Dynamic Data Deduplication in Cloud Storage

Waraporn Leesakul, Paul Townend, Jie Xu
School of Computing
University of Leeds, Leeds, LS2 9JT
United Kingdom
{scwl, p.m.townend, j.xu} @leeds.ac.uk

Abstract—**Cloud computing plays a major role in the business domain today as computing resources are delivered as a utility on demand to customers over the Internet. Cloud storage is one of the services provided in cloud computing which has been increasing in popularity. The main advantage of using cloud storage from the customers' point of view is that customers can reduce their expenditure in purchasing and maintaining storage infrastructure while only paying for the amount of storage requested, which can be scaled-up and down upon demand. With the growing data size of cloud computing, a reduction in data volumes could help providers reducing the costs of running large storage system and saving energy consumption. So data deduplication techniques have been brought to improve storage efficiency in cloud storages. With the dynamic nature of data in cloud storage, data usage in cloud changes overtime, some data chunks may be read frequently in period of time, but may not be used in another time period. Some datasets may be frequently accessed or updated by multiple users at the same time, while others may need the high level of redundancy for reliability requirement. Therefore, it is crucial to support this dynamic feature in cloud storage. However current approaches are mostly focused on static scheme, which limits their full applicability in dynamic characteristic of data in cloud storage. In this paper, we propose a dynamic deduplication scheme for cloud storage, which aiming to improve storage efficiency and maintaining redundancy for fault tolerance.**

Keywords— Cloud Computing; Cloud storage; Deduplication; Dependability; Avaiability

I. INTRODUCTION

Cloud computing has recently emerged as a popular business model for utility computing system. The concept of cloud is to provide computing resources as a utility or a service on demand to customers over the Internet. The concept of cloud computing is quite similar to grid computing, which aims to achieve resource virtualisation [1]. In grid computing, the organisations sharing their computing resources, such as processors, in order to achieve the maximum computing capacity, whereas cloud computing aims to provide computing resources as a utility on demand, which can scale up or down at any time, to multiple customers. This makes cloud computing play a major role in the business domain, whereas grid is popular in academic, scientific and engineering research [2].

Many definitions of cloud computing have been defined, depended on the individual point of view or technology used for system development. In general, we can define cloud computing as a business model that provide computing resources as a service on demand to customers over the Internet [3].

The essential characteristics of cloud computing have been defined in [3]. Cloud providers *pool computing resources* together to serve customers via a multi-tenant model. Computing resources are delivered over the Internet where customers can access them through *various client platforms*. Customers can access the resources *on-demand* at any time without human interaction with the cloud provider. From a customers' point of view, computing resources are infinite, and customer demands can rapidly change to meet business objectives. This is facilitated by the ability for cloud services to *scale resources up and down* on demand leveraging the power of virtualization. Moreover, cloud providers are able to *monitor and control* the usage of resources for each customer for billing purposes, optimization resources, capacity planning and other tasks.

Cloud storage is one of the services in cloud computing which provides virtualized storage on demand to customers. Cloud storage can be used in many different ways [4]. For example, customers can use cloud storage as a backup service, as opposed to maintaining their own storage disks. Organisations can move their archival storage to the cloud which they can achieve more capacity at the low-cost, rather than buying additional physical storage. Applications running in the cloud also require temporary or permanent data storage in order to support the applications.

As the amount of data in the cloud is rapidly increasing, customers expect to reach the on-demand cloud services at any time, while providers are required to maintain system availability and process a large amount of data. Providers need a way to dramatically reduce data volumes, so they can reduce costs while saving energy consumption for running large storage systems. Similar to other storages, storage in cloud environments can also use data deduplication technique.

Data deduplication is a technique whose objective is to improve storage efficiency. With the aim to reduce storage space, in traditional deduplication systems, duplicated data chunks identify and store only one replica of the data in storage. Logical pointers are created for other copies instead of storing redundant data. Deduplication can reduce both storage space and network bandwidth [7]. However such techniques can result with a negative impact on system fault tolerance. Because there are many files that refer to the same data chunk, if it becomes unavailable due to failure can result in reduced reliability. Due to this problem, many approaches and techniques have been proposed that not only provide solutions to achieve storage efficiency, but also to improve its fault tolerance. These techniques provide redundancy of data chunks after performing deduplication.

