International Journal of Emerging Technology in Computer Science & Electronics (IJETCSE)

ISSN: 0976-1353 Volume 14 Issue 1 –APRIL 2015.

# BLOOM FILTERING DE DUPLICATION FOR PATTERN RECOGNITION

RAHINI.R#

*M.E. - Computer Science and Engineering, A.V.C. College of Engineering, Mayiladuthurai, India.*

***Abstract* -Virtualization is becoming widely deployed in servers to efficiently provide many logically separate execution environments while reducing the need for physical servers. This approach saves physical CPU resources, it still consumes large amounts of storage because each virtual machine (VM) instance requires its own multi-gigabyte disk image. Existing systems have made efforts to reduce VM image storage consumption by means of de duplication within a Storage Area Network (SAN) cluster. SAN cannot satisfy the increasing demand of large-scale VM hosting because of its cost limitation. Liquid provides a scalable de duplication file system that has been particularly designed for large-scale VM deployment. Its design provides fast VM deployment with peer-to-peer (P2P) data transfer and low storage consumption by means of de duplication on VM images. It also provides a comprehensive set of storage features including instant cloning for VM images, on-demand fetching through a network, and caching with local disks by copy-on-read techniques. Experiments show that Liquid's feature perform well and introduce minor performance overhead.**

***Index Terms : Virtual machine, Data Block Storage ,Shared Caching ,Copy on Read ,P2P Data Block Sharing***

## I. INTRODUCTION

A virtual machine is an emulation of a particular computer system. Virtual machines operate based on the computer architecture and functions of a real or hypothetical computer, and their implementations may involve specialized hardware, software, or a combination of both. virtual machine provide a complete substitute for the targeted real machine and a level of functionality required for the execution of a complete operating system. Virtualization is widely used because it provides many logically separate execution environment while reducing the need of physical servers. It still consumes large amounts of storage because each virtual machine instance requires its own multi-gigabyte disk image. Existing systems have made efforts to reduce VM image

storage consumption by means of deduplication within a storage area network (SAN) cluster. an SAN cannot satisfy the increasing demand of large-scale VM hosting for cloud computing because of its cost limitation. Liquid is a technique that provides a scalable de duplication file system that has been particularly designed for large-scale VM deployment. Its design provides fast VM deployment with peer-to-peer (P2P) data transfer and low storage consumption by means of deduplication on VM images. It also provides a set of storage features including instant cloning for VM images, on-demand fetching through a network, and caching with local disks by copy-on-read technique. These are the features achieved by dimensional Bloom Filter, its a space-efficient probabilistic data structure that is used to test whether an element is a member of a set or not.

