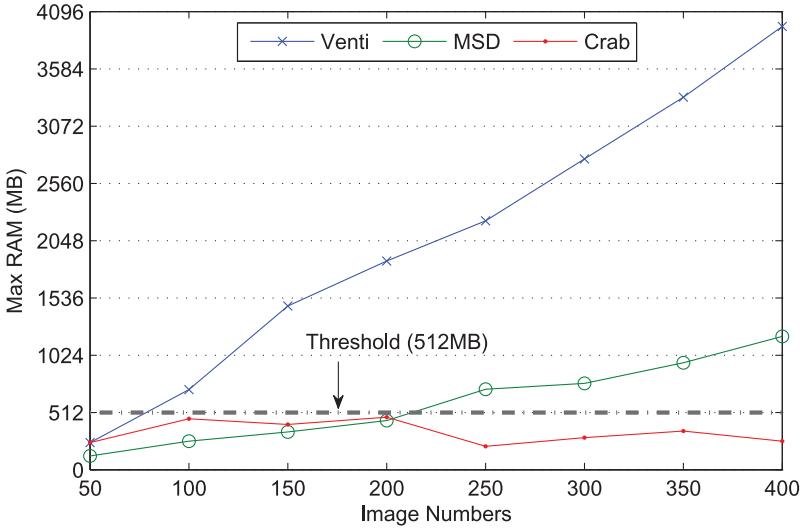


Fig. 13. Maximum memory requirements.

three approaches. From the figure, we can see that the memory requirements of Venti and MSD approaches become larger and larger with the increase of the backup image numbers. While the memory of the Crab approach is always under the threshold.

4.5. Threats to validity

In this work, we experiment with raw and vhd format image. However, there are many other kinds of image formats, such as vmdk, qcow and so on. In this sector, we will analyze the characteristic of each image format to clear the threats to validity of the observation in our experiment.

Raw format images are fixed-sized files, with one block for each block in the native host file systems. Initially, unused blocks are zero-filled. VHD format image is dynamic hard disk image. The file is at any as large as of the data actually written, together with the header and the footer. VMDK is a file format developed by VMware for its virtual appliance products, but is now an open format. It may have either fixed size or dynamic size. Here, qcow is a file format for disk image files used by QEMU. One of the main characteristics of qcow disk images is that files of this format grow when data are added. Different file formats may result in different experiment observations.

On the other hand, we can see that all these kind of images can be classified into two catalogs: flat mode and sparse mode (Tang, 2011). In flat mode, all sectors of the hard disk are stored in one flat file according to logical block addressing order. In sparse mode, a large hard drive can be created with only used space stored in the file. There may exist other modes and the observation in related tests can be different.

In this paper, we verify our work with raw and vhd formats. These two kinds image format are separately represented using flat mode and sparse mode. According to the above experiments, our method can work with both of these two different kinds of formats. It is because that we treated the virtual machine image as a whole file and do not care its inner content layout, we believe that our work would also work with other formats of images.

5. Related work

5.1. The state of the art of deduplication

Deduplication technology is accepted as a specialized technology to eliminate duplicate copies of repeating data for a set of files (Cox et al., 2002; Hunt et al., 1998; Bolosky et al., 2000). It can be used in various storage types and application scenarios which have different purpose and requirements. However, the deduplication system must be designed according to the specific requirements.

Chen et al. (2011) design a deduplication system within SSDs to increase the useful space and lifespan. Their challenges include limited resources (both computing and memory) and high data access performance. Clements et al. (2009) proposed a decentralized deduplication system DEDE that designed for SAN clustered file systems. The system is used for runtime virtual machine storage and aims not only shared storage for VM disks but also enable live migration, load balancing, and failover of VMs across hosts. Ng et al. (2011) proposed live deduplication file systems for virtual machine images which focus on the performance in importing and retrieving. Koller and Rangaswami (2010) use deduplication technology to improve the I/O performance by eliminating I/O operations and reducing the mechanical delays during I/O operations. Mao et al. (2014) use SSD-Assisted Read scheme to improve the read performance of deduplication-based storage system. There also exist many other studies (Zhang et al., 2010; Riteau et al., 2011; Bose et al., 2011; Owens and Wang, 2011) that use deduplication technology to speed up virtual machine migration.

Paulo and Pereira (2014) survey the existing block level deduplication work and classify the deduplication systems according to six criteria: granularity, locality, timing, indexing, technique, and scope.