However, current data deduplication mechanisms in cloud storage are static schemes applied agnostically to all data scenarios. This is a problem as data scenarios exhibit different data characteristics that require different levels of fault-tolerance requirements. For example, data usage in cloud changes overtime; some data chunks may be read frequently in a period of time, but may not be used in another period. Due to the drawback of static schemes, which cannot cope with changing user behavior, deduplication in cloud storages requires a dynamic scheme which has the ability to adapt to various access patterns and changing user behavior in cloud storages.

The contribution of this paper is a dynamic data deduplication scheme for cloud storage, in order to fulfil a balance between storage efficiency and fault tolerance requirements, and also to improve performance in cloud storage systems that experience changes in data scenarios and user patterns. The rest of this paper is organized as follows: section II presents background concepts and related work. Section III demonstrates a proposed system model. Section IV illustrates the simulation of the proposed system model. Section V describes the experimental result. Section VI discusses the future work. Finally section VII concludes this paper.

II. Background and Related Work

A. *Deduplication in Cloud Storages*

Data deduplication is a technique to reduce storage space. By identifying redundant data using hash values to compare data chunks, storing only one copy, and creating logical pointers to other copies instead of storing other actual copies of the redundant data [5], [6]. Deduplication reduces data volume so disk space and network bandwidth can be reduced which reduce costs and energy consumption for running storage systems [7].

Data deduplication can be applied at nearly every point which data is stored or transmitted in cloud storage [7]. Many cloud providers offer disaster recovery [8] and deduplication can be used to make disaster recovery more effective by replicating data after deduplication for speeding up replication time and bandwidth cost savings. Backup and archival storage in clouds can also apply data deduplication in order to reduce physical capacity and network traffic [9], [10]. Moreover, in live migration process, we need to transfer a large volume of duplicated memory image data [11]. There are three major performance metrics of migration to consider: total data transferred, total migration time and service downtime. Longer migration time and downtime would be lead to service failure. Thus, deduplication can assist in migration [12]. Deduplication can be used to reduce storage of active data such as virtual machine images. Factors to consider when using deduplication in primary storage is how to balance the trade-offs between storage space saving and performance impact [13]. Additionally, Mandagere, et al., [13] state that deduplication algorithms reflect the performance of deduplicated storage in terms of fold factor, reconstruction bandwidth, metadata overhead, and resource usage.

B. *Dependability Issues*

When performing deduplication, a portion of data chunks are much more important than others (For example, data chunks that are referenced by many files). Traditional deduplication approaches do not implement redundancy of data chunks. Thus, deduplication may reduce the reliability of the storage system due to the loss of a few important chunks that can lead to the loss of many files. As a result, the critical chunks should be replicated more than the less important data chunks in order to improve reliability of the system. The authors in [14], consider the effects of deduplication on the reliability of the archival system. They proposed an approach to improve reliability by developing a method to weigh and measure the importance of each chunk by examining the number of data files that share the chunk, and use this weight to identify the level of redundancy required for the chunk to guarantee QoS.

C. *Related Work*

Looking at system architectures of existing works of deduplication for cloud backup services such as SAM [10], AA-Dedupe [15], CABdedupe [16], and SHHC [17].

SAM [10] system architecture is composed of three subsystems: File Agent, Master Server and Storage Server. Clients subscribe to backup services, then File Agents are distribute and installed on their machines, while service provider provides Master Server and Storage Server in datacentre to serve the backup requests from clients.

Most of existing solutions that use deduplication technology primarily focus on the reduction of backup time while ignoring the restoration time. The authors proposed CABdedupe [16], a performance booster for both cloud backup and cloud restore operations, which is a middleware that is orthogonal and can be integrated into any existing backup system. CABdedupe consists of CAB-Client and CAB-Server, which is placed on the original client and server modules in existing backup systems.

The main aim of these related works are the following: SAM aims to achieve an optimal trade-off between deduplication efficiency and deduplication overhead, CABdedupe reduces both backup time and restoration time. AA-Dedupe [15] aims to reduce the computational overhead, increase throughput and transfer efficiency, while SHHC [17] tries to improve fingerprint storage and lookup mechanism, however has a concern of scalability. SHHC is a novel Scalable Hybrid Hash Cluster designed for improving response times to fingerprint lookup process. Because of a large number of simultaneous requests are expected in cloud backup services. In order to solve this problem, the hash cluster is designed for high load-balancing, scalability and minimizing the cost for each fingerprint lookup query. The hash cluster is designed as middleware between the clients and the cloud storage. It provides the fingerprint storage and lookup service.

There are other works on deduplication storages which their architectures are designed for scalability issue, for example; Extreme Binning [18], and Droplet [19].