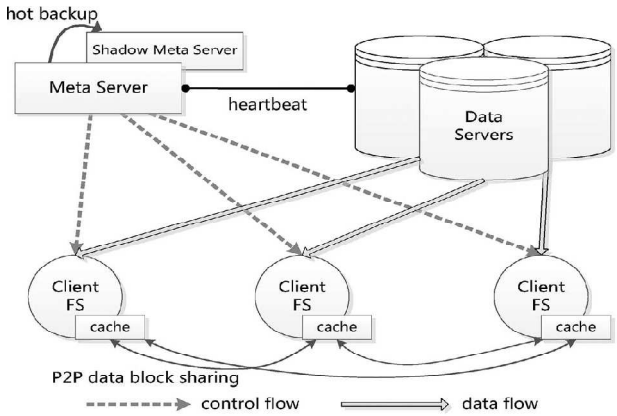



Fig 1. Deduplication without Bloom Filter

## II.RELATED WORK

In [1] C. Tang define that FVD is a holistic solution for both Cloud and non-Cloud environments. Its feature set includes flexible configurability, storage thin provisioning without a host file system, compact image, internal snapshot, encryption, copy-on-write, copy-on-read, and adaptive prefetching. The last two features enable instant VM creation and instant VM migration, even if the VM image is stored on direct-attached storage. The solution in FVD is to do copy-on-read (CoR) and adaptive prefetching, in addition to copy-on-write (CoW). CoR avoids repeatedly reading a data block from NAS, by saving a copy of the returned data on DAS for later reuse. Adaptive prefetching uses resource idle time to copy from NAS to DAS the image data that have not been accessed by the VM.FVD also supports instant VMmigration, even if the VM's image is stored on DAS. FVD can instantly migrate a VMwithout first transferring its disk image. As the VM runs uninterruptedly on the target host, FVD uses CoR and adaptive prefetching to gradually move the image from the source host to the target host, without user perceived downtime. This FVD only support neither copy-on-read (CoR) nor adaptive pre-fetching. Some virtualization solutions do support CoR or pre-fetching, but they are implemented for specific use cases.

In [2] K. Jin and E.L. Miller describe that Deduplication is an efficient approach to reduce storage demands in environments with large numbers of VM disk images. Deduplication of VM disk images can save 80% or more

of the space required to store the operating system and application environment; it is particularly effective when disk images correspond to different versions of a single operating system"lineage," such as Ubuntu or Fedora.The use of Deduplication to both reduce the total storage required for VM disk images and increase the ability of VMs to share disk blocks. This technique currently working to evaluate the amount of locality available in disk images. If there is significant locality, this can improve deduplication performance by co-locating nearby chunks from one disk image, since those chunks will likely be near one another in other disk images as well. Algorithm used for this technique is called DEFLATE algorithm, it is a data compression algorithm that uses a combination of the LZ77 algorithm and Huffman coding.

In [3] A. Liguori and E. Hensbergen proposed A content addressable storage (CAS) approach to coalescing duplicate data between multiple disk images used to provide a solution to the incremental drift of virtual disks. CAS solutions lend themselves to rapid cloning, snapshoting, and can be configured to implicitly provide temporal-based backups of images. In order to get a better idea of the performance and efficiency implications of using a CAS based image management system, constructed a prototype by CAS back end with a service-oriented file system to provide an organizational infrastructure and tested it with guest logical partition ns running under QEMU which provides the Virtual I/O infrastructure for KVM. A prototype for consolidating virtual disk images using a service-oriented file system. It provides a hierarchical organization, manages historical snapshots of drive images, and takes steps to optimize encoding based on partition type and file system. Present these experiences with building this prototype and using it to store a variety of drive images for QEMU and the Linux Kernel Virtual Machine (KVM). It was primarily developed as a backup archive server, and its implementation is single threaded and not constructed to scale under heavy load. Additionally, its performance is primarily bottlenecked by the requirement of in-directing block requests via the index. Content addressable storage uses SHA-1 for cryptographic hash function it is based on message digest principle.

**In [4]** C. Ng, M. Ma, T. Wong, P. Lee, and J. Lui created A live deduplication file system called LiveDFS, which enables deduplication storage of VM image files in an open-source cloud. In particular, target the open-source cloud platforms that are deployed in low-cost commodity hardware and operating systems. LiveDFS supports general file system operations, such as read, write, delete, while allowing inline deduplication. LiveDFS consists of several design features that make deduplication efficient and practical. LiveDFS has several distinct features, including spatial locality, prefetching of metadata. LiveDFS is POSIX compliant and implemented as a Linux kernel-space file system. Deploy our LiveDFS prototype as a storage layer in a cloud platform based on OpenStack, and conduct extensive experiments.A live deduplication file system is an open-source cloud that is deployed with low-cost commodity hardware configurations with limited memory foot prints. Since a cloud platform is typically a distributed system ,there is a plan to extend LiveDFS in a distributed setting. One challenging issue is to balance the trade-off between storage efficiency and fault tolerance. On one hand, deduplication reduces the storage space by removing redundant data copies on the other hand, it sacrifices fault tolerance with the elimination of redundancy.

