An Empirical Analysis of Similarity in Virtual Machine Images

K. R. Jayaram*, Chunyi Peng*, Zhe Zhang†, Minkyong Kim†, Han Chen†, Hui Lei†
IBM Thomas J. Watson Research Center
Hawthorne, NY, USA

ABSTRACT

To efficiently design deduplication, caching and other management mechanisms for virtual machine (VM) images in Infrastructure as a Service (IaaS) clouds, it is essential to understand the level and pattern of similarity among VM images in real world IaaS environments. This paper empirically analyzes the similarity within and between 525 VM images from a production IaaS cloud. Besides presenting the overall level of content similarity, we have also discovered interesting insights on multiple factors affecting the similarity pattern, including the image creation time and the location in the image's address space. Moreover, we found that similarities between pairs of images exhibit high variance, and an image is very likely to be more similar to a small subset of images than all other images in the repository. Groups of data chunks often appear in the same image. These image and chunk "clusters" can help predict future data accesses, and therefore provide important hints to cache placement, eviction, and prefetching.

Categories and Subject Descriptors

D.4.3 [**Operating Systems**]: File Systems Management Maintenance; D.4.2 [**Operating Systems**]: Storage Management Storage Hierarchies; C.2.4 [**Computer Communication Networks**]: Distributed Systems Distributed Applications

General Terms

Experimentation

*K. R. Jayaram (jayaram@purdue.edu) and Chunyi Peng (chunyip@cs.ucla.edu) were interns at IBM when this research was conducted. K. R. Jayaram is also affiliated with Purdue University and Chunyi Peng is also affiliated with the University of California, Los Angeles.
†{zhezhang,minkyong,chenhan,hlei} @us.ibm.com

Keywords

virtual machine, similarity, caching, distribution, IaaS

1. INTRODUCTION

The increased popularity of Infrastructure as a Service (IaaS) clouds has resulted in an explosion of the number of VM images. Amazon Elastic Compute Cloud (EC2), for example, has 6521 *public* VM images[2] (data on *private* EC2 VM images is unavailable). This has created crucial challenges in managing cloud environments. Today's production clouds widely use *cloud management systems* to automate and ease the administrative tasks of IaaS providers, examples of which include IBM Tivoli Virtual Deployment Engine (VDE) Beta[7], VMWare's VSphere[13], and Eucalytpus[10]. Many critical responsibilities of cloud management systems are tightly related to VM images, including (1) efficient and reliable storage of VM images; (2) low-latency retrieval of VM images for instantiation in response to customer requests; (3) prompt capture of VM images from running instances; and (4) fast transfer of VM image data across cloud servers to facilitate live VM migration.

Therefore, *one* key factor impacting the performance of cloud management systems is how to reduce the amount of VM image data to store and transfer. This, in turn, depends on the similarity in VM images. The similarity in VM images can be leveraged to reduce the total amount of image data to store. Deduplication techniques have been proposed to identify and remove duplicate data blocks when handling a collection of images [5, 8]. This can lead to significant space saving in image storage servers, and enable live VM instances to share data blocks. To efficiently design such deduplication techniques, it is essential to understand how much similarity can be found in real-world cloud environments under different schemes to divide images into chunks, and how the similarity changes when the collection of images grows.

VM image similarity can also be leveraged for content-aware caching and prefetching of image data. Because a typical VM image contains multiple gigabytes of data, it takes long time to copy an image from the image storage server to the target hypervisor host to instantiate a VM instance. By having a cache at the hypervisor level, multiple VM instances can share common parts of VM images. One example is the delta deployment mechanism in the Mirage [3] system, which keeps popular images in cache and only fetches a *diff* from the storage server when an image is requested that is not in the cache. In designing cache placement, eviction, and prefetching schemes, it is important to determine "hints" that indicate high chance of content similarity. For example, if the similarity is correlated to the position in the image, and the beginning portion of images are more likely to be common to each other, we can give higher priority to these chunks to stay in the cache. Another effective hint is the "affinity" among chunks, i.e., which chunks often appear together in the similar VM image? For techniques that cache entire images, such as delta deployment, it is also highly useful to understand which images are similar to each other, in that they contain a large number of common data chunks.

