National Conference on 'Innovative Research on Robotics, Circuits and Technology' (IRCT 2018)
Organized By : CIrcuit Branches of SCSVMV, (EEE, ECE, EIE & Mechatronics) , Kanchipuram, Tamil Nadu, India
In Assotiation with International Journal of Scientific Research in Science, Engineering and Technology
© 2018 IJSRSET | Volume 5 | Issue 1 | | Print ISSN: 2395-1990 | Online ISSN : 2394-4099

Analysis Prefetching Mechanism for VM Snapshot Deduplication Metadata in Cloud Storage

G. Poornima
Assistant Professor, Department Of. ECE, SCSVMV, Kanchipuram, Tamil Nadu, India

ABSTRACT

Cloud computing enables hardware and software resources to be accessed over the Internet. IaaS is one of the cloud services which offers computing power on demand by providing virtual machines to the consumers. As the number of users increase, IaaS cloud generates more number of VM images and snapshots. Hence, it is necessary to optimally utilize the storage space of IaaS to improve the performance. A well-known deduplication technique is used for efficient utilization of storage space. Since, the metadata generated is huge with the deduplication technique, it can only be maintained in the hard disk. Hence, prefetching a suitable subset of metadata is essential to improve I/O throughput. Thus, the objective of this paper is to compare the effectiveness of the similarity and locality indexing mechanisms with a common set of performance metrics to suggest a better indexing mechanism for IaaS cloud.

Keywords: Cloud storage, Deduplication, Storage optimization, Prefetching, Indexing mechanisms

I. INTRODUCTION

Cloud computing is a model that enables the consumers to access a shared pool of configurable computing resources over the Internet. It has several advantages that includes high availability, reliability, ease of management, disaster recovery and flexibility. Infrastructure as a Service (IaaS) is one of the cloud services that provides virtualized computing resources for the consumers to utilize. Due to the advantages of cloud computing, the adoption rate of IaaS consumers increases largely [15]. It has resulted in explosion of the number of virtual machine (VM) images and snapshots. While VM images create virtual machines, their corresponding VM snapshots preserve the state and data at a specific point in time. There may be different VM images related to different operating systems utilized by the consumers. Similarly, there may be several VM snapshots representing different states of virtual machines. As these VM images and VM snapshots are larger in size varying from several MBs to GBs, they occupy considerable amount of storage space.

In order to effectively utilize the storage space, it is necessary to identify and eliminate the redundant data among the various VM snapshots. A well-known optimization technique, namely, deduplication is helpful in eliminating the redundant data. It attempts to store only one instance of the data in the storage. The redundant data is substituted with a reference to the existing one in the storage. Though this technique saves storage space, the associated metadata overhead is huge. The metadata includes *Fingerprint Index* and *File Recipe* which are utilized to detect duplicates and to retrieve the files respectively. As this metadata is huge, it is typically saved in the hard disk. However, the challenge with inline deduplication of VM snapshot is to reduce the high latency due to the cost involved in accessing the metadata from the hard

disk. There are several indexing mechanisms that are available to prefetch a set of metadata entries into the RAM.Consequently, it becomes important to identify a better suited indexing mechanism for VM snapshot deduplication. Currently, these mechanisms have been analyzed individually by utilizing the performance metrics, namely, deduplication throughput and efficiency. However, the effectiveness of the prefetched set of metadata entries can be well estimated by the amount of space utilized in RAM and time taken to detect duplicates. Hence, the objective of this paper is to analyze the performance of the similarity and locality indexing mechanisms using a common set of evaluation metrics, namely, RAM footprint, duplicate detection time, deduplication throughput and space savings.

A. Contributions

The main contribution of this paper is to analyze the performance of similarity and locality indexing mechanisms for VM snapshot deduplication and to suggest a better suited indexing mechanism for IaaS cloud.

B. Paper Organization

The remainder of this paper is organized as follows. Section 2 reviews the related works in applying deduplication for VM snapshots. Section 3 describes the design of the performance evaluation system. Section 4 deals with the system implementation and Section 5 deals with the detailed performance analysis. Section 6 concludes the paper.

II. RELATED WORKS

Many research works have been proposed for prefetching the metadata into the RAM [7-13], as it is an important and challenging task in deduplication.