**In [5]** B. Zhu, K. Li, and H. Patterson analyzed the production Data Domain deduplication file system to relieve the disk bottleneck. These techniques include Summary Vector, a compact in-memory data structure for identifying new segments. Stream-Informed Segment Layout, a data layout method to improve on-disk locality for sequentially accessed segments and Locality Preserved Caching, which maintains the locality of the fingerprints of duplicate segments to achieve high cache hit ratios .To implementt a high-throughput Identical Segment Deduplication storage system at low cost. The key performance challenge is finding duplicate segments .An in-memory index of all segment fingerprints could easily achieve this performance, but the size of the index would limit system size and increase system cost. Early deduplication system can achieve only limited global compression and not duplication in every single client FS. Techniques used here is Stream-Informed Segment Layout and Locality Preserved Caching. SISL identifies the duplicate segment in Ram, inline before storing to disk. It stores related segments and finger prints together so large group can be read at once.A Locality Preserving dictionary maintains elements of similar key values stored close together for fast access with ranges of data with consecutive keys.

**In [6]** A.T. Clements, I. Ahmad, M. Vilayannur, and J. Li says that File systems hosting virtual machines typically contain many duplicated blocks of data resulting in wasted storage space and increased storage array cache footprint. Deduplication addresses these problems by storing a single instance of each unique data block and sharing it between all original sources of that data. While deduplication is well understood for file systems with a centralized component,these are the techniques investigated in a decentralized cluster file system, specifically in the context of VM storage. DEDE, a block-level deduplication system for live cluster file systems that does not require any central coordination, tolerates host failures, and takes advantage of the block layout policies of an existing cluster file system. In DEDE, hosts keep summaries of their own writes to the cluster file system in shared on-disk logs. Each host periodically and independently processes the summaries of its locked files, merges them with a shared index of blocks, and reclaims any duplicate blocks. DEDE manipulates metadata using general file system interfaces without knowledge of the file system implementation.This techniques are applicable beyond virtual machine storage and plan to examine DEDE in other settings in the future. For example, high-frequency deduplication could prevent temporary file system bloat during operations that produce large amounts of duplicate data (e.g., mass

software updates), and deferral of merge operations could help reduce file system fragmentation. future plan explore the trade-offs mentioned in this paper, such as block size versus metadata overhead, in-band versus out-of-band hashing, and sequential versus random index updates. DEDE represents just one of the many applications of deduplication to virtual machine environments.The next step for deduplication is to integrate and unify its application to file systems, memory compression, network bandwidth optimization, achieve end-to-end space and performance optimization.

## III. DESCRIPTION OF THE SCHEME

### A. *De Duplicator For Pattern Recognition*

Data deduplication is a specialized data compression technique for eliminating duplicate copies of repeating **data**. De Duplicator consists of three components, a single meta server with hot backup, multiple data servers, and multiple clients . Each of these components is typically a commodity Linux machine

running a user-level service process.VM images are split into fixed size data blocks. Each data block is identified by its unique fingerprint, which is calculated during deduplication process. Liquid represents a VM image via a sequence of fingerprints which refer to the data blocks inside the VMimage.

*B.Meta Server*

A MetaServer is considered a central broker providing a collated view for dispersed web resources. It is used to collect data from various web services, web pages, databases, or other online resources/repositories and then present the combined results to the client using a standard web protocol

The meta server maintains information of file system layout. This includes file system namespace, fingerprint of data blocks in VM images, mapping from fingerprints to data servers, and reference count for each data block. To ensure high availability, the meta server is mirrored to a hot backup shadow meta server.

*C.Data server*

A database server is a computer program that provides database services to other computer programs or computers. The data servers are in charge of managing data blocks in VM images. They are organized in a distributed hash table (DHT) fashion, and governed by the meta server. Each data server is assigned a range in the fingerprint space by the meta server. The meta server periodically checks the health of data servers, and issues data migration or replication instructions to them when necessary. Data server provide image content to hypervisors.