5.2. Deduplication acceleration

There are many work (including this work) focus on the acceleration of deduplication. Zhu et al. (2008) use bloom filter algorithm (Bloom, 1970) to fast the duplicated chunk identification. They argued that their method could support one billion base segments in 1GB of memory. However, its false positive is about 2.17% to 2.40%. As we know that, in virtual machine image backup environment, a single chunk mistake will cause a serious data loss and even a damage of several images. So their work are not suitable for the VM image deduplication. In our work, we use MD5 to do the hash computing. The single collision rate of MD5 is 2^{-128}. When the numbers of blocks is 2^{15}(4KB per block, about 4ZB data), the collision rate is 10^{-9} (Hollingsworth and Miller, 1997). The lower single collision rate can ensure the data availability.

Please cite this article as: J. Xu et al., Clustering-based acceleration for virtual machine image deduplication in the cloud environment, The Journal of Systems and Software (2016), http://dx.doi.org/10.1016/j.jss.2016.02.021

Other work usually take advantage of data similarity and locality. Lillibridge et al. (2009) break up an incoming stream into relatively large segments and deduplicate each segment against only a few of the most similar previous segments with the sampling method. This could reduce the disk seek times and gives a performance improvement in index lookup. However, the ratio of the sampling space to the fingerprints size is fixed, since it depends on the ratio of the chunks size to segments size. That means the total sampling space will grow larger and larger with the growth of storage data. Finally, it will run out of memory. Nevertheless, our work could resolve this problem once and for all, because the principle our clustering could ensure the size of fingerprints in single group is always less than the available memory. Besides, our method works in the preprocessing stage, so it does not need segment comparison during the deduplication stage which can lift the processing speed. Zhang et al. (2012) use the locality characteristic to do VM snapshot deduplication. They classify the deduplication of VM snapshot into two categories: inner-VM and cross-VM. They use distributed multilevel deduplication to conduct segment level and block level inner VM deduplication. Cross-VM deduplication is performed by excluding a small number of popular common data blocks from being backed up. However, their work can only work with virtual machine image snapshot. Our method can work with image, snapshot and template. Moreover, our work focus on the preprocessing method before the deduplication and we use the most regular deduplication method introduced in (Quinlan and Dorward, 2002) in deduplication stage while Zhang *et al.* focus on deduplication process. Xia et al. (2011) and Xia et al. (2014) believe that many existing deduplication work perform poorly in certain situation for they only consider the locality or the similarity. So, in their work, the join the two dimensions together to improve the overall performance of deduplication. This work also focus on deduplication stage and complicates the deduplication process while our work focus on preprocess stage. The key technique of our work does not conflict, and it is possible to merge our work together to further improve the performance.

In summary, the biggest differences between our work and the existing acceleration work are that we focus on the preprocessing phase while the other work focus on the deduplication phase and there is no conflict with the key steps. In particular, some of the existing work can be integrated into our work to further speed up the deduplication process. Based on this consideration, we do not compare the existing acceleration method with ours in this paper.

6. Conclusion

The deduplication technology can save a huge storage space in virtual machine image backup in a cloud environment. However, it may result in a heavy performance degradation to the applications running on the hosted virtual machine. In our environment, the application performance could be reduced by 15% to 20%. In our previous work, we have exploited the feature of the virtual machine image and introduced a key improvement in deduplication technology aiming at reducing the resource overhead in virtual machine image deduplication approach. In this work, we revisit various common scenarios of VM image, employ clustering as the key technology to local duplication, and emphasize timing issuers in particular. Experimental results show that it will accelerate the backup process with a little increment of disk space usage.

Furthermore, VM deduplication backup in cloud environment is complex. In this work, we focus on the mode of “one to one”, which represents one backup storage serves for one runtime storage. However, this mode simplifies the problem complexity. In practice, a backup storage server often serves multiple runtime storage, which is symbolized in “many to one” mode. That will cause the serious concurrency conflict (Wei et al., 2015) and comprehensive backup strategy selection (Wang et al., 2014). In the future, we will focus to resolve the many to one deduplication problem.

Acknowledgments

This work was supported by the National Key Basic Research Program of China (project no. 2014CB340702), the National Natural Science Foundation of China (project no. 61379045 and 61402450), and Beijing Natural Science Foundation (project no. 4154088).

References

Bhagwat, D., Eshghi, K., Long, D.D., Lillibridge, M., 2009. Extreme binning: Scalable, parallel deduplication for chunk-based file backup. In: Proceedings of the IEEE International Symposium on Modeling, Analysis and Simulation of Computer and Telecommunication Systems, (MASCOTS' 09). IEEE, pp. 1–9.

Bloom, B.H., 1970. Space/time trade-offs in hash coding with allowable errors. Commun. ACM 13 (7), 422–426.