Extreme Binning is used to build a distributed file backup system. The architecture of such system is composed of several backup nodes. Each backup node consists of a compute core and RAM along with a dedicated attached disk. The first task when a file arrives to the system for backup is, it must be chunked. The system can delegate this task to any one of the backup nodes by choosing one according to the system load at that time. After chunking, stateless routing algorithm is used to route the chunked file by using its chunk ID. The chunked file will be routed to a backup node where it will be deduplicated and stored.

Droplet, a distributed deduplication storage system designed for high throughput and scalability. It consists of three components: a single meta server that monitors the entire system status, multiple fingerprinting servers that run deduplication on input data stream, and multiple storage nodes that store fingerprint index and deduplicated data blocks.

Meta server maintains information of fingerprinting and storage servers in the system. When new nodes are added into the system, they need to be registered on the meta server first. The meta server provides a routing service with this information. The client first connects to the meta server and queries for list of fingerprinting servers, and then connects to one of them. After this, a raw data stream containing backup content will be sent to this fingerprinting server, which calculates data block fingerprints and replies results to the client. Fingerprint servers check duplicated fingerprint by querying storage servers.

The nature of data in cloud storage dynamic [20], [21]. For example, data usage in cloud changes overtime, some data chunks may be read frequently in period of time, but may not be used in another time period. Some datasets may be frequently accessed or updated by multiple users at the same time, while others may need the high level of redundancy for reliability requirement. Therefore, it is crucial to support this dynamic feature in cloud storage. However, current approaches are mostly focused on static scheme, which limits their full applicability in dynamic characteristic of data in cloud storage.

III. PROPOSED SYSTEM MODEL

A. *Overall Architecture*

Our system is currently based on client-side deduplication using whole file hashing. Hashing process is performed at the client, and connects to any one of Deduplicators according to their loads at that time. The deduplicator then identifies the duplication by comparing with the existing hash values in Metadata Server. In traditional deduplication systems, if it is a new hash value, it will be recorded in metadata server, and the file will be uploaded to File Servers, its logical path will also recorded in metadata server. If it does exist, the number of references for the file will be increased.

Some systems may keep a number of copies of each file with a static number. However, the files with a large number of references may require more replicas in order to improve availability. To solve this issue, some existing works introduced level of redundancy into deduplication systems. However, identifying level of redundancy by number of references is a poor measurement because files with fewer references may be critical files.

In order to improve availability while maintaining storage efficiency, we propose a deduplication system which considers both the dynamicity and taking Quality of Service (QoS) of the Cloud environment into consideration. In our system model, after identifying the duplication, the Redundancy Manager then calculates an optimal number of copies for the file based on number of references and level of QoS necessary. The numbers of copies are dynamically changed based on the changing number of references, level of QoS and demand for the files. The changes are monitored, for example, when a file is deleted by a user, or the level of QoS of the file has been updated, this will trigger the redundancy manager to re-calculate an optimal number of copies.

Our proposed system model is shown in figure 1. The system is composed of the following components:

- *Load Balancer*: after hashing process with SHA-1, clients send a fingerprint (hash value) to a deduplicator via the load balancer. The load balancer responds to requests from clients sending to any one of deduplicators according to their loads at that time.
- *Deduplicators*: a component designed for identifying the duplication by comparing with the existing hash values stored in metadata server.
- *Cloud Storage*: a *Metadata Server* to store metadata,

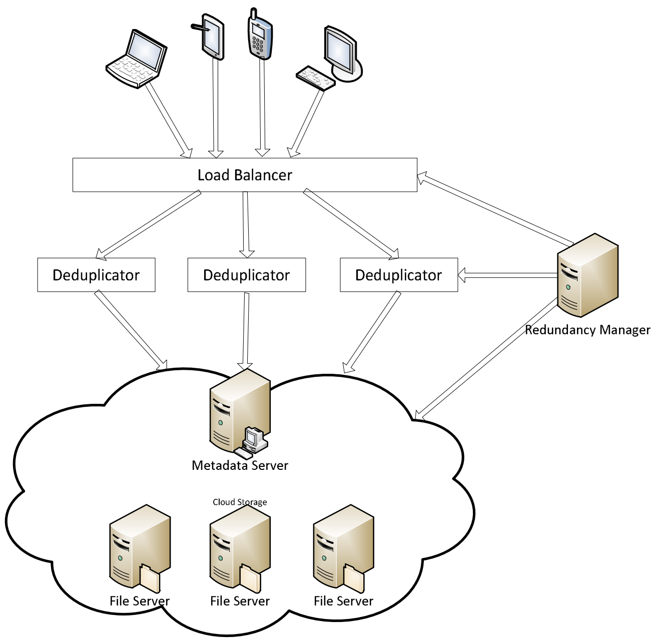


Figure 1: Proposed System Model

and a number of *File Servers* to store actual files and their copies.

