Multi-level Selective Deduplication for VM Snapshots in Cloud Storage

Wei Zhang[⋆†], Hong Tang[†], Hao Jiang[†], Tao Yang[⋆], Xiaogang Li[†], and Rachael Zeng[†]
[⋆]UC Santa Barbara
[†]Alibaba.com

Abstract

In a virtualized cloud computing environment, frequent snapshot backup of virtual disks improves hosting reliability while storage demand of such operations is huge. A dirtybit-based technique can identify unmodified data between versions and full deduplication with fingerprint comparison can remove more redundant content while it requires more computing resource. 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.

1 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 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 provided 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, 2, 6] 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. 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 and even each virtual disk is viewed 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 resource so that most of resource is allocated 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.

With these in mind, we have developed a decentralized multi-level solution to conduct segment-level and block-level within each VM to localize deduplication when possible. It then makes global inter-VM deduplication efforts by using a small number of popular common data blocks. Our study shows that there are a larger number of popular common content blocks shared by many users while they only take a small amount of resource for global deduplication. Leveraging popular blocks makes the fault tolerance management easier while it can still effectively accomplish a significant percentage of inter-VM deduplication compared to a full deduplication algorithm.

The rest of the paper is arranged as follows. Section 2 discusses on some background and related work. Section 3 discusses the requirements and design options. Section 4 presents our snapshot backup architecture with multi-level selective deduplication Section 5 presents our evaluation results on the effectiveness of multi-level deduplication for snapshot backup. Section 6 concludes this paper.

2 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 re-

moving 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 files 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, content chunk lookup for fingerprint comparison from disk storage becomes too slow to be scalable. Several techniques have been proposed to speedup searching of duplicate content [14, 2, 6]. 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. 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 disk, 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.

3 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 cloud cluster. The amount of VM data to backup is huge and cost of complete backup can be prohibitively high without deduplication and compression support. Thus cost minimization is required to deliver a deplorable solution.

   On the other hand, the computing resource allocated for backup is often very small. This is because snapshot data is not retrieved frequently by users and the allocation of cluster resource is typically prioritized towards active services such as online web services and map-reduce jobs. Thus it is not easy to accept any backup behavior that affects VMs performance or stability of active users. At Aliyun, it is required that while CPU and disk usage should be small or modest during backup time, the memory footprint of deduplication should not exceed 1% of memory at each node, which is about 480MB.
2. *Fast backup speed.* The snapshot backup operations need to be completed quickly every day (e.g. 1-2 hours). Often a cloud service has few hours of light workload each day (e.g. midnight), which creates an opportunity for backup. But a longer use of bandwidth and computing resource for backup can create noticeable contention with the existing cloud, which is not preferred for cloud production system operation. 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 a backup service should not decrease the degree of fault tolerance. If data failure in a backup service affects many users, such a solution is not desirable.

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 fast lookup of identical content chunks. This competes memory resource with the existing applications.

   Even approximation [2, 6] can be used to reduce

memory requirement, a key weakness of global content fingerprint comparison is that it can affect fault isolation. Because a content chunk is compared with a content signature collected from other users, this artificially creates data dependence among different cloud users. The loss of a common chunk can affect many users whose VM images share this chunk. Adding replicas for this global dataset can enhance the availability, but 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 duplication process is first conducted among data within each VM and then is conducted across VMs to utilize a common dataset. Given the concern that searching duplicates across VMs is a global feature which can affect parallel performance and complicate failure management, we would like to use a small common dataset while still maintaining a cost-effective deduplication ratio. For this purpose, we exploit the data characteristics of snapshots and only store most popular data.

4 Multi-level Selective Deduplication

4.1 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 few varieties of mainstream OS are supported. 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, Aliyun's distributed file system carefully selects the location of primary replica of VM's image files such that they are located on the same physical machine of VM instance. During snapshot backup, concurrent disk write is logged so that a consistent snapshot version can be derived.