VM images are similar due to several factors. First, several public virtual appliance libraries contain VM images with the same or similar operating systems. For example, out of the 18 OpenSUSE VM images in the EC2 public library, 8 contain the OpenSUSE version 11.4 OS and 5 contain OpenSUSE 11.3 [2]. Next, of the images containing the same OS, many contain the same or similar applications. For example, of the 45 VM images in the Turnkey library containing Ubuntu 10.04.1 LTS, 20 images contain the Apache2 web server [1]. Finally, many users create *private* images by slightly modifying public images, e.g. with passwords, public/private keys and software patches.

The main challenge in analyzing content similarity, just like deduplication in storage systems, is to identify (large) chunks that are common to two or more images. To that end, similarity detection techniques can be classified into two types: (1) *white-box* techniques, which typically understand the file system structure of VM images, and compute MD5/SHA hashes of files to identify common files, and (2) *black-box* techniques, which as the name suggests, do not understand the semantics of VM images. Such techniques typically break a VM image into fixed/variable sized chunks, and identify common chunks by using their hash keys. The advantage of white-box techniques is that the knowledge of an image's semantics can lead to increased similarity and deduplication, as revealed in [6], but the disadvantage is that whole images should either be reconstructed at the hypervisor or the hypervisor should be modified to handle specialized image formats. In particular, it is difficult for an image chunk cache to maintain the file system semantics information. For this reason, we choose to use black-box techniques in our study.

The main goal of this paper is to conduct an in-depth and empirical evaluation of the similarity in VM images used in a realistic cloud environment. Through this evaluation, we provide insights on the overall level of similarities in VM image data as well as different factors affecting the similarity. Compared with existing studies on this topic, we make two major contributions:

1. We analyze a *large collection* of VM images (525 in total) from a *production* data center, which reflects a snapshot of the image repository. Many of these images were created by real cloud users, and reflect their usage pattern. Previous work on VM image similarity [5, 8] either analyzes "master" or "template" VM images downloaded from the web, or randomly selected from the image repository. We also quantify the effect of a wide variety of chunking schemes with realistic chunk sizes. The chunk sizes we select are based on the requirements of cloud management systems.

2. Beyond measuring the overall level of similarity, we study the correlation of similarity with many factors that could serve as hints to cache management. We have discovered distinct similarity patterns in different regions of the image address space. We have also identified large variance in the pair-wise similarity between data chunks, which indicates the existence of chunk *clusters* that are likely to appear together in the same image. Image pairs that are highly similar to each other have also been observed, which serves a rationale for deploying image-level caches.

2. RELATED WORK

Content similarity among files, file systems, and VM disk images is important in various aspects of system management, and has hence been widely studied by researchers.

In [11], through extensive analysis, the authors demonstrate that popular media files do exhibit similarity among each other. By leveraging similarity and allowing clients to download from similar sources, download times are significantly improved. A novel MR(multi-resolution) handprinting mechanism has been proposed in [12] to further enhance the efficiency of similarity detection. Since the goal of this body of work is to facilitate the transfer of entire media files, the analysis has been focused on detecting the overall level of similarity among files without covering detailed similarity patterns. Moreover, the similarity results for media files are not directly applicable in the context of VM image files.