*D. Heartbeat protocol*

The meta server in Liquid is in charge of managing all data servers. It exchanges a regular heartbeat message with each data server, in order to keep an up to date vision of their health status.

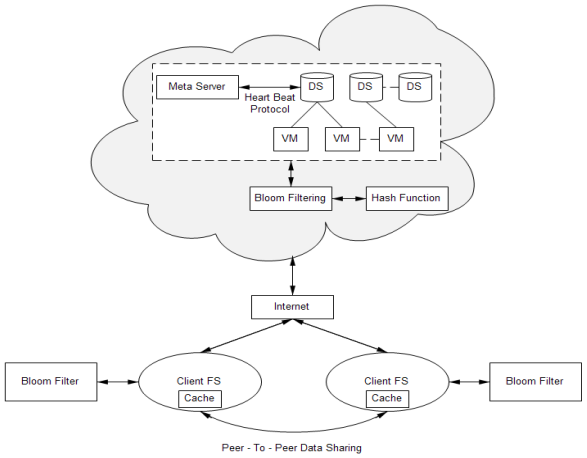



Fig 2. Liquid Architecture

The meta server exchanges heartbeat messages with data servers in a round-robin fashion. This approach will be slow to detect failed data servers when there are many data servers. To speedup failure detection, whenever a data server or client encounters connection problem with another data server, it will send an error signal to the meta server.

## IV. PERFORMANCE OF BLOOM FILTER

A Bloom filter is a space-efficient probabilistic data structure ,that provides 100% pattern recognition for pattern matching. A Bloom filter uses an array of m bits, all set to 0 initially, to represent the existence information of n fingerprints. k different hash functions must also be defined, each of which maps a fingerprint to one of the m array positions randomly with a uniform distribution. When adding a new fingerprint into the Bloom filter, the k hash functions are used to map the new fingerprint into k bits in the bit vector, which will be set as 1 to indicate its existence. To query for a fingerprint, this feed it to each of the k hash functions to get k array positions. If any of the bits at these positions is 0, the element is definitely not in the set; otherwise, if all are 1, then either the element is in the set, or the bits have been to 1 by chance during the insertion of other elements. An example is given in Fig. 3, representing the set x; y; z. The olid, dashed, and dotted arrows show the positions in the bit array that each set element is mapped to. The element w is not in the set x; y; z, because it hashes to one bit-array position containing 0.

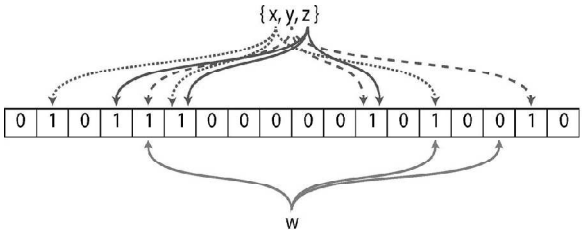



Fig 3.Bloom Filter Data Structure

*A.Virtual Machine*

Cloud provide VM service for all user, to create a VM images for users disk size and VM images are split into fixed size data blocks. Each data block is identified by its unique fingerprint, which is calculated during deduplication process. Liquid represents a VM image via a sequence of fingerprints which refer to the data blocks inside the VMimage.

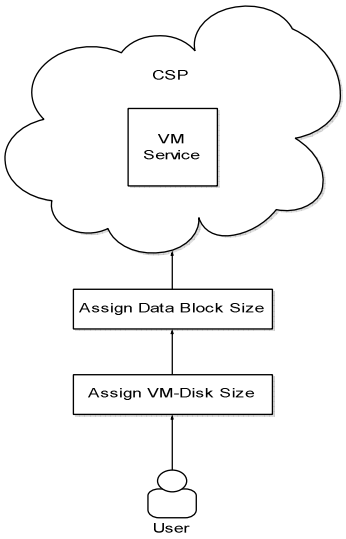



Fig 4.VM creation

*B. Data Block Storage*

Data storage to implement its own storage for data blocks. Data blocks are split into groups according to their fingerprints. Three files are created for each group, i.e., an extent file containing all the data blocks' content, an index file mapping a fingerprint to corresponding data block's offset and reference count, and a bitmap file indicating if certain slot in the extent file is valid. The bitmap file and index file are small enough to be loaded into memory. For example, with 256 KB data block size, and 1 TB size of unique data blocks, we will only have an index file size of 80 MB, and a bitmap file size of 512 KB.

