Multi-level Selective Deduplication for VM Snapshots in Cloud Storage

Wei Zhang*†, Hong Tang†, Hao Jiang†, Tao Yang*, Xiaogang Li†, Rachael Zeng†
*Dept. of Computer Science, UC Santa barbara. Email: {wei, tyang}@cs.ucsb.edu
†Alibaba Inc. Email: {hongtang, haojiang, xiaogang, rachael}@alibaba-inc.com

***Abstract*—In a virtualized cloud computing environment, frequent snapshot backup of virtual disks improves hosting reliability but storage demand of such operations is huge. While dirtybit-based technique can identify unmodified data between versions, full deduplication with fingerprint comparison can remove more redundant content at the cost of computing resources. This paper presents a multi-level selective deduplication scheme which integrates inner-VM and cross-VM duplicate elimination under a stringent resource requirement. This scheme uses popular common data to facilitate fingerprint comparison while reducing the cost and it strikes a balance between local and global deduplication to increase parallelism and improve reliability. Experimental results show the proposed scheme can achieve high deduplication ratio while using a small amount of cloud resources.**

I. Introduction

In a virtualized cloud environment such as ones provided by Amazon EC2 and Alibaba Aliyun, each instance of a guest operating system runs on a virtual machine, accessing virtual hard disks represented as virtual disk image files in the host operating system. Because these image files are stored as regular files from the external point of view, backing up VM's data is mainly done by taking snapshots of virtual disk images.

A snapshot preserves the data of a VM's file system at a specific point in time. VM snapshots can be backed up incrementally by comparing blocks from one version to another and only the blocks that have changed from the previous version of snapshot will be saved [3], [13].

Frequent backup of VM snapshots increases the reliability of VM's hosted in a cloud. For example, Aliyun, the largest cloud service provider by Alibaba in China, provides automatic frequent backup of VM images to strengthen the reliability of its service for all users. The cost of frequent backup of VM snapshots is high because of the huge storage demand. Using a backup service with full deduplication support [9], [14] can identify content duplicates among snapshots to remove redundant storage content, but the weakness is that it either adds the extra cost significantly or competes computing resource with the existing cloud services. In addition, data dependence created by duplicate relationship among snapshots adds the complexity in fault tolerance management, especially when VMs can migrate around in the cloud.

Unlike the previous work dealing with general file-level backup and deduplication, our problem is focused on virtual disk image backup. Although we treat each virtual disk as a file logically, its size is very large. On the other hand, we need to support parallel backup of a large number of virtual disks in a cloud everyday. One key requirement we face at Alibaba Aliyun is that VM snapshot backup should only use a minimal amount of system resources so that most of resources is kept for regular cloud system services or applications. Thus our objective is to exploit the characteristics of VM snapshot data and pursue a cost-effective deduplication solution. Another goal is to decentralize VM snapshot backup and localize deduplication as much as possible, which brings the benefits for increased parallelism and fault isolation.

By observations on the VM snapshot data from production cloud, we found snapshot data duplication can be easily classified into two categories: *inner-VM* and *cross-VM*. Inner-VM duplication exists between VM's snapshots, because the majority of data are unchanged during each backup period. On the other hand, Cross-VM duplication is mainly due to widely-used software and libraries such as Linux and MySQL. As the result, different VMs tend to backup large amount of highly similar data.

With these in mind, we have developed a decentralized multi-level solution to conduct segment-level and block-level inner-VM deduplication to localize the deduplication effort when possible. It then makes cross-VM deduplication by excluding a small number of popular common data blocks from being backed up. Our study shows that common data blocks occupy significant amount of storage space while they only take a small amount of resources to deduplicate. Seperating deduplication into multi levels effectively accomplish the major space saving goal compare the global complete deduplication scheme, at the same time it makes the backup of different VMs to be indepdent for better fault tolerance.

The rest of the paper is arranged as follows. Section II discusses on some background and related work. Section III discusses the requirements and design options. Section IV presents our snapshot backup architecture with multi-level selective deduplication Section V presents our evaluation results on the effectiveness of multi-level deduplication for snapshot backup. Section VI concludes this paper.

II. Background and Related Work

In a VM cloud, several operations are provided for creating and managing snapshots and snapshot trees, such as creating snapshots, reverting to any snapshot, and removing snapshots. For VM snapshot backup, file-level semantics are normally not provided. Snapshot operations are taken place at the virtual

device driver level, which means no fine-grained file system metadata can be used to determine the changed data. Only raw access information at disk block level are provided.

VM snapshots can be backed up incrementally by identifying file blocks that have changed from the previous version of the snapshot [3], [13], [12]. The main weakness is that it does not reveal content redundancy among data blocks from different snapshots or different VMs.