Bolosky, W.J., Corbin, S., Goebel, D., Douceur, J.R., 2000. Single instance storage in windows 2000. In: Proceedings of the Fourth USENIX Windows Systems Symposium. Seattle, WA, pp. 13–24.

Bose, S.K., Brock, S., Skeoch, R., Rao, S., 2011. Cloudspider: Combining replication with scheduling for optimizing live migration of virtual machines across wide area networks. In: Proceedings of the Eleventh IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (CCGrid). IEEE, pp. 13–22.

Chen, F., Luo, T., Zhang, X., 2011. CAFTL: A content-aware flash translation layer enhancing the lifespan of flash memory based solid state drives. In: Proceedings of the Ninth USENIX Conference on File and Storage Technologies (FAST), vol. 11.

Clements, A.T., Ahmad, I., Vilayannur, M., Li, J., et al., 2009. Decentralized deduplication in san cluster file systems. In: Proceedings of the 2009 USENIX Annual Technical Conference, pp. 101–114.

Cox, L.P., Murray, C.D., Noble, B.D., 2002. Pastiche: Making backup cheap and easy. ACM SIGOPS Oper. Syst. Rev. 36 (SI), 285–298.

Eastlake, D., Jones, P., Us secure hash algorithm 1 (sha1).

Fu, Y., Jiang, H., Xiao, N., Tian, L., Fang, L., 2011. AA-Dedupe: An application-aware source deduplication approach for cloud backup services in the personal computing environment. In: Proceedings of the 2011 IEEE International Conference on Cluster Computing (CLUSTER). IEEE, pp. 112–120.

Hollingsworth, J.K., Miller, E.L., 1997. Using content-derived names for configuration management. In: ACM SIGSOFT Software Engineering Notes, vol. 22. ACM, pp. 104–109.

Hunt, J.J., Vo, K.-P., Tichy, W.F., 1998. Delta algorithms: An empirical analysis. ACM Trans. Softw. Eng. Methodol. (TOSEM) 7 (2), 192–214.

ISCAS,. Once cloud platform. <http://www.once.com.cn/OncePortal/oncecloud> (accessed 17.12.14.).

Jayaram, K., Peng, C., Zhang, Z., Kim, M., Chen, H., Lei, H., 2011. An empirical analysis of similarity in virtual machine images. In: Proceedings of the Middleware 2011 Industry Track Workshop. ACM, p. 6.

Jin, K., Miller, E.L., 2009. The effectiveness of deduplication on virtual machine disk images. In: Proceedings of SYSTOR 2009: The Israeli Experimental Systems Conference. ACM, p. 7.

Koller, R., Rangaswami, R., 2010. I/o deduplication: Utilizing content similarity to improve i/o performance. ACM Trans. Storage (TOS) 6 (3), 13.

Lillibridge, M., Eshghi, K., Bhagwat, D., Deolalikar, V., Trezis, G., Camble, P., 2009. Sparse indexing: Large scale, inline deduplication using sampling and locality. In: Fast, vol. 9, pp. 111–123.

Mao, B., Jiang, H., Wu, S., Fu, Y., Tian, L., 2014. Read-performance optimization for deduplication-based storage systems in the cloud. ACM Trans. Storage (TOS) 10 (2), 6.

Min, J., Yoon, D., Won, Y., 2011. Efficient deduplication techniques for modern backup operation. IEEE Trans. Comput. 60 (6), 824–840.

Muthitacharoen, A., Chen, B., Mazieres, D., 2001. A low-bandwidth network file system. In: Proceedings of the 2001 ACM SIGOPS Operating Systems Review, vol. 35. ACM, pp. 174–187.

Ng, C.-H., Ma, M., Wong, T.-Y., Lee, P.P., Lui, J., 2011. Live deduplication storage of virtual machine images in an open-source cloud. In: Proceedings of the Twelfth International Middleware Conference. International Federation for Information Processing, pp. 80–99.

Owens, R., Wang, W., 2011. Non-interactive OS fingerprinting through memory de-duplication technique in virtual machines. In: Proceedings of the IEEE Thirty International Performance Computing and Communications Conference (IPCCC). IEEE, pp. 1–8.

Paulo, J., Pereira, J., 2014. A survey and classification of storage deduplication systems. ACM Comput. Surv. (CSUR) 47 (1), 11.

Policroniades, C., Pratt, I., 2004. Alternatives for detecting redundancy in storage systems data. In: Proceedings of the USENIX Annual Technical Conference, General Track, pp. 73–86.

Quinlan, S., Dorward, S., 2002. Venti: A new approach to archival storage. In: FAST, 2, pp. 89–101.