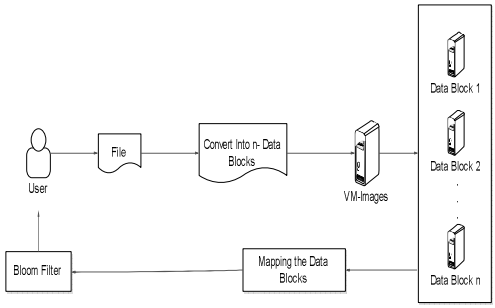



Fig 5.Data block storage

*C.Shared Cache*

The client side of Liquid file system maintains a shared cache, which contains recently accessed data blocks. Data blocks in a shared cache are read-only and shared among all VMimages currently opened. When being requested for a data block, Liquid first tries to look it up in a shared cache. A cache miss will result in the requested data block being loaded into the shared cache. When the shared cache is filled up, cached data blocks are replaced using the least recently used (LRU) policy. This layer of caching mechanism improves reading performance of VM images and ensures smooth operation of VMs.

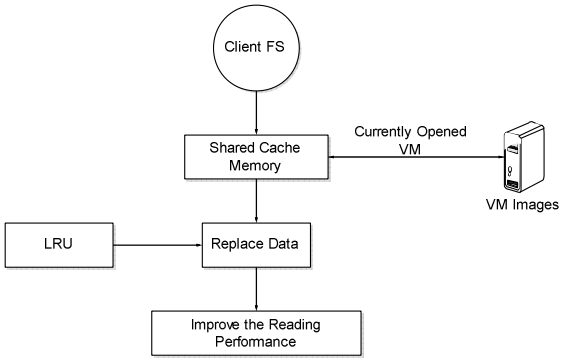



Fig 6. Shared Cache

*D. Copy on Read*

Liquid uses the copy-on-read technique to bring data blocks from data servers and peer clients to local cache on demand as they are being accessed by a VM. This technique allows booting a VM even if the data blocks in the VM image have not been all fetched into local cache, which brings significant speedup for VM boot up process. Moreover, since only the accessed portion of data blocks are fetched, network bandwidth consumption is kept at a low rate, which is way more efficient than the approach of fetching all data blocks into local cache and then booting the VM.

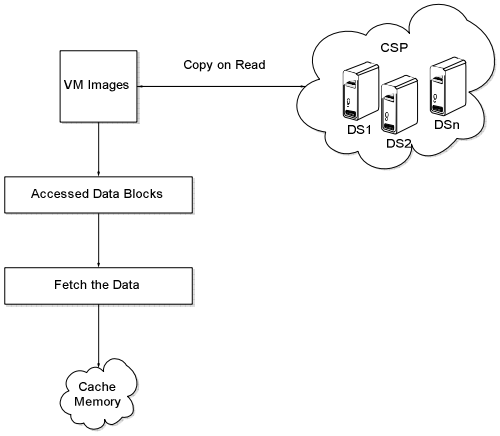



Fig 7. Copy On Read

*E. P2P Data Block Sharing*

Liquid is its P2P data block sharing scheme. Liquid alleviates burden on data servers by sharing data blocks among all client nodes in a peer-to-peer fashion, eliminating network IO bottlenecks. Liquid implements its own peer-to-peer distribution protocol with inspiration from BitTorrent protocol. Each client node or a data server is a valid data block provider, and publishes a Bloom filter where the fingerprints of all their data blocks are compacted into it.