In a recent paper [9], Meyer et al. present a deduplication study of file systems from 857 desktop computers at Microsoft over a 4 weeks time span. This work compares chunk-based and file-based methods, and finds that whole-file deduplication can achieve 67% of the space saving of block-level deduplication for live file systems, and 87% for backup storage respectively. Another interesting insight is that 96% of files are entirely linear in the block address space. Since the actual value of any non-duplicated hash is irrelevant to this work, a novel two-pass mechanism has been used to save machine time in post processing. In the first pass, all hash values that appear more than once are inserted into a Bloom filter [4]. In the second pass, all hash values are compared to the Bloom filter so that all unique values are removed. The insights revealed in this paper are highly relevant to our work, and the post processing mechanism can be leveraged in our future study of larger data sets. However, the content data studied in this paper consists of running or backed up file systems, rather than disk images of virtual machines. Moreover, the file systems studied are from desktop computers running the Windows OS, while our images are installed with the Linux OS and various Linux software packages.

Deduplication for VM image files has also been studied. In [5], a block-level deduplication mechanism is proposed for VMware's VMFS file system. It leverages similarity among VM images to avoid storing redundant data blocks. This work focuses on techniques to efficiently detect and merge duplication, and is orthogonal and complementary to our study.

In [8], a similarity study is conducted to evaluate the efficacy of deduplication using both fixed size chunking and Rabin fingerprinting schemes to store a set of 52 VM images downloaded from online repositories. It is shown that VM images share a lot of common data if they use the same OS and just install different applications. This study also finds that fixed-size chunking schemes lead to similar deduplication ratio as their variable-size counterparts. Our study differs from this work in two major ways. First, we study a large number of VM images from a production cloud data center, which represents realistic duplication and growth patterns of image content data. Most importantly, it captures the behavior of real users in creating customized images. Second, our study goes beyond deduplication for effective storage. Our goal is to *also* provide insights for the design of VM image cache on hypervisor hosts. Therefore, we have analyzed the correlation of content similarity with a number of factors, including the region in a VM image's address space, and clusters of similar chunks and images.

3. EMPIRICAL ANALYSIS

3.1 Methodology and Data Selection

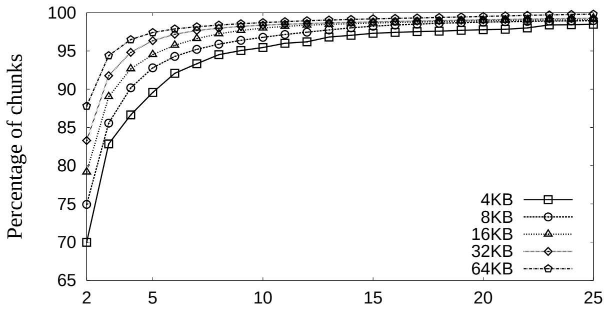
Our analysis is based on the VM image repository of a production cloud data center. The set of 525 VM images we study reflect the snapshot of the repository on 12th June 2010. All the 525 images are based on Linux or IBM AIX operating systems. They are installed with typical server applications including Apache Tomcat, Hadoop, and so forth. Each VM image contains varying amounts of zero-filled blocks, depending partially on the size of the image and the OS/applications installed in it. In the 525 VM images used in our analysis, the proportion of zero-filled blocks varies between 35% and 55%. Since zero-filled blocks are not actually stored on disks or caches, we ignore them in the results reported in this paper. We divide each image file into chunks using different chunking schemes, and hash each chunk. In total, we examine eight chunking schemes: (a) Fixed size chunking with MD5 hashing of chunks (chunk sizes were 4KB, 8KB,16KB, 32KB and 64KB) and (b) Rabin fingerprinting (where the *minimum* chunk sizes are 8KB, 16KB and 32KB).

Rabin fingerprinting is a method for implementing public key fingerprints using polynomials over a finite field, and has, among others two parameters r_s and r_v. As opposed to fixed size chunking and hashing, Rabin fingerprinting selects blocks not based on a specific offset but rather by some property of the block contents. It slides a window of size r_s a over the file, computing the Rabin fingerprint of the window. When the low r_v bits of the checksum equals a "special value", the fingerprinting scheme ends the current block and begins a new one. This has the effect of shift-resistant variable size blocks.