The pattern of the VM images is analyzed in the following two works. Jin et al. [6] have studied both the inter and intra similarity of various VM images. It is found from the experimental results that the VM images of same operating system or the versions of operating system are considered to have more duplicate blocks than the VM images of different operating system. Jayaram et al. [5] have performed an empirical analysis on 525 VM images of a public cloud storage. Both fixed and variable size chunking of various block sizes (4 KB, 8 KB, 16 KB, 32 KB and 64 KB) are applied on the VM images and it is seen that both the chunking mechanisms yield the same deduplication ratio. If the inter similarity among VM images is higher, more number of duplicates can be expected.

The Three Level Index (3LI) [1] is used to prefetch a suitable subset of fingerprints based upon the prefix from the disk. The fingerprints are organized in the hard disk in the form of several Hash node Tables (HTs). Each HT consists of fingerprint entries based on a certain prefix. The prefix of the fingerprint can be obtained by a 3LI which is maintained in the RAM. During the write operation, first, second and third 8 bits of any incoming fingerprint is compared against the entries present in the first, second and third level indices respectively using a comparator. If the prefixes are found, subset level will be checked. Otherwise, the new value is updated in the index. If the first 24 bit entry of the fingerprint matches in the 3LI, the corresponding HT is prefetched from the disk to RAM to detect duplicates. The motivation behind the 3LI is not clear.

In Two Level Index [2], the existing VM snapshots available in the disk are partitioned into 128 MB block groups. The groups are further divided into 4 KB blocks. The $(n+k)$ bit prefix fingerprint entries corresponding to every block in a block group and the block address where the fingerprint resides are maintained in 2LI. At first level, the first n bit of the fingerprint and a pointer to the second level is stored. Subsequently, the second level consists of the next k bits and the location of the block group. During the read of VM snapshots, every $(n+k)$ bit prefix of fingerprint is compared against the 2LI. If the $(n+k)$

bit prefix matches the 2LI, the block groups which have this fingerprint are prefetched into the RAM. These prefetched block groups help in identifying the future fingerprints. This speeds up the lookup process of VM snapshots. The 2LI prefetches more than one group based on the locality.

III. SYSTEM DESIGN

When deduplication is adopted in the cloud storage, it incurs the overhead of maintaining metadata, namely, *Fingerprint Index* and *File Recipe*. Since this metadata is huge, it is typically maintained in the disk. However, it increases the cost of finding the duplicates as it involves many disk seeks. Thus, a suitable indexing mechanism to prefetch the most relevant set of fingerprints is required to reduce the write latency for VM snapshots. Hence, the proposed Performance Evaluation System (PES) compares two existing indexing mechanisms to find a suitable one for VM snapshot deduplication. The PES consists of chunker, similarity detector, locality detector and the prefetcher as shown in Figure 1.

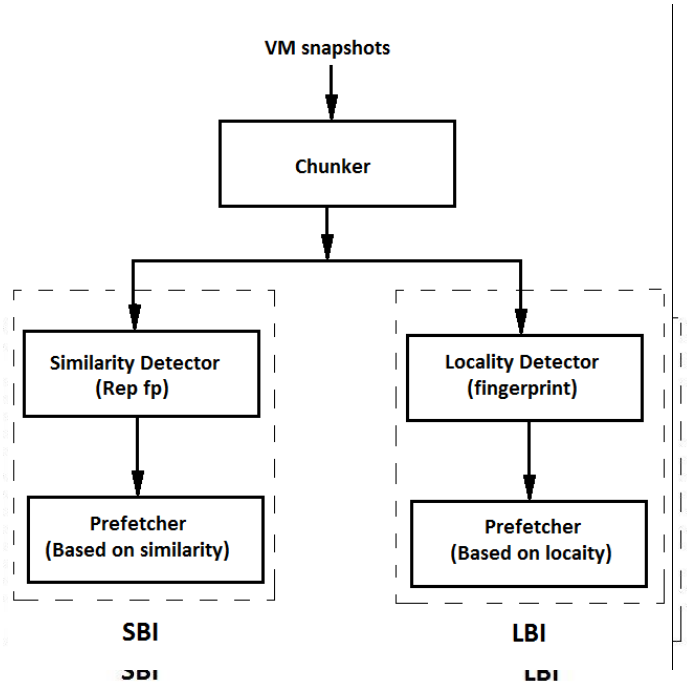

Figure 1. Design of PES

C. Chunker

The chunker divides every incoming VM snapshot into a set of fixed or variable sized blocks. Further, a cryptographic hash algorithm, namely, SHA1 is used to generate fingerprints corresponding to these blocks. These fingerprints are the unique identifiers for those corresponding blocks.