Data deduplication techniques can eliminate redundancy globally among different files from different users. Backup systems have been developed to use content hash (finger prints) to identify duplicate content [9], [11]. Today's commercial data backup systems (e.g. from EMC and NetApp) use a variable-size chunking algorithm to detect duplicates in file data [7], [4]. As data grows to be big, fingerprint lookup in such schemes becomes too slow to be scalable. Several techniques have been proposed to speedup searching of duplicate content. For example, Zhu et al. [14] tackle it by using an in-memory Bloom filter and prefetch groups of chunk IDs that are likely to be accessed together with high probability. It takes significant memory resource for filtering and caching. NG et al. [8] use a related filtering technique for integrating deduplication in Linux file system and the memory consumed is up to 2GB for a single machine. That is still too big in our context discussed below.

Duplicate search approximation [2], [6] has been proposed to package similar content in one location, and duplicate lookup only searches for chunks within files which have a similar file-level or segment-level content fingerprints. That leads to a smaller amount of memory usage for storing meta data in signature lookup with a tradeoff of the reduced recall ratio.

III. Requirements and Design Options

We discuss the characteristics and main requirements for VM snapshot backup in a cloud environment. which are different from a traditional data backup.

1) *Cost consciousness.* There are tens of thousands of VMs running on a large-scale cluster. The amount of data is so huge such that backup cost must be controlled carefully. On the other hand, the computing resources allocated for snapshot service is very limited because VM performance has higher priority. At Aliyun, it is required that while CPU and disk usage should be small or modest during backup time, the memory footprint of snapshot service should not exceed 500MB at each node.
2) *Fast backup speed*. Often a cloud has a few hours of light workload each day (e.g. midnight), which creates an small window for automatic backup. Thus it is desirable that backup for all nodes can be conducted in parallel and any centralized or cross-machine communication for deduplication should not become a bottleneck.
3) *Fault tolerance.* The addition of data deduplication should not decrease the degree of fault tolerance. It's not desirable that small scale of data failure affects the backup of many VMs.

There are multiple choices in designing a backup architecture for VM snapshots. We discuss the following design options with a consideration on their strengths and weakness.

1) *An external and dedicated backup storage system.* In this architecture setting, a separate backup storage system using the standard backup and deduplication techniques can be deployed [14], [2], [6]. This system is attached to the cloud network and every machine can periodically transfer snapshot data to the attached backup system. A key weakness of this approach is communication bottleneck between a large number of machines in a cloud to this centralized service. Another weakness is that the cost of allocating separate resource for dedicated backup can be expensive. Since most of backup data is not used eventually, CPU and memory resource in such a backup cluster may not be fully utilized.
2) *A decentralized and co-hosted backup system with full deduplication.* In this option, the backup system runs on an existing set of cluster machines. The disadvantage is that even such a backup service may only use a fraction of the existing disk storage, fingerprint-based search does require a significant amount of memory for fingerprint lookup of searching duplicates. This competes memory resource with the existing VMs.
   Even approximation [2], [6] can be used to reduce memory requirement, one key weakness the hasn't been addressed by previous solutions is that global content sharing affects fault isolation. Because a content chunk is compared with a content signature collected from other users, this artificially creates data dependency among different VM users. In large scale cloud, node faiures happen at daily basis, the loss of a shared block can affect many users whose snapshots share this data block. Without any control of such data sharing, we can only increase replication for global dataset to enhance the availability, but this incurs significantly more cost.

With these considerations in mind, we propose a decentralized backup architecture with multi-level and selective deduplication. This service is hosted in the existing set of machines and resource usage is controlled with a minimal impact to the existing applications. The deduplication process is first conducted among snapshots within each VM and then is conducted across VMs. Given the concern that searching duplicates across VMs is a global feature which can affect parallel performance and complicate failure management, we only eliminate the duplication of a small but popular data set while still maintaining a cost-effective deduplication ratio. For this purpose, we exploit the data characteristics of snapshots and collect most popular data. Data sharing across VMs is limited within this small data set such that adding replicas for it could enhance fault tolerance.

IV. Multi-level Selective Deduplication

A. Snapshot Service Architecture

Our architecture is built on the Aliyun platform which provides the largest public VM cloud in China based on

Xen [1]. A typical VM cluster in our cloud environment consists of from hundreds to thousands of physical machines, each of which can host tens of VMs.

A GFS-like distributed file system holds the responsibility of managing physical disk storage in the cloud. All data needed for VM services, which include runtime VM disk images and snapshot backup data, reside in this distributed file system. During the VM creation, a user chooses her flavor of OS distribution and the cloud system copies the corresponding pre-configured base VM image to her VM as the OS disk, and an empty data disk is created and mounted onto her VM as well. All these virtual disks are represented as virtual machine image files in our underline runtime VM storage system. The runtime I/O between virtual machine and its virtual disks is tunneled by the virtual device driver (called TapDisk at Xen). To avoid network latency and congestion, our distributed file system place the primary replica of VM's image files at physical machine of VM instance. During snapshot backup, concurrent disk write is logged to ensure a consistent snapshot version is captured.