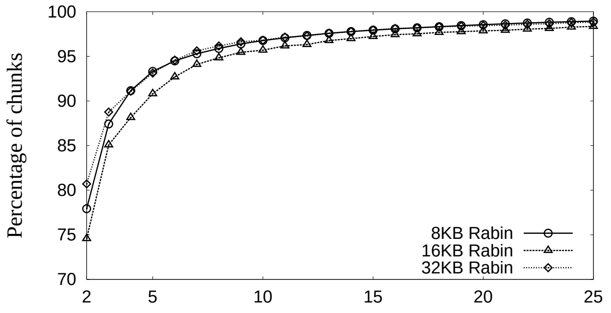
Several implementations of Rabin fingerprinting allow the user to specify maximum and the *desired* average size of the chunk. Specifying the minimum chunk size is useful, especially in VM management middleware where the overhead of handling very small chunks can be large. Specifying the maximum chunk size can also be useful, e.g., in caching middleware which has to allocate cache frames. But, intuitively, specifying the maximum chunk size leads to less effective Rabin fingerprints, because it overrides normal chunk boundaries. To analyze the effect of specifying a maximum chunk size, we used a maximum chunk size of 8.5KB in the 8KB Rabin fingerprinting scheme, and did not specify any maximum chunk size in the 16KB and 32KB schemes.

Our analysis is designed to reveal several important insights in similarities among VM images. First, we measure the overall level of similarity within and between VM images. Second, we present how the similarity changes when the VM image repository grows over time. Next, we correlate the duplication with the position of the chunk in the image file. Finally, we detect groups of images that are "clustered" together in terms of containing large number of similar chunks, and groups of chunks that often appear in the same image.

3.2 Overall Chunk Duplication



(a) Chunk duplication CDF with fixed size chunking schemes



(b) Chunk duplication CDF with Rabin fingerprinting schemes

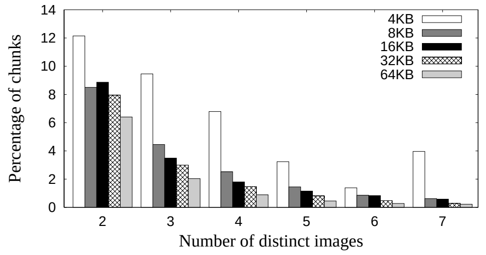
Figure 1: Cumulative distribution function (CDF) of the number of duplications for each distinct non-zero chunk, under different chunking schemes

To evaluate the overall level of duplication, we generate the cumulative distribution function (CDF) of the number of times that a chunk appears *within and across* VM images, as illustrated in Figure 1. For each value x on the X-axis, the graph in Figure 1 shows the percentage of *non-zero* chunks that appear fewer than x times.

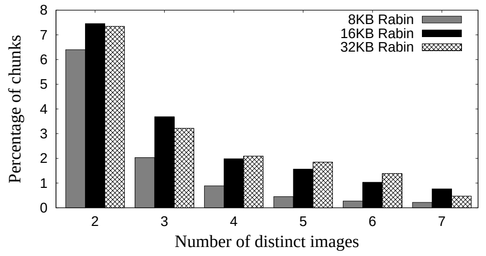
From Figure 1(a) which corresponds to fixed size chunking, we observe two trends: (1) Even fixed size chunking is able to detect a significant level of deduplication. For example, with a 4KB chunk size, approximately 30% of chunks occur at least twice, and around 12% of chunks are duplicated at least 5 times. (2) The chunk duplication level decreases quickly with chunk size. With the 4KB chunk size, around 70% chunks appear only once in the 525 VM images, and around 78% chunks appear fewer than 5 times. However, with the 32KB chunk size, the corresponding numbers are approx. 83% and 97%.