D. Similarity Based Index

The *Similarity Detector* utilizes Similarity Based Index (SBI) [3] to prefetch the fingerprints based on the similarity of files. Every incoming VM snapshot is divided into the variable sized blocks by using the variable-size chunking mechanism. Subsequently, the fingerprints are found for those blocks and the representative fingerprint is found. The representative fingerprint is the minimum fingerprint in a set of fingerprints corresponding to that VM snapshot. According to Broder's theorem, if the minimum fingerprints of two files are equal, the contents of two files are said to be similar up to 80%. The SBI consists of two levels as shown in Figure 2. For every VM snapshot, the first level consists of the fingerprint corresponding to the whole VM snapshot, the representative fingerprint and a pointer to the second level index for every VM snapshot. This representative fingerprint is helpful in finding the similar VM snapshots. The second level consists of the fingerprints corresponding to the similar VM snapshots.

During write operation, the *Similarity Detector* finds the fingerprint for the entire incoming VM snapshot. If a match is found, then the snapshot is found to be a duplicate. Otherwise, the snapshot is divided into a set of variable sized blocks. Further, the representative fingerprint of an incoming VM snapshot is compared against the first level. If it matches, then the incoming VM snapshot is similar. Hence, the fingerprints of VM snapshot are compared against the second level. Further, if the fingerprints are present in the second level index, then the

reference counts are updated. Otherwise, the entries are inserted into the second level index.

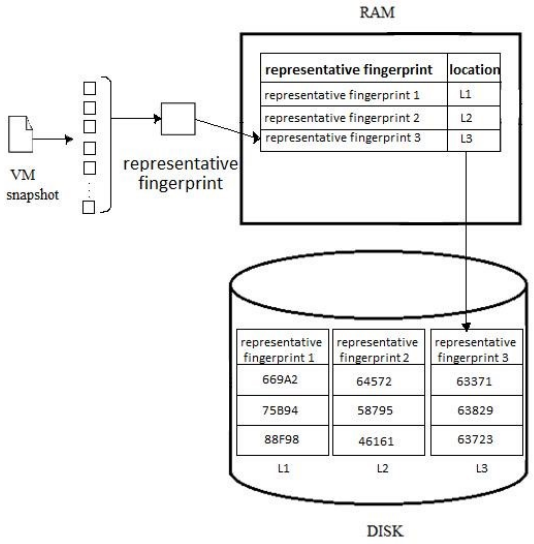

Figure 2. Similarity Based Index

E. Locality Based Index

In Locality Based Index (LBI) [4], every incoming VM snapshot is divided into a set of variable sized blocks. The fingerprints of these blocks are saved in the same order of arrival in the *fpstore* as shown in Figure 3. The *fpstore* and the blocks are maintained in a *container*. During the read operation of VM snapshots, when a fingerprint is matched in the *fpstore* of a *container*, the entire *fpstore* of the *container* is prefetched from the disk to RAM to match the future fingerprints by using Locality *detector*.

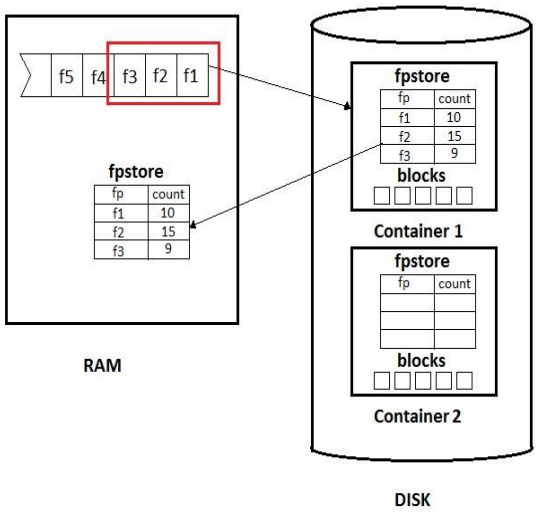

Figure 3. Locality Based Index

IV. SYSTEM IMPLEMENTATION

This section discusses the implementation and dataset used to construct SBI and LBI.