Figure 1 shows the architecture view of snapshot service with selective deduplication at each node. The snapshot broker provides the functional interface for snapshot backup, access, and deletion. In accomplishing such a function, this broker accesses the snapshot store managed by the underlying distributed file system. 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 4.2. The inter-VM deduplication is conducted by the broker to access a common data block set (CDS) across multiple VMs. The CDS hash index is stored in the distributed memory cache.

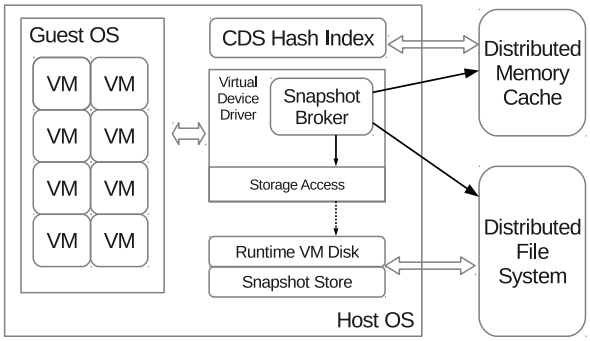

Figure 1: Snapshot backup architecture of each node.

The snapshot store residing in the distributed file system supports data access operations such as *get*, *put* and *delete*. Other operations include data block traverse and resource usage report. Snapshots are stored in our distributed file system, just like regular virtual data files. A key difference is that snapshot storage does not need to be co-located within VM instances and in fact they can even live in a different cluster to improve the data reliability. When a computing cluster is not available, we are still able to restore users' VMs from another cluster which contains snapshot backup and run these VMs.

With Aliyun's distributed file system support, the snapshot store operates *append-only* files. Each virtual disk has its own data store, all data blocks after deduplication will be stored into that system. Since all the underline data structures is append only, upon a delete request, the corresponding data will only be marked rather than being deleted. A compaction will take place when deleted data has accumulated to a certain threshold, thus reclaiming the disk space .

4.2 Inner-VM Deduplication

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

The inner VM deduplication contains two levels of duplicate detection efforts and the representation of each

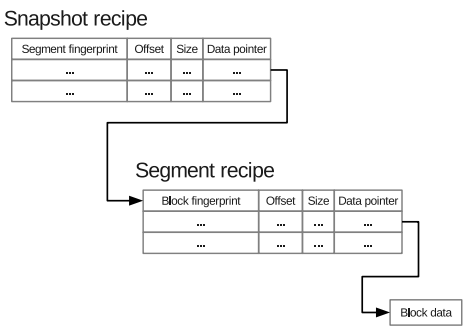

Figure 2: An example of snapshot representation.

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]. We use 4KB as the average block size. The segment metadata (called segment recipe) records its content block hashes and data reference. The snapshot-level recipe contains a set of segment recipes and other meta data information.

- *Level 1 Segment modification detection*. If a segment is not changed, indicated by a dirty bit, its content blocks are not changed also. In this case, segment-level duplicate detection is more efficient than the block level.
- *Level 2 Block fingerprint comparison.* If a segment is modified, content blocks of this segment are compared with the fingerprints of the existing content blocks within the same segment in the previous version of this snapshot (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 at each snapshot there are a large number of content blocks which are not modified from one snapshot version to another. 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 snapshot file has been modified or not for 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.

4.3 Cross-VM Deduplication with CDS

The third-level of 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 recent 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 residing in the underlying distributed file system.

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 blocks in snapshots. Second a smaller CDS 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 operating system blocks along with popular software installations are not frequently modified or deleted. We have sampled the popularity of common blocks in the OS and data disks from a dataset containing over 1,000 VMs and each has 10 snapshots.

The left part of Figure 3 shows the duplicate count for different 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. The right part of Figure 3 shows the duplicate count for OS disk snapshot blocks sorted by their ranking. These two curves exhibit that the popularity of common blocks partially follows a Zipf-like distribution.