Figure 1 shows the architecture view of our snapshot service at each node. The snapshot broker provides the functional interface for snapshot backup, access, and deletion. The inner-VM deduplication is conducted by the broker to access meta data in the snapshot data store and we discuss this in details in Section IV-B. The cross-VM deduplication is conducted by the broker to access a common data set (CDS) (will discuss in Section IV-C, whose block hash index is stored in a distributed memory cache.

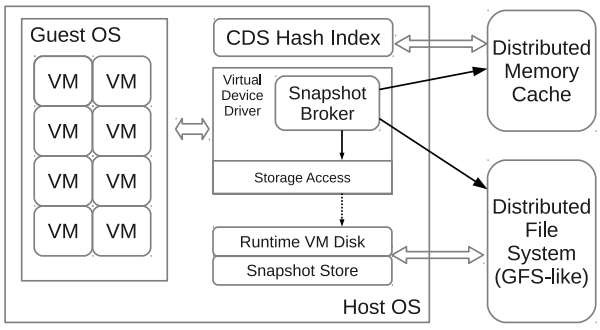

Fig. 1. Snapshot backup architecture of each node.

The snapshot store supports data access operations such as *get*, *put* and *delete*. Other operations include data block traverse and resource usage report. The snapshot data does not need to be co-located with VM instances, and in fact they can even live in a different cluster to improve the data reliability: when one cluster is not available, we are still able to restore its VMs from another cluster which holds its snapshot data.

Under the hood of snapshot store, it organizes and operates snapshot data in the distributed file system. We let each virtual disk has its own snapshot store, and no data is shared between any two snapshot stores, thus achieve great fault isolation. For those selected popular data that shared by many VM snapshot stores, we could easily increase its availability by having more replications.

B. Inner-VM Deduplication

The first-level deduplication is logically localized within each VM. Such localization increases data independency between different VM backups, simplifies snapshot management and statistics collection during VM migration and termination, and facilitates parallel execution of snapshot operations.

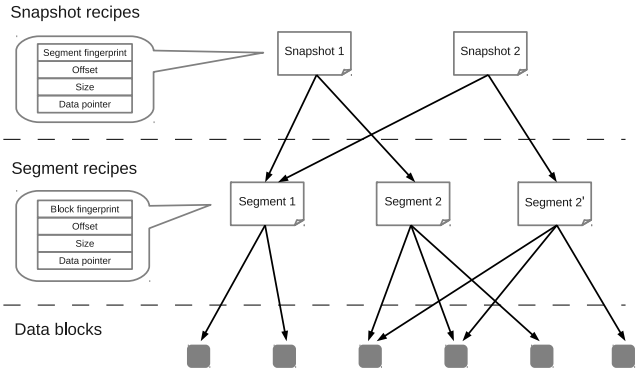

Fig. 2. An example of snapshot representation.

The inner VM deduplication contains two levels of duplicate detection efforts and the representation of each snapshot is correspondingly designed as a two-level level index data structure in the form of a hierarchical directed acyclic graph as shown in Figure 2. An image file is divided into a set of segments and each segment contains hundreds of content blocks from the bottom level. These blocks are of variable-size, partitioned using the standard chunking technique [7] with 4KB as the average block size. Segment metadata (called segment recipe) records its content block hashes and data pointers. The snapshot recipe contains a list of segments and other meta data information.

- *Level 1 Segment modification detection*. If a segment is not changed, indicated by a dirty bit embedded in the virtual disk driver, its content blocks are not changed as well, thus its segment recipe can be simply reused. Operations at this level have almost no cost and most of unmodified data are filtered here.
- *Level 2 Block fingerprint comparison.* If a segment is modified, we perform fine-grained deduplication using content blocks of this segment to compared with the same segment in the previous snapshot (also called parent snapshot). Partial duplication within the segment can be detected and eliminated.

We choose this two-level structure because in practice we observe that during each backup period only a small amount of VM data are added or modified. As the result, even

the meat data of two snapshots can be highly similar, thus aggregating a large number of content blocks as one segment can significantly reduce the space cost of snapshot meta data.

How can we choose the length of a segment? Instead of using variables-sized segments, we take a simple approach to let every segment being aligned to the page boundary of each virtual image file. For Aliyun, each VM image file is represented as a virtual disk file format (called *vhd* at Xen) and we use a dirty bit to capture if a page (or segment) of a virtual disk file has been modified or not to ease the segment-level deduplication. A dirty bit array is used to indicate which segments have been modified or not. Each page (segment) size in our implementation uses 2MB, which contains a large number of content blocks.

Once level-2 deduplication is activated for those segments that have been modified, it requires memory to load block fingerprints from the corresponding parent snapshot's segment. This scheme processes one segment at time and each segment of 2MB contains about 500 content blocks on average given 4KB is the average block size. That only takes a tiny amount of space to hold their fingerprints.

C. Cross-VM Deduplication with CDS

The level-3 deduplication is to identify duplicated data blocks among multiple VMs through the index cache of common data set (CDS). CDS is the most popular content blocks among snapshots across all VMs. Each index entry contains the block fingerprint and a reference pointer to the location of its real content in the snapshot content block store.