F. Implementation

The experimentation of the PES is performed on Intel Core i7 2.93 GHz machine with 8 GB RAM on Cent 64-bit operating system. To improve the precision of PES evaluation, each test is run three times under the same experimental settings.

In SBI, the fingerprint of the entire VM snapshot, representa -tive fingerprint and the location of the second level index are maintained as an in-memory *HashMap*. The representative fingerprint of the incoming VM snapshots is found and checked against the first in-memory *HashMap* table using *containsKey()* method of *HashMap* class. If a match is found, then the entire set of fingerprints corresponding to the incoming VM snapshot are compared against the second level in-memory *HashMap* table using *containsKey()* method of *HashMap* class.

In LBI, the fingerprints of every incoming VM snapshot are placed in a text file by processing according to the order of arrival. The text file represents the *fpstore*. Further, the text file and the corresponding blocks are stored in a folder which represents the *container*. When an initial fingerprint of the incoming VM snapshot matches the first fingerprint of the *container*, then the *fpstore* of the corresponding *container* is prefetched into the RAM. The fingerprints of the incoming VM snapshot are compared against the prefetched fingerprints using *containsKey()* method of *HashMap* class for deduplication. If a match is not found, then the fingerprints of the incoming snapshot are saved into a new *container*.

G. Dataset

The dataset for PES is a set of VM snapshots collected

from VMware cloud data center [14] as shown in Table 1. This dataset consists of VM snapshots of most popular operating systems, namely, Debian, Fedora, Mint, OpenSUSE and Ubuntu. This dataset is utilized for analyzing the performance of both indexing mechanisms.

Table 1. VM snapshots dataset

OS Type	No. of Snap-shots	Size (in MB)
Debian	10	28.25
Fedora	30	84.9
Mint	20	48.5
OpenSUSE	15	54.25
Ubuntu	50	86.1
Total	125	302.25

V. RESULTS AND DISCUSSION

This section deals with the detailed performance analysis of SBI and LBI mechanisms by utilizing a common set of evaluation metrics.

H. Metrics for evaluvation

The indexing mechanisms, namely, SBI and LBI have been compared using a set of common metrics, namely, RAM footprint, duplicate detection time, space savings and deduplication throughput.

1) **RAM footprint**

The fingerprints prefetched from the disk utilize a considerable amount of RAM memory during deduplication. The performance of a system depends on the utilization of the RAM memory.

2) **Duplicate detection time**

During write operation, the fingerprints of every incoming VM snapshot are compared with the prefetched fingerprints in order to find duplicates. The time taken to perform this operation is the duplicate detection time. The lower the duplicate detection time is, the higher will be the performance of the indexing mechanism

3) **Space savings**

Space savings refers to the amount of disk space saved after the application of deduplication. It is measured by the difference in the size of the snapshot before and after deduplication. The space saving depends on the prefetched subset of fingerprints.

4) **Deduplication throughput**

Deduplication throughput is found to evaluate the effectiveness of the indexing mechanisms. It represents the number of blocks written into the disk per unit time. The higher the deduplication throughput is, the higher will be the performance of the indexing mechanism.

I. *Analysis of PES*

The dataset has been split into two parts, namely, index building dataset and a deduplicating dataset. The index building dataset consists of 30 VM snapshots of size 50 MB. The average size of a VM snapshot is 10 MB. The deduplicating dataset consists of 20 VM snapshots of size 25 MB. Experiments have been conducted to build the indices SBI and LBI by utilizing the index building dataset. Further, the deduplicating dataset has been given as input to analyze the performance of the indexing mechanism.

1) **Comparison of indexing schemes for RAM footprint**

The fingerprints prefetched from disk are maintained at in-memory *HashMap* table. The RAM utilization is found by using Runtime class in Java library.

Figure 4. Average RAM footprint

While LBI prefetches the fingerprints of a single *container*, SBI prefetches the fingerprints of all similar VM snapshots. Hence, the average RAM footprint of the LBI is less when compared to that of SBI as shown in Figure 4.