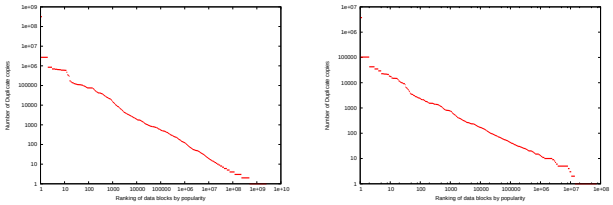

Figure 3: Duplicate count of common data blocks in a log scale on the left. The right is for common OS blocks.

As the number of VMs increases, the popularity of common blocks for data disks changes. But this increase has a small impact on OS block popularity because there are a limited number of OS releases used by VMs. We let CDS cover the common OS blocks first and our CDS selection strategy is designed as follows.

- First, most popular items from OS disks are selected. Our experiment data shows that about 100GB data from 7 most popular OS releases can at least cover over 70% of OS snapshots and the index of these 100GB OS blocks only takes about total 1GB in the CDS index and then memory requirement per node for duplicate search a large cluster is tiny. We expect that the OS CDS index may grow up to 2GB in order to cover large varieties of commonly deployed OS releases in practice.
- Second, the degree of duplication is computed for snapshot blocks of data disks and the most popular items are selected. As discussed in Section 5, that about 1% of data blocks are selected based on their duplicate counts and these popular blocks can cover about 38% of data blocks appeared in various snapshots after level 1 and level deduplication. The memory size per node used for CDS is within 150MB.

Thus in total the CDS index size per node takes less than 160MB in a large cluster bigger than 100 nodes based on Aliyun's experiment data, covering over 53% of blocks in OS and data disks after inner VM deduplication. This memory usage is well below the limit required by Aliyun.

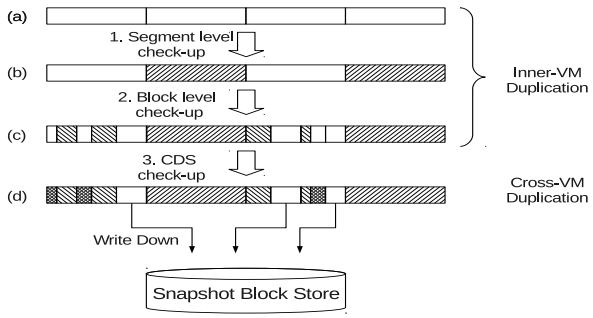

Figure 4: Illustration of snapshot deduplication dataflow.

4.4 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 array of this disk file 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 version (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 :* We continue the previous step and 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 block store and the returned reference 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 also.

If there is no parent snapshot available, which happens when a VM creates its first snapshot, only CDS-based

checkup will be conducted.

5 Evaluation

We have implemented a snapshot deduplication scheme on the Aliyun's cloud platform. Our experimental evaluation has the following two objectives: 1) Analyze the commonality of content data blocks and the popularity of hot items. We examine the impacts of CDS size on deduplication ratio. 2) Assess the effectiveness of 3-level deduplication for reducing the storage requirement of snapshot backup.

5.1 Experimental setup

An Aliyun cluster we target is over 1000 nodes with 16 cores and 48GB memory and each node hosts 25 VMs. We are running our deduplication/backup service on 100 nodes. Memory usage is about 150MB space per node during backup and the CPU usage is very small during the experiments. Based on the data studied, each VM has about 40GB of storage data usage on average and each of OS disk and data disk 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.

We have used a storage dataset of 1323 VM users collected from 100 Aliyun's cloud nodes to evaluate the effectiveness. In this dataset, there are 10 snapshots per each VM user and the total amount of space investigated is 23.5 terabytes. Each VM file is divided into content blocks of variable sizes [7, 10] with an average size of 4KB. The signature for variable-sized blocks is computed using their SHA-1 hash. Each segment is of size 2MB. Popularity is computed by using 90% of dataset, which reflects our setting that the system recomputes CDS every 1-2 days to catch up the popularity trend.