At the runtime, the CDS index resides in a distributed memory lookup table implemented using Memcached [5] to leverage the aggregated memory in a cluster. Th usage of memory at each machine is small and thus this scheme does not lead to a memory resource contention with the existing cloud services. CDS raw data stored in the distributed file system has multiple copies in different machines for the purpose of fault tolerance and while providing high read throughput.

To control the size of searchable common data in this global setting, we focus on those items that are most popular based on the historical snapshot data and the popular analysis is conducted periodically to ensure meta data freshness to match the latest data access trend. There are two advantages to exploit this. First, a smaller CDS reduces overall resource requirement while covering most of data duplication. Second, knowing this small set of data is shared heavily makes the fault tolerance management easier because we can replicate more copies to mitigate the failure.

In Aliyun's VM cloud, each VM explicitly maintains one OS disk, plus one or more data disks. During VM's creation, its OS disk is directly copied from user's chosen base image. Given the separation of OS disks and data disks, we study their characteristics separately. We expect that data related to OS and popular software installations are not frequently modified or deleted, to facilitate analysis we call them *OS-related* data, and the rest of data, either from OS or data disks, are called *user-related* data. We have studied the popularity of common blocks in the OS and data disks from a dataset containing over 1,000 VMs, taking their first snapshots to watch the cross-VM duplication pattern (scale of OS disk sampling is smaller due to performance impact to user VMs). Figure 3 shows the duplicate count for unique data blocks sorted by their ranking in terms of the duplication count. Y axis is the popularity of a data block in a log scale measured its duplicate count among snapshots. X axis is the identification of data blocks in a log scale sorted by their duplicate rank. The rank number 1 is the block with the highest number of duplicates. These two curves exhibit that the popularity of common blocks partially follows a Zipf-like distribution.

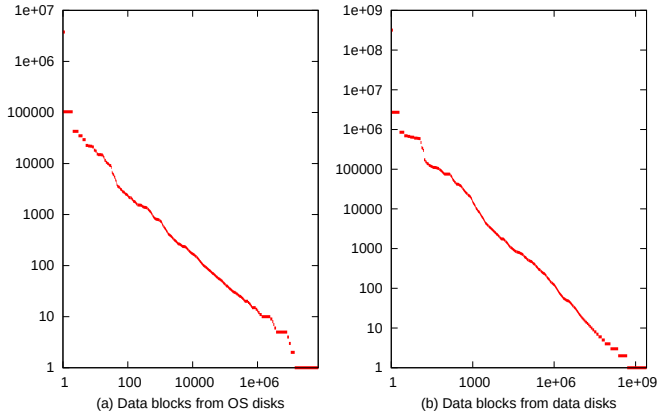

Fig. 3. Number of duplicates vesus ranking

Base on the Zipf-like distribution pattern, many previous analysis and results on web caching may apply. In general this indicates a small set of popular data dominates data duplication. Although we already know OS-related data are heavily duplicated among OS disks, it still surprises us that user-related data follow a similar pattern. Therefore, we collect the CDS by performing global reference counting through map-reduce for all blocks in snapshot store, and select the most popular ones base on the memory limitation of CDS hash index.

The CDS collected from user-related data is generally proportional to the data size. As discussed in Section V, selecting about 1% of data (after level 1 and level 2 deduplication) can cover about 38% of data blocks. Consider we allow maximum 25 VMs per machine, each VM has about 30GB of user-related data, having 10 snapshots in its snapshot store and the data change ratio during each backup is 10%, this translates to 15GB CDS data per machine. Consider each CDS hash index entry cost about 40 bytes and the average block size is 4KB, this leads to a 1:100 ratio so the memory cost by CDS hash index is about 150MB per machine. On the other hand, the CDS from OS-related data is only relvant to the number of OS releases. Our experiment on 7 major OS releases shows

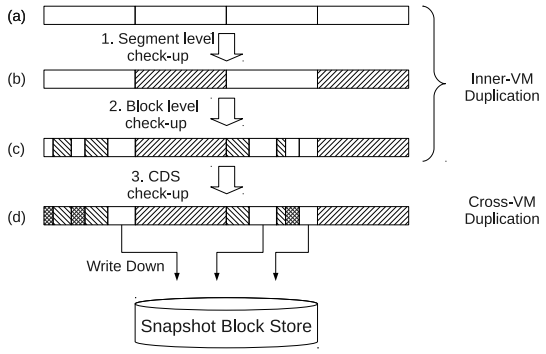

Fig. 4. Illustration of snapshot deduplication dataflow.

that about 100GB of data is never modified, and we expect it won't grow over 200GB in the near future. So it would cost the entire cluster 2GB of memory to store its hash index, or 20MB per machine on a 100 node cluster. On average each OS disk has about 10GB of data that can be completely eliminated in this way. Thus in total the CDS index size per node takes less than 170MB in a large cluster bigger than 100 nodes, covering over 47% of blocks in OS and data disks after inner-VM deduplication. This memory usage is well below the limit required by our VM services.