2) **Comparison of indexing schemes for duplicate detection time**

A Timer *startTimer* is started immediately after chunking the snapshots into blocks by using chunker and *endTimer* is stopped immediately after duplicate detection using *check(fingerprint)*. The difference between the *startTimer* and *endTimer* gives the duplicate detection time in millisecond by using *System.currentTimeMillis()* of Java library. In order to improve the accuracy, it is also measured in nanosecond, *startNanoTimer* and *endNanoTimer* is calculated by using *System.nanoTime()* and *System.currentTimeMillis()* method of Java library.

Since the considered workload for deduplication is characterized by the VM snapshot corresponding to Linux distribution, the inter-similarity was more. Hence, the probability of the fingerprints of incoming VM snapshots to get matched with entries in the SBI is more when compared to LBI as shown in Figure 5.

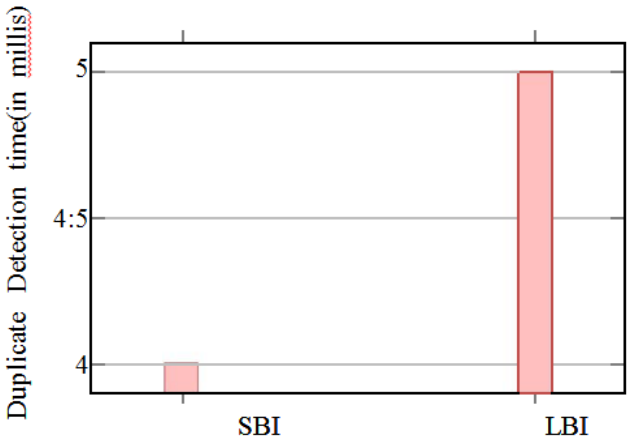

Figure 5. Average duplicate detection time

3) **Comparison of indexing schemes for space savings**

Space savings signifies the amount of storage space used by the snapshot by utilizing the index. The size of the directory is measured by using *FileUtils.sizeOfDirectory()*. The files used for deduplication is saved in *filesUsed* directory and the files are chunked and saved into the *chunkDirectory*. The difference between the *filesUsed* and *chunkDirectory* returns the space saving. The number of duplicates detected for SBI is more when compared to LBI. Hence, the average space savings of SBI is higher than LBI as shown in Figure 6.

4) **Comparison of indexing schemes for deduplication throughput**

The throughput is measured in terms of number of blocks written per second by using *DIRECTORYNAME.length()*. The throughput has been found by using sample dataset as shown in Table 1.

The number of fingerprints prefetched from the disk is maintained as in memory *HashMap* table. The SBI prefetches the fingerprints based on the similar VM snapshots, whereas LBI prefetches the fingerprints corresponding to a particular snapshot. Due to the higher inter similarity between the snapshots, the SBI prefetches more number of blocks when compared with LBI. Hence, the number of blocks prefetched from the disk is higher for SBI when compared to

LBI. The throughput analysis of the indexing mechanism is shown in Figure 7.