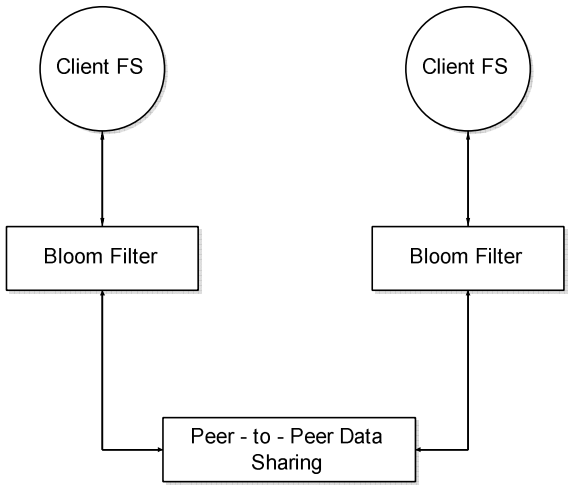


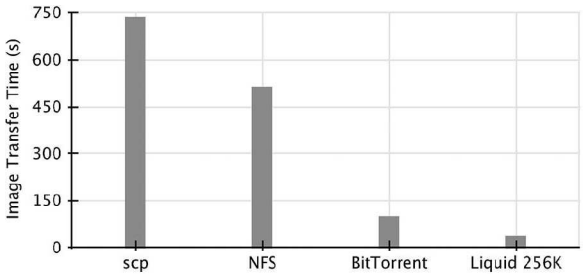

Fig 8. P2P data block Sharing

## V.IMPLEMENTATION AND RESULTS

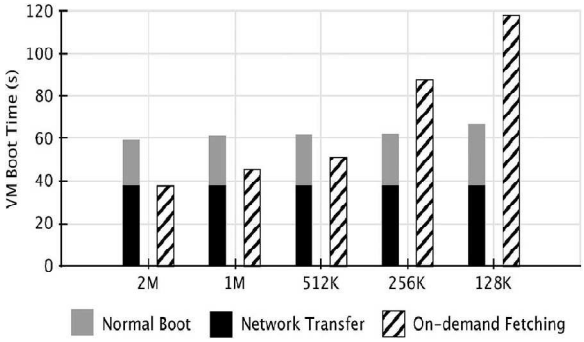
The common distributed storage systems, HYDRAstor andMAD2 propose effective distributed architectures for deduplication. The former uses distributed hash table to distribute data, and the latter uses Bloom Filter Array as a quick index to quickly identify non-duplicate incoming data.LiveDFS enables deduplication storage of VM images in an open-source cloud; however, it only focuses on deduplication on a single storage partition, which is difficult to handle the problems in distribute systems.

Amazon's Dynamo used DHT to organize its content. Data on the DHT are split into several virtual nodes, and are migrated for load balance in unit of virtual nodes. Liquid follows this approach by splitting data blocks into shards according to their fingerprint, and management of data in unit of shards. Lustre is a parallel and distributed file system, generally used for cluster computing. The architecture of Lustre file system is similar to Liquid, but there are several important differences. Liquid file system has reduced storage consumption by using deduplication technology, and solve the bottleneck problem of metadata server owing to our P2P data block sharing scheme among all client nodes.

GFS , HDFS , and OpenStack provides high availability by replicating stored data into multiple chunk servers. In Liquid, every client is also a replica storing data blocks frequently used. This indicates that Liquid has high fault tolerance capability. Even if all data servers are down, a client would still be able to fetch data block from peer clients. Moreover, compared with these systems, our Liquid has reduced storage consumption by 44 percent at 512KB data block size, eliminating the influence of back-up copy. Meanwhile, the I/O performance loss is just less than 10 percent.