One would expect that Rabin fingerprinting increases the duplication level when compared to fixed size chunking. However, from Figure 1(b), we observe that Rabin fingerprinting degrades the duplication level for the 8KB chunk size, and increases the duplication level for the 16KB and 32KB chunk sizes. Using Rabin fingerprinting with the 8KB chunk size, 22.1% of chunks appear at least twice, as compared to 25.1% with fixed 8KB chunks. Rabin fingerprinting with the 16KB chunk size causes 25.4% of chunks to appear at least twice, as compared to 20.8% with fixed 16KB chunks. This is because, for 16KB and 32KB chunking schemes, Rabin fingerprinting of the VM images in our data set produced chunks with standard deviations of 12.58KB and 9.74KB respectively, whereas specifying a maximum size produced chunks with a standard deviation of 0.01KB for the 8KB chunking scheme. Thus, we find that increasing the variability of the chunk size increases the level of duplication in Rabin fingerprinting when compared to fixed size chunking. Based on these results, we conclude that for Rabin fingerprinting to outperform fixed size chunking, the maximum chunk size should either not be specified, or should be set to a large value (e.g. minimum + 2 × std. deviation) when absolutely necessary (e.g. in caching middleware).

3.3 Duplication Across VM Images



(a) Inter-image duplication with fixed size chunking

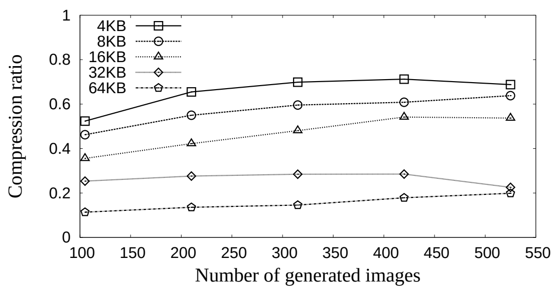


(b) Inter-image duplication with Rabin fingerprinting

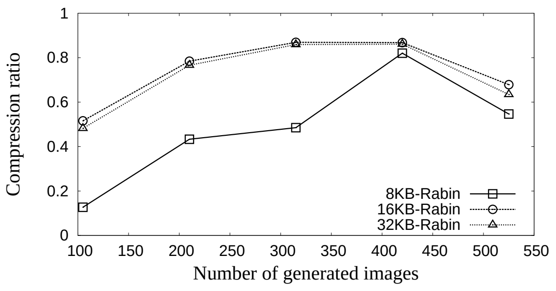
Figure 2: Duplication between VM images

In Figure 1, we analyzed duplication of chunks *within and between* VM images. Figure 2 quantifies the percentage of chunks that are duplicated *between* VM images. Figure 2 plots the percentage of *unique non-zero* chunks that are common to x VM images, where $1 \leq x \leq 7$. The percentage of chunks common to 8 VM images or more is too low (below 0.5%), and therefore not included in Figure 2. We would like to emphasize that Figure 2 is a popularity distribution, and not a CDF, the chunks that are common to x images are not included in calculating the percentage for $x+1$. We observe from Figure 2(a) that: (1) Even with fixed size chunking, up to 12.15% of chunks are common to 2 images, and up to 6.79% of chunks are common to 4 images. (2) For each value x on the X-axis, with fixed size chunking, the percentage of chunks common to x VM images decreases as the size of the chunk increases from 4KB to 64KB. (3) Except for 4KB chunks, for other chunk sizes, the percentage of chunks common to x images decreases as x increases. From Figure 2(b), we observe a similar trend to Figure 1. When compared to fixed size chunking, using Rabin fingerprinting increases the percentage of chunks common to two images or more, only when the maximum chunk size is not specified, i.e. in the case of our 16KB and 32KB Rabin fingerprinting schemes.