- *Redundancy Manager*: a component to identify the initial number of copies, and monitor the changing level of QoS.

IV. Simulation Environment

CloudSim [22] and HDFS Simulator [23] are both Java-based toolkits that have the difference purposes for simulation. CloudSim is used for modelling and simulating of cloud computing environments and infrastructure which is intended to be used for experimenting with various scheduling and allocation algorithms, while HDFS Simulator is a simulation of replication mechanism in Hadoop Distributed File System.

Although CloudSim provides some storage related classes which can be extended, the existing architecture is not yet mature, and requires an additional module which supports simulation of cloud storage in order to evaluate new replication management strategies. HDFS Simulator is more applicable to our work, as HDFS Simulator already provides replication mechanisms, even if the replication degree is a predefined and static value. However, it is possible to modify the source code in order to introduce replication dynamicity. Moreover, we can perform experiments by simulating events like the changing level of QoS.

The mechanism of this work is evaluated through the use of simulation, as it enables researchers an opportunity to simulate large-scale cloud environments, specifically failure events in the cloud as well as assist in evaluation QoS metrics such as availability and performance.

The concepts of HDFS Simulation have been adapted to simulate our proposed system model. We create one Namenode as Metadata server, and five Datanodes as File servers. Metadata in XML format is kept in metadata server. File servers store the copies of files.

There are three events which we simulated: upload, update, and delete. The upload event is when the file is first uploaded to the system. If files already exist in the system, and have been uploaded again, the number of copies of the files will be recalculated according to the highest level of QoS, this is for an update event. For a delete file event, users can delete their files, but the files will not permanently deleted from the system if there are any other users refer to the same files.

A. Upload

Deduplicator calls a hash value of the uploaded file from client, and then checks for any duplicates with the same existing hash value in metadata server. If it is a new file, the new metadata of the file will be added to the system and the file will be uploaded to file server. The replicas of the file will be created according to the level of QoS of the upload file.

B. Update

In the case of existing file, the metadata of the file will be updated and the system may need to create or delete the replicas of the file according to the maximum value of QoS of the file.

C. Delete

The deduplicator checks the number of files which refer to the same hash value user wants to delete. If there is only one reference to the hash, all replicas of the file will be deleted. On the other hand if there are any other files that refer to the hash, only the metadata will be updated, and the number of replicas of the file may need to decrease according to the maximum value of QoS.

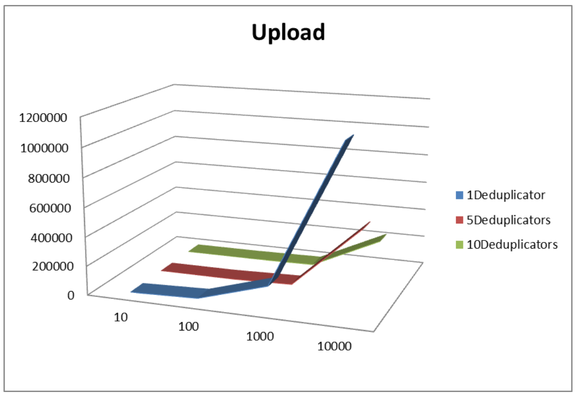
V. Experimental Results

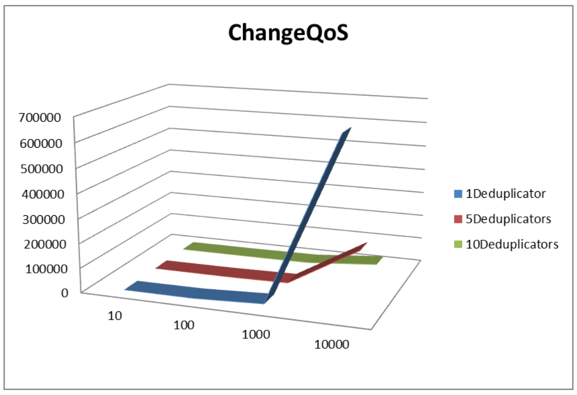
We perform experiments on the simulation of our proposed model. The experiments are performed for one, five, and ten deduplicators.