Direct copying by the scp utility takes the longest time, with the source node being a hot spot and its network bandwidth saturated .Liquid avoids fetching redundant data blocks multiple times, and achieves much better distribution speed than plain BitTorrent protocol. Redundancy is not utilized by BitTorrent protocol, and all the redundant zero data blocks are all transferred through a network.




Compared with normal VM booting where all data blocks inside the VM image have already been fetched to local cache, VMbooting with on-demand fetching takes several times longer duration.

## VI.CONCLUSION

Liquid provides good IO performance while doing de duplication work in the meantime. This is achieved by caching frequently accessed data blocks in memory cache, and only run deduplication algorithms when it is necessary. Liquid supports instant VM image cloning by copy-on write technique, and provides on-demand fetching through network, which enables fast VM deployment. P2P technique is used to accelerate sharing of data blocks, and makes the system highly scalable. Periodically exchanged Bloom filter of data block fingerprints enables accurate tracking with little network bandwidth consumption.

Deduplication on VM images is proved to be highly effective. However, special care should be taken to achieve high IO performance. On demand Caching blocks avoid running expensive deduplication algorithms frequently, thus improves IO performance. Making the system scalable by means of P2P technique is challenging because of the sheer number of data blocks to be tracked. By compacting data block fingerprints into Bloom filters, the management overhead and meta data transferred over network could be greatly reduced.

Future work of LIQUID technique is to Improve the performance of write operation in because write performance is degraded due to the frequent need to allocate new data block, also to develop a Multi Dimensional Vector in Bloom Filtering algorithm, this technique improves the data storage and avoid the duplication in large file system efficiently. Mainly these are the techniques used to adapt for any bit operating system with Scheduling on large scale file system on MapReduce. Further work on liquid try to reduce overhead of block size and metadata for storing and migrating large size of file systems, also focus on in-band and out-of-band hashing, and sequential random index updates.

## VII.REFERENCES

[1] J. Katcher, ‘‘Postmark: A New File System Benchmark,’’ Network Applicance Inc., Sunnyvale, CA, USA, Technical Report TR3022,Oct. 1997.
[2] A. Liguori and E. \Hensbergen, ‘‘Experiences with Content Addressable Storage and Virtual Disks,’’ in Proc. WIOV08,San Diego, CA, USA, 2008.
[3] S. Bhattacharya, M. Cao, A. Dilger, A. Mathur, A. Tomas, and L. Vivier, ‘‘The New ext4 Filesystem: CURRENT STATUS and Future Plans,’’ in Proc. Linux Symp. 2007.
[4] K. Jin and E.L. Miller, ‘‘The Effectiveness of Deduplicatin on VirtualMachin Disk Images,’’2009.
[5] T. Wong, P. Lee, and J. Lui, ‘‘Live Deduplication Storage of Virtual Machine Images in an Open-Source Cloud,’’ in Proc. Middleware, 2011.
[6] K. Pepple, ‘‘ Deploying OpenStack. Sebastopol”, CA, USA: O’Reilly Media 2011.
[7] S. Quinlan and S. Dorward, ‘‘Venti: A New Approach to Archival Storage,’’ in Proc. FAST Conf. File Storage Technol., 2002.
[8] I. Ahmad ,A.T. Clements, M. Vilayannur, and J. Li, ‘‘Decentralized Deduplication in San Cluster File Systems,’’ in Proc. Conf. USENIXAnnu. Techn. Conf, 2009.
[9] P. Schwan ,‘‘Lustre: Building a File System for 1000-Node Clusters,’’ in Proc. Linux Symp, 2003.
[10] R. Chansler, , H. Kuang, S. Radia, and K. Shvachko ‘‘TheHadoop Distributed File System,’’ in Proc. IEEE 26th Symp. MSST, 2010.

**Rahini R** (rahinir91@gmail.com) received her B.E. degree of computer science from Periyar Maniammai University,Vallam, Tamilnadu. She currently is an M.E. candidate with the department of computer science. Her area of interests includes deduplication in cloud computing.