5.2 Effectiveness of 3-level Deduplication

Figure 5 shows the overall impact of 3-level deduplication. The X axis shows the overall impact in (a), impact on OS disk in (b), and impact on data disk 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 data size to about 22% of original data, namely it delivers close 78% reduction. Level-2 elimination applied after level 1 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

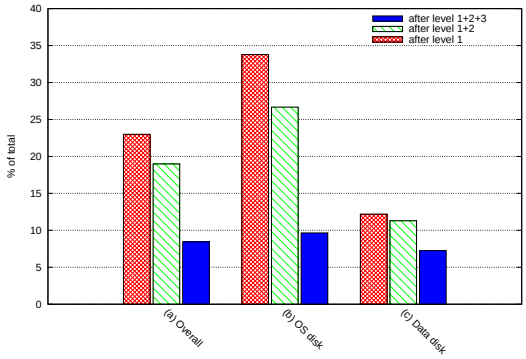

Figure 5: Impacts of 3-level deduplication. The height of each bar is the data size after deduplication divided by the original data size and the unit is percentage.

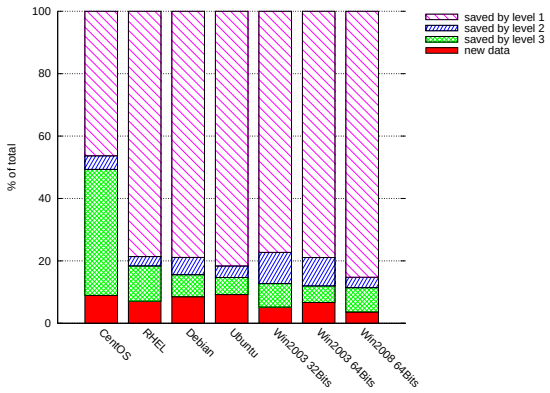

Figure 6: Impact of 3-level deduplication for OS releases.

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.

To see the impact of multi-level deduplication on different OS releases, we choose 7 major OSes in Aliyun's VM cloud platform and they are: Win2008 Server 64 bits, Win2003 Server 32 bits, Win2003 Server 64 bits, RHEL, CentOS, Ubuntu Server and Debian (all Linux distributions are 64 bits). Then we examine 5 VM user disks from each OS, and 10 snapshots for each VM. These VMs have been actively used by their owners. Figure 6 shows the impact of different levels of deduplication for different OS releases. In this experiment, we tag each block as "new" if this block cannot be deduplicated by our scheme and thus has to be written to the snapshot block 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 marked unchanged in parent's block recipe. With this tagging, we compute the percentage of dupli-

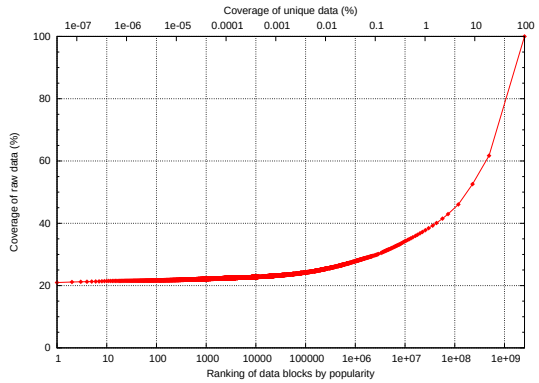

Figure 7: Cumulative coverage of popular common user data blocks.

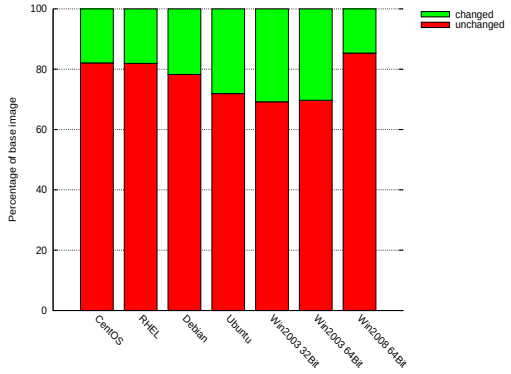

Figure 8: Percentage of completely common blocks among different VMs for the same OS release.

cation accomplished by level-1 inner VM deduplication, level-2 inner VM deduplication and level 3 cross-VM deduplication. 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.