D. Illustration of multi-level deduplication process

We illustrate the steps of 3-level deduplication in Figure 4, which can be summarized as 5 steps:

1) *Segment level checkup.* As shown in Figure 4 (a), when a snapshot of a plain virtual disk file is being created, we first check the dirty bitmap to see which segments are modified. If a segment is not modified since last snapshot, it's data pointer in the recipe of the parent snapshot can be directly copied into current snapshot recipe (shown as the shadow area in Figure 4 (b)).
2) *Block level checkup.* As shown in Figure 4 (b), for each dirty segment, we divide it into variable-sized blocks, and compare their signatures with the corresponding segment recipe in the previous snapshot (called parent snapshot). For any duplicated block, we copy the data pointer directly from the parent segment recipe.
3) *CDS checkup.* For the remaining content blocks whose duplicate status is unknown, Figure 4 (d) shows a further check to compare them with the cached signatures in the CDS by querying the CDS hash index. If there is a match, the corresponding data pointer from the CDS index is copied into the segment recipe.
4) *Write new snapshot blocks :* If a data block cannot be found in the CDS index, this block is considered to be a new block and such a block is to be saved in the snapshot store, the returned data pointer is saved in the segment recipe.
5) *Save recipes.* Finally the segment recipes are saved in the snapshot block store also. After all segment recipes are saved, the snapshot recipe is complete and can be saved.

If there is no parent snapshot available, which happens when a VM creates its first snapshot, only CDS-based checkup will be conducted. Most of the cross-VM duplication, such as OS-related data, is eliminated at this stage.

V. Evaluation

We have implemented the snapshot deduplication scheme on the Aliyun's cloud platform. Objectives of our experimental evaluation are: 1) Analyze the commonality of content data blocks and the popularity of hot items. 2) Assess the effectiveness of 3-level deduplication for reducing the storage cost of snapshot backup. 3) Examine the impacts of CDS size on deduplication ratio.

A. Experimental setup

At Aliyun our target is to backup cluster up to 1000 nodes with 25 VMs on each. Based on the data studied, each VM has about 40GB of storage data usage on average, OS disk and data disk each takes about 50% of storage space. The backup of VM snapshots is completed within two hours every day, and that translates to a backup throughput of 139GB per second, or 500TB per hour. For each VM, the system keeps 10 automatically-backed snapshots in the storage while a user may instruct extra snapshots to be saved.

Since it's impossible to perform large scale analysis without affecting the VM performance, we sampled two data sets from real user VMs to measure the effectiveness of our deduplication scheme. Dataset1 is used study the detail impact of 3-level deduplication process, it compose of 35 VMs from 7 popular OSes: Debian, Ubuntu, Redhat, CentOS, Win2003 32bit, win2003 64 bit and win2008 64 bit. For each OS, 5 VMs are chosen, and every VM come with 10 full snapshots of it OS and data disk. The overall data size for this 700 full snapshots is 17.6 TB.

Dataset2 contains the first snapshots of 1323 VMs' data disks from a small cluster with 100 nodes. Since inner-VM deduplication is not involved in the first snapshot, this data set helps us to study the CDS deduplication against user-related data. The overall size of dataset2 is 23.5 TB.

All data are divided into 2 MB fix-sized segments and each segment is divided into variable-sized content blocks [7], [10] with an average size of 4KB. The signature for variable-sized blocks is computed using their SHA-1 hash. Popularity of data blocks are collected through global counting and the top 1% will fall into CDS, as discussed in Section IV-C.

B. Effectiveness of 3-level Deduplication

Figure 5 shows the overall impact of 3-level deduplication on dataset1. The X axis shows the overall impact in (a), impact on OS disks in (b), and impact on data disks in (c). Each bar in the Y axis shows the data size after deduplication divided by the original data size. Level-1 elimination can reduce the

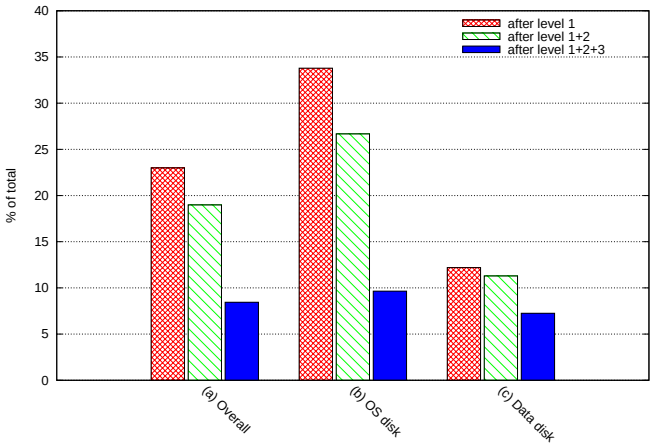

Fig. 5. Impacts of 3-level deduplication. The height of each bar is the data size after deduplication divided by the original data size and the unit is percentage.