All the files used in the experiments have been created with stochastic contents and properties. There are various sizes of files used in this experiment: 100 KB, 150 KB, 200 KB, 250 KB, 300 KB, 500 KB, 800, 1 MB, 2 MB. We test upload, update, and delete events on ten files, a hundred files, a thousand files, and ten thousand files. For testing the changing level of QoS, each file has been randomly assigned its level of QoS (1-5). A single QoS value of 1-5 indicates the level of redundancy of each file. Files with higher level of QoS will be replicated more than the lower ones.

When a single deduplicator is used, the system faced scalability problems taking a longer time when the number of files increased as shown in Figure 2. This is because under the heavy load with more requests and more users, a single deduplicator cannot maintain the performance of the system. When the number of deduplicators is increased to five and ten, the results show that it helps to reduce the processing time.

For uploads, when all the files have been uploaded to the system for the first time, comparing the time taken by one deduplicator to five and ten deduplicators. Adding more deduplicators when the number of upload files increase, could help to reduce the processing time. The results show in Table I. When the numbers of upload files are ten and a hundred files, using five deduplicators can reduce 85.75% and 94.20% of the processing time taken by one deduplicator, while ten deduplicators can save more time at 90.85% and 97.55%. When the number of upload files has been increased to a thousand files, five and ten deduplicators can still help to reduce the processing time but they are decreased to 91.40% and 95.58% respectively. However, time saving significantly decrease when the number of upload files are increased to ten thousands as five and ten deduplicators can reduce 60.10% and 79.71% of processing time.





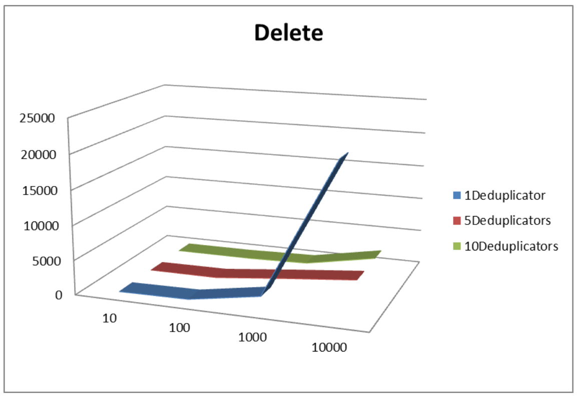


Figure 2: Experimental results

When files have already have been uploaded to the system, we perform experiments for the case when there is a changing level of QoS, which means the number of copies of files in the system could be changed according to the maximum value of QoS. The results of update files show that when the number of files increase, adding more deduplicators can help to reduce the processing time. When the numbers of files are ten, a hundred files, one thousand and ten thousands files, using five deduplicators can reduce 41.78% and 61.79%, 63.78% and 75.25% of the processing time taken by one deduplicator, while ten deduplicators can save more time at 75.02%, 75.34%, 82.09% and 96.17%. We found that, when the numbers of files are ten, one hundred and one thousand, time saving by adding more deduplicators are less than time saving for the upload cases. However, when the numbers of files are increased to one thousand and ten thousands files, the time saving by five and ten deduplicators still increase, in contrast to the upload cases.

We perform experiments to delete files. Adding more deduplicators can also to reduce the processing time, but the results of delete files are slightly different from the upload and update cases. The results show that when the numbers of files are ten, a hundred files, one thousand and ten thousands files, using five deduplicators can reduce 93.42% and 69.31%, 40.74% and 90.28% of the processing time taken by one deduplicator, while ten deduplicators can save more time at 98.68%, 90.59%, 85.87% and 90.03%. We can see that, for the delete case, time saving by adding more deduplicators are decreased when the numbers of files are increased from ten to one hundred and one thousand files. However, when the numbers of files are increased to ten thousands, more deduplicators help to increase time saving.

TABLE I. PERCENTAGE OF TIME SAVING USING FIVE AND TEN DEDUPLICATORS

Number of files	**Upload**		**Update**		**Delete**	
	Five	***Ten***	***Five***	***Ten***	***Five***	***Ten***
10	85.75	90.85	41.78	75.02	93.42	98.68
100	94.20	97.55	61.79	75.34	69.31	90.59
1000	91.40	95.58	63.78	82.09	40.74	85.87
10000	60.10	79.71	75.25	96.17	90.28	90.03

The experimental results are not necessarily surprising. Adding more deduplicators can help to reduce the processing time. However, we still need to find out what is the optimal number of deduplicators to be added into the system according to the events and the number of files at that time. Moreover, the results need to be evaluated against static scheme.