5.3 Coverage of common data blocks

We further study how CDS covers content blocks used in image files using 90% of the collected dataset. Figure 7 shows the cumulative coverage ratio of popular common user data blocks. The X axis is the rank of common user data blocks. 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 cover about 38% of total data blocks appeared in in all data snapshots. The corresponding CDS index uses about 150MB memory per machine.

Because there are a limited number of OS releases supported by Aliyun, 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. We consider the earlier snapshot for each OS disk as the base image. For every block in the base image, we classify this block as "completely common" 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 common 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 blocks.

5.4 A comparison of perfect and CDS-based deduplication

After level-1 and level 2 elimination, our experiment shows that the full deduplication would reduce the storage further by 45% to 53% of user disk space. If we put all these unique data into CDS, we could achieve full deduplication, but the CDS size is not affordable. So we 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 the CDS method divided by full deduplication saving ratio. With a 100GB CDS data (namely 1GB CDS index) can still accomplish about 75% of what perfect deduplication can do.

Figure 10 shows how CDS space saving ratio compared to the full deduplication is affected by the dataset size. The calculation is done after level 1 and level 2 deduplication. X axis uses the data size collected from up to 1323 VMs, which occupies about 53 machines. 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

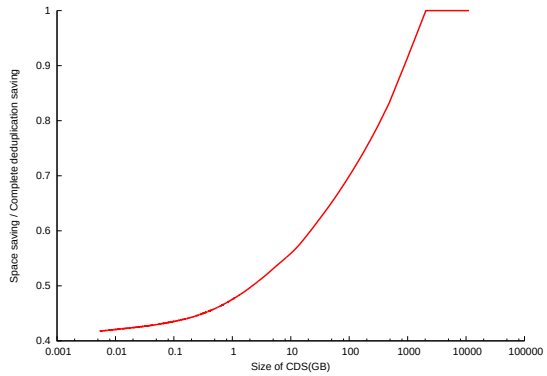

Figure 9: Relative ratio of space size compared to full deduplication when CDS size changes.

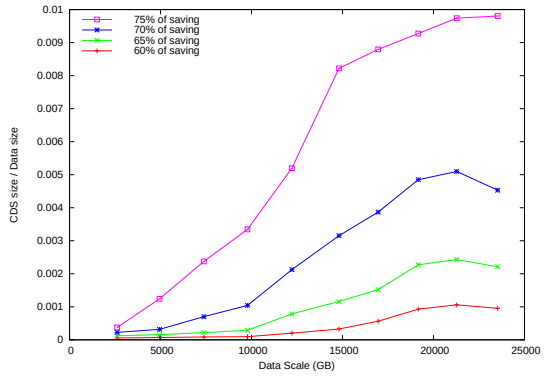

Figure 10: The relative size ratio of CDS over the data size.

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 cluster of 1,000 nodes, we expect that using 1% of data disks can cover more what we have seen from this 1323 VM dataset, assuming that the average behavior of every subcluster with 53 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.

6 Conclusions

The main contribution of this paper is a multi-level selective deduplication scheme among VM snapshots, using less than 0.5% of memory per node to meet a stringent cloud resource requirement. Inner-VM deduplication localizes backup data dependence and exposes more parallelism while global deduplication using a small common data set appeared in OS and data disks effectively covers a large amount of snapshot blocks. Our evaluation using Aliyun's data has shown that level 1 deduplication can reduce the storage need by 78% while level 2 inner fingerprint comparison contributes 4.5%Level 3 can add additional 10.5% reduction and accomplish 75% of what full fingerprint-based deduplication can do. 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.