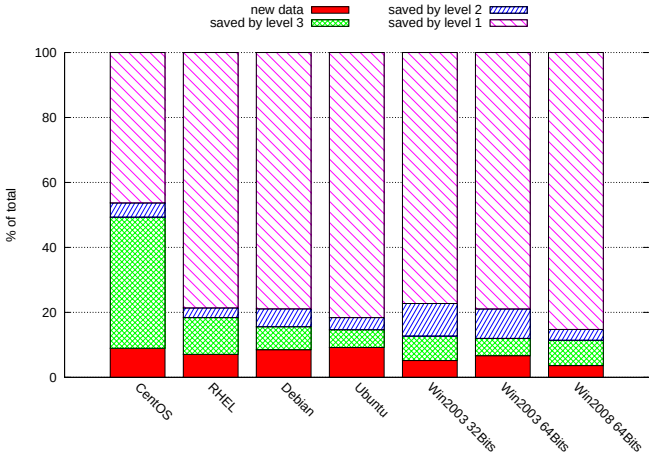

Fig. 6. Impact of 3-level deduplication for OS releases.

data size to about 22% of original data, namely it delivers close 78% reduction. Level-2 elimination is applied to data that could pass level-1, it reduces the size further to about 18.5% of original size, namely it delivers additional 4.5% reduction. Level-3 elimination together with level 1 and 2 reduces the size further to 8% of original size, namely it delivers additional 10.5% reduction. Level 2 elimination is more visible in OS disk than data disk, because data change frequency is really small when we sample last 10 snapshots of each user in 10 days. Nevertheless, the overall impact of level 2 is still significant. A 4.5% of reduction from the original data represents about 450TB space saving for a 1000-node cluster.

Figure 6 shows the impact of different levels of deduplication for different OS releases. In this experiment, we tag each block in 350 OS disk snapshots from dataset1 as "new" if this block cannot be deduplicated by our scheme and thus has to be written to the snapshot store; "CDS" if this block can be found in CDS; "Parent segment" if this block is marked unchanged in parent's segment recipe. "Parent block" if this block is

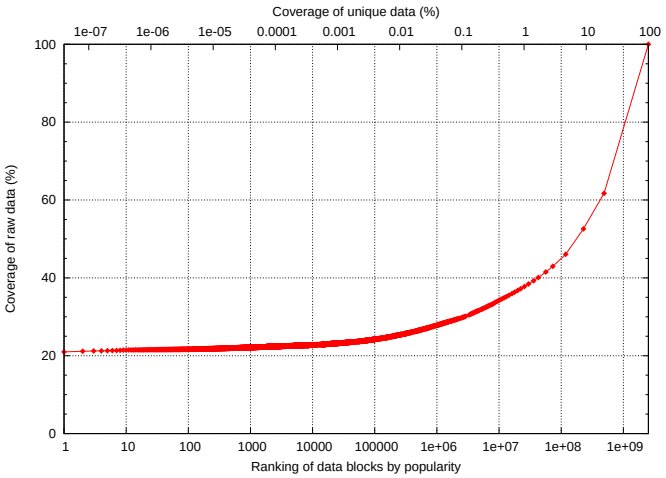

Fig. 7. Cumulative coverage of popular common user data blocks.

marked unchanged in parent's block recipe. With this tagging, we compute the percentage of deduplication accomplished by each level. As we can see from Figure 6, level-1 deduplication accomplishes a large percentage of elimination, this is because the time interval between two snapshots in our dataset is quite short and the Aliyun cloud service makes a snapshot everyday for each VM. On the other hand, CDS still finds lots of duplicates that inner VM deduplication can't find, contributing about about 10% of reduction on average.

It is noticable that level-1 deduplication doesn't work well for CentOS, a significant percentage of data is not eliminated until they reach level-3. It shows that even user upgrade his VM system heavily and frequently such that data locality is totally lost, those OS-related data can still be identified at level-3.

In gengral we see a stable data reduction ratio for all OS varieties, ranging from 92% to 97%, that means the storage cost of 10 full snapshots combined is still smaller than the original disk size. And compare to today's widely used copy-on-write snapshot technique, which is similar to our level-1 deduplication, our solution cut the snapshot storage cost by 64%.