3.4 Compression Ratio with the Growth of Image Repository



(a) Compression ratio with fixed size chunking schemes



(b) Compression ratio with Rabin fingerprinting schemes

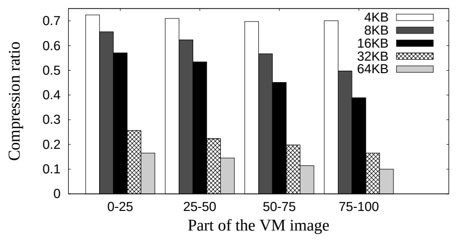
Figure 3: The overall compression ratio changing with the growth of the image repository

In this subsection, we analyze the impact of similarity on the size of the repository used to store VM images. Compression ratio for a set of VM images is defined as $1 - \frac{|Unique\ chunks|}{|Non-zero\ chunks|}$.

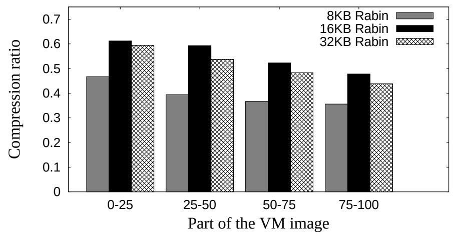
Figure 3 illustrates the change in compression ratio as more images are added to the repository. We measure the compression ratio at five evenly-spaced points during the growth of the repository. The image repository we use provides the creation time for each image, enabling us to preserve the order of addition of images in the results presented in Figure 3. From Figure 3(a), we observe that fixed size chunking performs relatively well with compression ratios of up to 0.69 in the case of 4KB chunking scheme. Even 8KB and 16KB fixed size chunking schemes attain compression ratios above 0.5. Following the trends in overall deduplication, compression ratio decreases with increasing chunk size.

We observe from Figure 3(b) that the compression ratios of 16KB and 32KB Rabin fingerprinting schemes are higher than their fixed-size counterparts, but that of the 8KB scheme is lower. Also, compression ratio does not increase monotonically as the number of VM images in the repository is increased, for example, between images 421-525 the compression ratio decreases for both fixed and Rabin chunking schemes. The trend in compression ratio depends on the number of unique chunks contained in the images added.

3.5 Similarity at Different Positions in Image



(a) Compression ratio with fixed size chunking



(b) Compression ratio with Rabin fingerprinting schemes

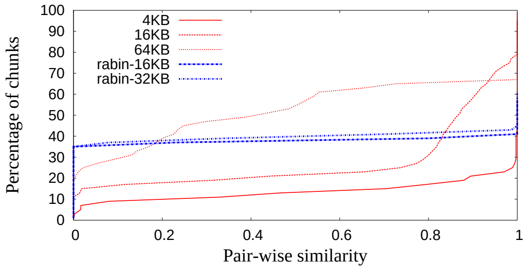
Figure 4: Compression ratio in different regions of the image address space. "0-25" on the X-axis means the first 25% of the image.

In this section, we examine whether the overall similarity level varies while considering chunks in different portions of images. For this experiment, we divide non-zero chunks into four bins, where each bin corresponds to chunks occurring in a quarter of the image. A chunk may belong to more than one bin. We then measure the compression ratio considering the chunks in each bin, corresponding to all the chunking schemes. The results are shown in Figure 4. For both fixed-size chunking and Rabin fingerprinting, we observe that the compression ratio is the highest among chunks in the first quarter of the images, and decreases towards the later portions of the image. This decrease, however, is very low in the case of the 4K fixed size chunking scheme – the compression ratio is approximately 0.7 among chunks in all bins. This indicates that with larger chunk sizes, which are more applicable to caches, it is promising to use position-based cache eviction mechanisms and give earlier portions of an image higher priority to stay in the cache. Also, with Rabin fingerprinting, the compression ratio for 8KB is worse than that of 16KB and 32KB in all portions of the image. This is because, as mentioned earlier in Section 3.1, the 8KB Rabin fingerprinting scheme used a maximum chunk size of 8.5KB.