Riteau, P., Morin, C., Priol, T., 2011. Shrinker: Improving live migration of virtual clusters over wans with distributed data deduplication and content-based addressing. In: Euro-Par 2011 Parallel Processing. Springer, pp. 431–442.
Rivest, R., The md5 message-digest algorithm.
Tang, C., 2011. FVD: A high-performance virtual machine image format for cloud. In: Proceedings of the 2011 USENIX Annual Technical Conference.
Tolia, N., Kozuch, M., Satyanarayanan, M., Karp, B., Bressoud, T.C., Perrig, A., 2003. Opportunistic use of content addressable storage for distributed file systems. In: Proceedings of the USENIX Annual Technical Conference, General Track, pp. 127–140.
Wang, Y., Tang, S., Tan, C.C., 2014. Elastic data routing in cluster-based deduplication systems. In: Proceedings of the 2014 IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS). IEEE, pp. 117–118.
Wei, H., De Biasi, M., Huang, Y., Cao, J., Lu, J., 2015. Verifying pipelined-ram consistency over read/write traces of data replicas. IEEETrans. Parallel Distrib. Syst.
Won, Y., Kim, R., Ban, J., Hur, J., Oh, S., Lee, J., 2008. Prun: Eliminating information redundancy for large scale data backup system. In: Proceedings of the 2008 International Conference on Computational Sciences and Its Applications (ICCSA'08). IEEE, pp. 139–144.
Xia, W., Jiang, H., Feng, D., Hua, Y., 2011. Silo: A similarity-locality based near-exact deduplication scheme with low ram overhead and high throughput. In: Proceedings of the 2011 USENIX Annual Technical Conference.
Xia, W., Jiang, H., Feng, D., Tian, L., 2014. Combining deduplication and delta compression to achieve low-overhead data reduction on backup datasets. In: Proceedings of the 2014 Data Compression Conference (DCC). IEEE, pp. 203–212.
Xu, J., Zhang, W., Ye, S., Wei, J., Huang, T., 2014. A lightweight virtual machine image deduplication backup approach in cloud environment. In: Proceedings of the 2014 IEEE Thirty-eighth International Annual Computer Software and Applications Conference (COMPSAC). IEEE.
Zhang, W., Tang, H., Jiang, H., Yang, T., Li, X., Zeng, Y., 2012. Multi-level selective deduplication for VM snapshots in cloud storage. In: Proceedings of the Fifth IEEE International Conference on Cloud Computing (CLOUD). IEEE, pp. 550–557.
Zhang, W., Yang, T., Narayanasamy, G., Tang, H., 2013. Low-cost data deduplication for virtual machine backup in cloud storage. In: Proceedings of the Fifth USENIX Workshop on Hot Topics in Storage and File Systems. USENIX.
Zhang, X., Huo, Z., Ma, J., Meng, D., 2010. Exploiting data deduplication to accelerate live virtual machine migration. In: Proceedings of the 2010 IEEE International Conference on Cluster Computing (CLUSTER). IEEE, pp. 88–96.
Zhu, B., Li, K., Patterson, R.H., 2008. Avoiding the disk bottleneck in the data domain deduplication file system. In: Fast, vol. 8, pp. 1–14.

Jiwei Xu is a Ph.D. candidate of the University of Chinese Academy of Science. His current research interests are software engineering and distributed computing.

Wenbo Zhang received the Ph.D. degree from the Graduate School of Chinese Academy of Science. He is a professor at the Technology Center of Software Engineering, Institute of Software, Chinese Academy of Sciences. His current research software engineering and distributed computing.

Zhenyu Zhang received the Ph.D. degree from the University of Hong Kong. He is an associate professor at the State Key Laboratory of Computer Science, Institute of Software, Chinese Academy of Sciences. His current research interests are program debugging and testing for software and systems, and the reliability issues of web-based services and cloud-based systems. He has published research results in venues such as Computer, ICSE, FSE, ASE and WWW.

Tao Wang received the Ph.D. degree from the Graduate School of Chinese Academy of Science. He is a research assistant at the Technology Center of Software Engineering, Institute of Software, Chinese Academy of Sciences. His current research software engineering and distributed computing.

Tao Huang received the Ph.D. degree from the Graduate School of Chinese Academy of Science. He is a professor at the State Key Laboratory of Computer Science, Institute of Software, Chinese Academy of Sciences. His current research software engineering and distributed computing.

Please cite this article as: J. Xu et al., Clustering-based acceleration for virtual machine image deduplication in the cloud environment, The Journal of Systems and Software (2016), http://dx.doi.org/10.1016/j.jss.2016.02.021