C. Coverage of common data blocks

One of our biggest advantage is small memory footprint for deduplication, because we only eliminate a small amount of highly popular data. we use dataset2 to demostrate the coverage of popular data blocks on user-related data, as shown in Figure 7. The X axis is the rank of cmomon user data blocks, and Y shows how much raw data can be covered given the size of CDS. Let S_i and F_I be the size and duplicate count of the i-th block ranked by its duplicate rank. Then Y axis is the coverage of the common dataset covering data items from rank 1 to rank i. Namely

$$\frac{\sum_{i=1}^{i} S_i * F_i}{\text{Total data size}}.$$

Thus with about 1% of blocks on data disks, CDS can

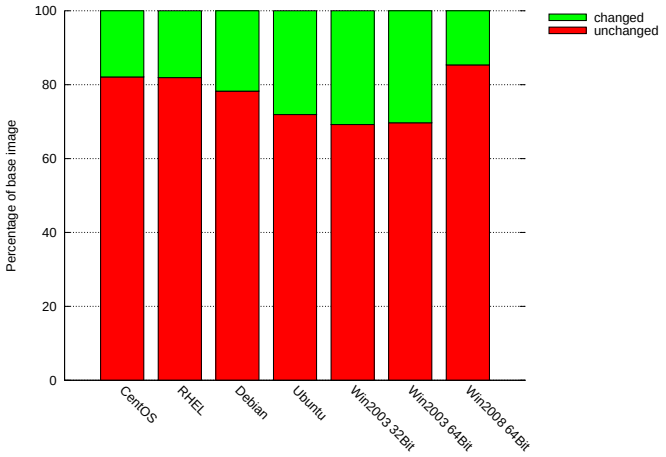

Fig. 8. Percentage of completely common blocks among different VMs for the same OS release.

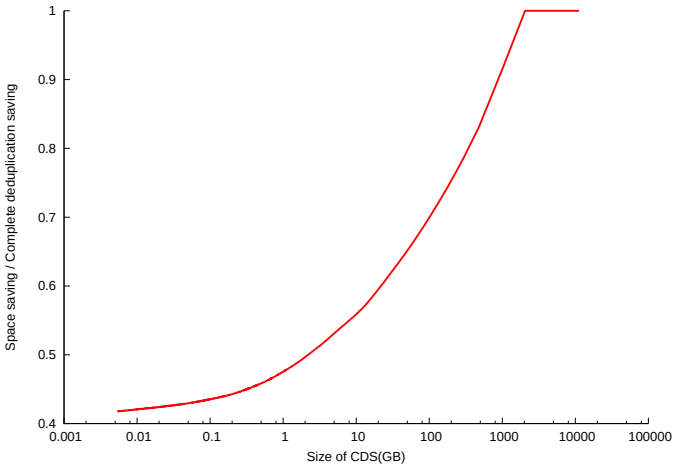

Fig. 9. Relative ratio of space size compared to full deduplication when CDS size changes.

cover about 38% of total data blocks appeared in in all data snapshots. The corresponding CDS index uses no more than 150MB memory per machine, which can be easily co-located with other cloud services.

Because there are a limited number of OS releases, we study common blocks among OS disks loaded with the same OS release. In Figure 8, we list a conservative commonality study in 7 major OS versions supported in Aliyun. For every block in the base image, we classify this block as "unchanged" if this block has appeared in all snapshots of the same OS release even they are used by different VMs. Figure 8 shows that for each OS, at least 70% of OS blocks are completely unchanged among all snapshots of the same OS. Some latest release of OS tends to have a higher percentage of content change while old release tends to have more variations of content versions. That can be interpreted as that users with very old version of operating systems have a lower tendency to update their OS versions and this causes a larger discrepancy among OS snapshots of these users.

Based on the above analysis, we have selected a small set of most popular OS blocks, which is about 100GB OS data and its corresponding CDS index takes about 1GB memory space in total. They can cover sufficiently over 70% of OS-related data.

D. A comparison of perfect and CDS-based deduplication

After level-1 and level 2 elimination, we find that the complete deduplication would reduce the storage cost further by 50%. If we put all these unique data into CDS, we could achieve complete deduplication, but fingerprint lookup in such huge CDS hash index would become a bottleneck as discussed in many pervious works. So we use dataset2 to evaluate how much space saving of deduplication can be achieved when varying the CDS size.

Figure 9 shows the relationship between CDS cache size and relative space saving ratio compared to the full deduplication. The unit of CDS size is gigabytes. We define *space saving ratio* as the space saving of CDS method divided by full deduplication space reduction. With a 100GB CDS data (namely 1GB CDS index) can still accomplish about 75% of what perfect deduplication can do.

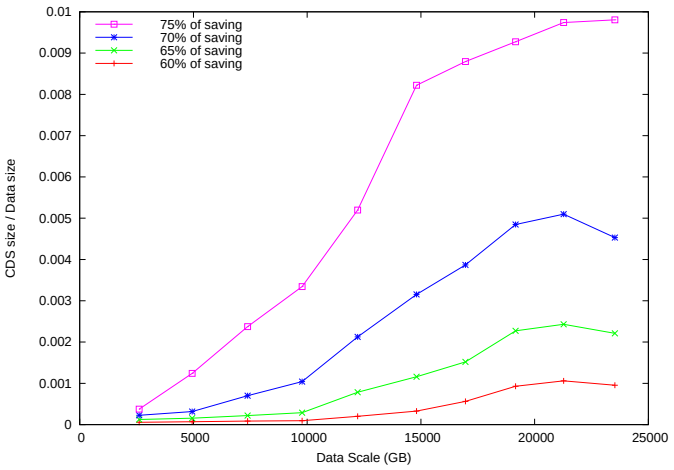

Fig. 10. The relative size ratio of CDS over the data size.