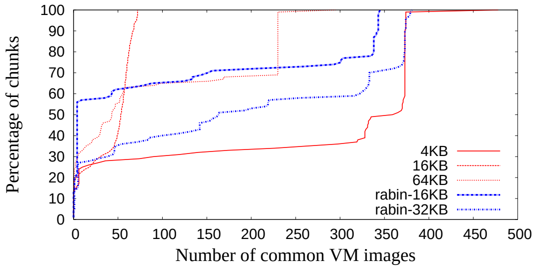
3.6 Pairwise Similarity between Chunks

To observe the affinity between image chunks that often appear together in the same image, we conduct a pairwise similarity study of all the distinct chunks appearing in the 525 images. The pairwise similarity between chunks A and B is defined to be $S(A,B) = \frac{|C(H(A),H(B))|}{|H(A)|}$, where $H(A)$ and $H(B)$ are the sets of VM images containing chunk A and B, and $C(H(A),H(B))$ is the intersection of $H(A)$ and $H(B)$, or in other words, the set of images containing both A and B. A similar definition is used in [12].

Figure 5(a) presents the CDF of chunk pairwise similarity. Due to the large number of chunks, we only present the results for "popular" chunks that appear at least 100, 500, or 1000 times depending on the chunking scheme. We can predict the pairwise similarity for "unpopular" chunks is low (almost zero) because they appear less frequently and most of them never appear in the same VM images (that is, $|C(H(A),H(B))| = 0$). Considering the pairwise similarity as defined above depends $|H(A)|$, its value is relatively higher when A only appears in a few images. Therefore, we use Figure 5(b) to show the absolute number of images that a pair of chunks appear together in. In both figures, we can observe a sudden increase in the CDF curve, indicating a small percentage of chunks with high similarity among each other.



(a) Pairwise similarity



(b) Number of common images

Figure 5: CDF of pairwise chunk similarity for the popular chunks

3.7 Pairwise Similarity between Images

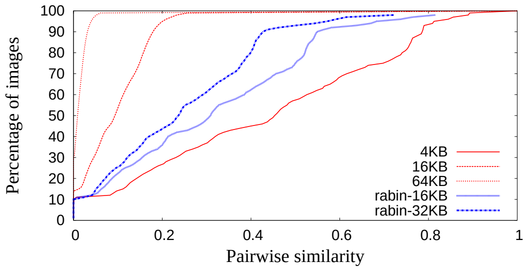


Figure 6: CDF of pairwise similarity among VM images

In this section, we examine the pairwise similarity between any two VM images, i.e., the number of common chunks appearing in both images. Similar to chunk pairwise similarity, we define the similarity between two VM images A and B to be $S(A,B) = \frac{|C(H(A),H(B))|}{|H(A)}|$, where $H(A)$ and $H(B)$ are the set of chunks contained in VM images A and B, respectively, and $C(H(A),H(B))$ is the set of common chunks in A and B.

Since each image contains a large number of chunks, calculating the pairwise similarity between each pair of images requires extremely large amount of computation. Therefore, we sample the whole VM image set and select 185 random images to conduct pairwise similarity study. Figure 6 shows the CDF of pairwise similarity using different chunking schemes. Figure 6 reflects trends observed earlier with Figure 1, i.e., a smaller chunk size leads to greater pairwise similarity. The maximum pairwise similarity is obtained by using 4KB fixed size chunking, but it is interesting to note that 16KB Rabin fingerprinting also performs reasonably well – approximately 10% of VM have pairwise similarity higher than 53%.

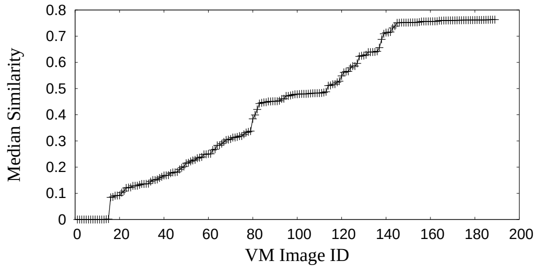


Figure 7: Median similarity for each VM image (in ascending order)

We further explore the distribution of VM image pairwise similarity. We find out that the similarity between VM images is not uniformly distributed. We use 4KB as an example to show how VM image pairwise similarity varies across VM images and the distribution pattern can be applied to chunks schemes except the similarity is lower. For each VM image, Figure 7 shows the median of its pairwise similarities with the other 184 images using 4KB fixed size chunking and all these VM images are sorted in the ascending order of its median similarity. It can be seen that there are around 40 images with median similarity level of above 70%. In an intelligent cache management mechanisms, those images will be given higher priority since their content is useful in reconstructing other images.