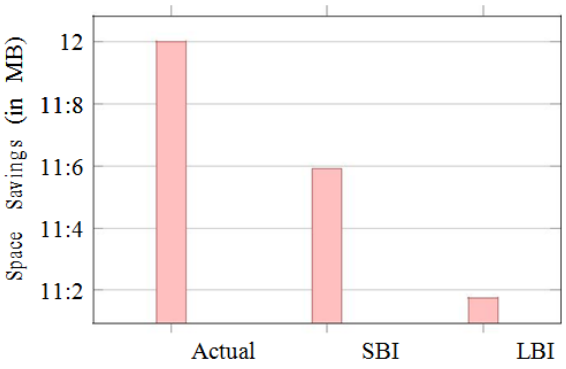

Figure 6. Average space savings

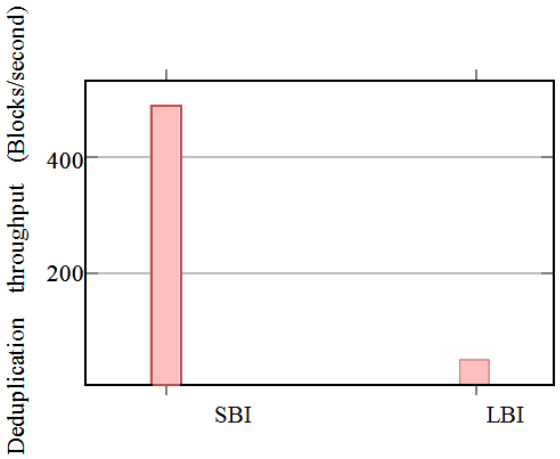

Figure 7. Average deduplication throughput

VI. CONCLUSION

The existing indexing mechanisms, SBI and LBI have been implemented. The index building dataset of size 50 MB has been utilized to build the indices. Further, the deduplicating dataset of size 25 MB has been given as the input to perform deduplication. During deduplication, the average RAM footprint consumed, the time involved for detecting duplicates, space occupied for storing the deduplicating dataset and the numbers of blocks written per second for each indexing mechanism have been found. From the experimental investigation, it is found that the RAM footprint, duplicate detection time and the throughput of SBI are 15%, 20% and 90% more and space savings is 5% less when compared to that of LBI.

VII. REFERENCES

[1]. J. Wang, Z. Zhao, Z. Xu, H. Zhang, L. Li and Y. Guo, I-sieve: An inline high performance deduplication system used in cloud storage, In Proceedings of IEEE International Symposium on Tsinghua Science and Technology, 2015, pp.17-27.

[2]. C. H. Ng, M. Ma, T. Y. Wong, P. P. Lee and J. Lui, Live deduplication storage of virtual machine images in an open-source cloud, In Proceedings of 12th International Middleware Conference, International Federation for Information Processing, 2011, pp. 80-99.

[3]. D. Bhagwat, K. Eshghi, D. D. Long and M. Lillibridge, Extreme binning: Scalable, parallel deduplication for chunk-based file backup, In Proceedings of Modeling, Analysis and Simulation of Computer and Telecommunication Systems, IEEE, 2009, pp. 1-9.

[4]. B. Zhu, K. Li and R. H. Patterson, Avoiding the Disk Bottleneck in the Data Domain Deduplication File System, In Proceedings of File And Storage Technologies, 2008, Vol. 8, pp. 1-14.

[5]. K. R. Jayaram, C. Peng, Z. Zhang, M. Kim, H. Chen and H. Lei, An empirical analysis of similarity in virtual machine images, In Proceedings of Middleware Industry Track Workshop, 2011, pp.1-6.

[6]. K. Jin and E. L. Miller, The effectiveness of deduplication on virtual machine disk images, In Proceedings of Israeli Experimental Systems Conference, 2009, pp.1-7.

[7]. J. Xu, W. Zhang, S. Ye, J. Wei, and T. Huang, A lightweight virtual machine image deduplication backup approach in cloud environment in proceedings of Computer Software and Applications Conference (COMPSAC), IEEE,2014, pp. 503-508.

[8]. W. Zhang, H. Tang, H. Jiang, T. Yang, X. Li, and Y. Zeng, Multi-level selective deduplication for

vm snapshots in cloud storage in proceedings of Cloud Computing (CLOUD), 2012, pp. 550-557.

[9]. T. T. Thwel and N. L. Thein, An efficient indexing mechanism for data deduplication in proceedings of Current Trends in Information Technology (CTIT) International Conference, 2009, pp. 1-5.

[10]. Zhao, X., Zhang, Y., Wu, Y., Chen, K., Jiang, J. and Li, K., Liquid: A scalable deduplication file system for virtual machine images. Parallel and Distributed Systems, 2014, pp.1257-1266.

[11]. Zhang, W., Agun, D., Yang, T., Wolski, R. and Tang, H., VM-centric snapshot deduplication for cloud data backup in proceedings of Mass Storage Systems and Technologies, 2015, pp. 1-12.

[12]. Campello, D., Crespo, C., Verma, A., Rangaswami, R. and Jayachandran, P., Coriolis: Scalable VM Clustering in Clouds in proceedings of ICAC, 2013, pp. 101-105.

[13]. Debnath, B.K., Sengupta, S. and Li, J., ChunkStash: Speeding Up Inline Storage Deduplication Using Flash Memory in proceedings of USENIX annual technical conference, 2010.

[14]. http://www.trendsigma.net/vmware/

[15]. http://www.rightscale.com/blog/cloud-industry-insights/iaas-vs-paas-2015-cloud-trends-state-cloud-survey