With dataset2, Figure 10 shows how CDS space saving ratio compared to the full deduplication is affected by the dataset size. In this experiment we first set out a goal of space saving ratio completed, then watch how much data needs to be placed in CDS cache to achieve this goal. From the graph we can see a 75% saving ratio lead to a stable ratio between CDS size and data size, which requires 1% of data to be placed in CDS.

When we deal with a large cluster of 1,000 nodes, we expect that using 1% of data disks can cover more than what we have seen from this 1323 VM dataset, assuming that the average behavior of every subcluster with 100 machines exhibits a similar commonality. In addition to this, CDS for OS disks will become even more effective when there are more VMs sharing the same collection of OS releases.

VI. Conclusions

In this paper we propose a multi-level selective deduplication scheme for snapshot service in VM cloud. Inner-VM deduplication localizes backup data dependency and exposes more parallelism while cross-VM deduplication with a small common data set effectively covers a large amount of duplicated data. Our solution accomplishes the majority of potential global deduplication saving while still meets stringent cloud resources requirement. Evaluation using real user's VM data shows our solution can accomplish 75% of what complete global deduplication can do. Compare to today's widely-used snapshot technique, our scheme reduces almost two-third of snapshot storage cost. Our scheme uses a very small amount of memory on each node, and leaves room for additional optimization we are further studying.

References

[1] P. Barham, B. Dragovic, K. Fraser, S. Hand, T. Harris, A. Ho, R. Neugebauer, I. Pratt, and A. Warfield. Xen and the art of virtualization. *SIGOPS Oper. Syst. Rev.*, 37:164–177, Oct. 2003.

[2] D. Bhagwat, K. Eshghi, D. Long, and M. Lillibridge. Extreme binning: Scalable, parallel deduplication for chunk-based file backup. In *Modeling, Analysis Simulation of Computer and Telecommunication Systems, 2009. MASCOTS '09. IEEE International Symposium on*, pages 1 –9, 21-23 2009.

[3] A. T. Clements, I. Ahmad, M. Vilayannur, and J. Li. Decentralized deduplication in san cluster file systems. In *Proceedings of the 2009 conference on USENIX Annual technical conference*, USENIX'09, pages 8–8, Berkeley, CA, USA, 2009. USENIX Association.

[4] C. Dubnicki, L. Gryz, L. Heldt, M. Kaczmarczyk, W. Kilian, P. Strzelczak, J. Szczepkowski, C. Ungureanu, and M. Welnicki. HYDRAstor: a Scalable Secondary Storage. In *FAST '09: Proccedings of the 7th conference on File and storage technologies*, pages 197–210, Berkeley, CA, USA, 2009. USENIX Association.

[5] B. Fitzpatrick. Distributed caching with memcached. *Linux J.*, 2004:5–, Aug. 2004.

[6] M. Lillibridge, K. Eshghi, D. Bhagwat, V. Deolalikar, G. Trezis, and P. Camble. Sparse indexing: Large scale, inline deduplication using sampling and locality. In *FAST*, pages 111–123, 2009.

[7] U. Manber. Finding similar files in a large file system. In *Proceedings of the USENIX Winter 1994 Technical Conference*, pages 1–10, 1994.

[8] C.-H. Ng, M. Ma, T.-Y. Wong, P. P. C. Lee, and J. C. S. Lui. Live deduplication storage of virtual machine images in an open-source cloud. In *Middleware*, pages 81–100, 2011.

[9] S. Quinlan and S. Dorward. Venti: A new approach to archival storage. In *FAST '02: Proceedings of the Conference on File and Storage Technologies*, pages 89–101, Berkeley, CA, USA, 2002. USENIX Association.

[10] M. O. Rabin. Fingerprinting by random polynomials. Technical Report TR-CSE-03-01, Center for Research in Computing Technology, Harvard University, 1981. http://www.xmailserver.org/rabin.pdf.

[11] S. Rhea, R. Cox, and A. Pesterev. Fast, inexpensive content-addressed storage in foundation. In *USENIX 2008 Annual Technical Conference on Annual Technical Conference*, pages 143–156, Berkeley, CA, USA, 2008. USENIX Association.

[12] Y. Tan, H. Jiang, D. Feng, L. Tian, and Z. Yan. Cabdedupe: A causality-based deduplication performance booster for cloud backup services. In *IPDPS*, pages 1266–1277, 2011.

[13] M. Vrable, S. Savage, and G. M. Voelker. Cumulus: Filesystem backup to the cloud. *Trans. Storage*, 5:14:1–14:28, Dec. 2009.

[14] B. Zhu, K. Li, and H. Patterson. Avoiding the disk bottleneck in the data domain deduplication file system. In *FAST'08: Proceedings of the 6th USENIX Conference on File and Storage Technologies*, pages 1–14, Berkeley, CA, USA, 2008. USENIX Association.