4. CONCLUSIONS

In this paper, we have conducted an empirical study of a large collection of VM images (525 in toal) from a production cloud using black-box similarity detection techniques. One of the key conclusions of our analysis is that fixed size chunking works very well. By using the right chunk size, it is possible to detect significant duplication among VM images, and to obtain very good compression ratios (up to 80%). For fixed size chunking, the level of duplication decreases as the chunk size increases. But from a storage systems standpoint, even 8KB and 16KB chunking schemes yield reasonably good compression ratios. For Rabin fingerprinting, the key to detecting duplication is to avoid using a maximum chunk size. We also observe that the duplication is maximum at the beginning of the image and decreases towards the end.

Overall, the data favor the design and implementation of smart VM image management middleware. The similarity among VM images, and the knowledge of image clusters can be leveraged to design smart image distribution schemes, which take the data center topology into account and help VM provisioning, cloning and migration. The presence of popular chunks common to many VM images also support the use of CDN-like image distribution schemes. The increasing similarity between VM images can also be leveraged to design hypervisor-level caches which avoid redundant data transfer during provisioning and execution.

5. ACKNOWLEDGEMENTS

The authors wish to thank Xiaolan Zhang, Todd Mummert, Darrell Reimer and Vasanth Bala for their suggestions and help with the Mirage VM image format and repository.

6. REFERENCES

[1] A. Swartz and L. Siri. Turnkey Linux Virtual Appliance Library. See `http://www.turnkeylinux.org/`.

[2] Amazon Web Services (AWS) Inc. Elastic Compute Cloud (EC2). See `http://aws.amazon.com`. VM image data retrieved from an author's AWS console on Aug 7, 2011.

[3] G. Ammons, V. Bala, T. Mummert, D. Reimer, and Z. Xiaolan. Virtual machine images as structured data: The mirage image library. In *HotCloud '11*, Portland, OR, June 2011.

[4] B. H. Bloom. Space/time trade-offs in hash coding with allowable errors. *Commun. ACM*, 13.

[5] A. T. Clements, I. Ahmad, M. Vilayannur, and J. Li. Decentralized deduplication in san cluster file systems. USENIX'09.

[6] D. Reimer and A. Thomas and G. Ammons and T. Mummert and B. Alpern and V. Bala. Opening Black Boxes: Using Semantic Information to Combat Virtual Machine Image Sprawl. In *VEE '08*.

[7] IBM. Tivoli Virtual Deployment Engine (VDE) Beta.

[8] K. Jin and E. L. Miller. The effectiveness of deduplication on virtual machine disk images. In *SYSTOR'09*.

[9] D. T. Meyer and W. J. Bolosky. A study of practical deduplication. FAST'11.

[10] D. Nurmi, R. Wolski, C. Grzegorczyk, G. Obertelli, S. Soman, L. Youseff, and D. Zagorodnov. The eucalyptus open-source cloud-computing system. CCGRID '09.

[11] H. Pucha, D. G. Andersen, and M. Kaminsky. Exploiting similarity for multi-source downloads using file handprints. NSDI'07.

[12] K. Tangwongsan, H. Pucha, D. G. Andersen, and M. Kaminsky. Efficient similarity estimation for systems exploiting data redundancy. INFOCOM'10.

[13] VMWare Inc. vSphere Data Center Virtualization. See `http://www.vmware.com/products/vsphere/`.