VI. FUTURE WORK

A. *Multiple Metadata Servers*

Currently within the mechanism there is a single metadata server; this could cause scalability problems and also result in a single point of failure. As mentioned in [24], in typical workloads, over 60 percent of the operations performed are metadata operations. As a result, metadata can be replicated in order to improve performance and availability of the overall system. If we change from single to multiple metadata servers, distributed metadata management has to deal with inconsistency among different metadata servers.

B. *Consistency*

Other works not mentioned concerns consistency, which may not be discussed due to the reliance of consistency from their cloud storage provider. As our solution also considers the level of redundancy, so consistency problem could be occurred.

As state in CAP theorem [25], in distributed system “No data store can simultaneously respect the properties of Consistency, Availability, and Partition Tolerance”. We may consider BASE properties in order to achieve eventually consistency and availability.

C. *Monitoring Access pattern*

We also consider the changing of users' demand of files. A component in redundancy manager will monitor file access activities. If users' demand for a particular file is suddenly high, additional copies will be created, and they will be removed when the access rate is back to normal.

D. *Evaluation*

We are planning to evaluate availability and performance of the proposed system. For availability evaluation, using the simulation of failure events with components to monitor and detect the failures. Then we can measure the values of Mean Time to Failure (MTTF) and Mean Time to Repair (MTTR), and use these values to calculate availability.

VII. CONCLUSION

Cloud storage services provided in cloud computing has been increasing in popularity. It offers on demand virtualized storage resources and customers only pay for the space they actually consumed. As the increasing demand and data store in the cloud, data deduplication is one of the techniques used to improve storage efficiency. However, current data deduplication mechanisms in cloud storage are static scheme, which limits their full applicability in dynamic characteristic of data in cloud storage.

In this paper, we propose a dynamic data deduplication scheme for cloud storage, in order to fulfill a balance between changing storage efficiency and fault tolerance requirements, and also to improve performance in cloud storage systems. We dynamically change the number of copies of files according to the changing level of QoS. The experimental results show that, our proposed system is performing well and can handle with scalability problem. We also plan to monitor the changing of users' demand of files. Furthermore, we plan to evaluate availability and performance of the system.

REFERENCES

[1] I. Foster, Z. Yong, I. Raicu, and S. Lu, "Cloud Computing and Grid Computing 360-Degree Compared," in Grid Computing Environments Workshop, 2008. GCE '08, 2008, pp. 1-10.

[2] T. Dillon, W. Chen, and E. Chang, "Cloud Computing: Issues and Challenges," in Advanced Information Networking and Applications (AINA), 2010 24th IEEE International Conference on, 2010, pp. 27-33.

[3] T. G. Peter Mell, "The NIST Definition of Cloud Computing," National Institute of Standards and Technology NIST Special Publication 800-145, September 2011.

[4] SNIA Cloud Storage Initiative, "Implementing, Serving, and Using Cloud Storage," Whitepaper 2010.

[5] D. Harnik, B. Pinkas, and A. Shulman-Peleg, "Side Channels in Cloud Services: Deduplication in Cloud Storage," Security & Privacy, IEEE, vol. 8, pp. 40-47, 2010.

[6] S. Guo-Zi, D. Yu, C. Dan-Wei, and W. Jie, "Data Backup and Recovery Based on Data De-Duplication," in Artificial Intelligence and Computational Intelligence (AICI), 2010 International Conference on, 2010, pp. 379-382.

[7] SNIA, "Advanced Deduplication Concepts," 2011.

[8] V. Javaraiah, "Backup for cloud and disaster recovery for consumers and SMBs," in Advanced Networks and Telecommunication Systems (ANTS), 2011 IEEE 5th International Conference on, 2011, pp. 1-3.

[9] L. L. You, K. T. Pollack, and D. D. E. Long, "Deep Store: An Archival Storage System Architecture," presented at the Proceedings of the 21st International Conference on Data Engineering, 2005.

[10] T. Yujuan, J. Hong, F. Dan, T. Lei, Y. Zhichao, and Z. Guohui, "SAM: A Semantic-Aware Multi-tiered Source De-duplication Framework for Cloud Backup," in Parallel Processing (ICPP), 2010 39th International Conference on, 2010, pp. 614-623.

[11] S. Kumar Bose, S. Brock, R. Skeoch, N. Shaikh, and S. Rao, "Optimizing live migration of virtual machines across wide area networks using integrated replication and scheduling," in Systems Conference (SysCon), 2011 IEEE International, 2011, pp. 97-102.

[12] S. K. Bose, S. Brock, R. Skeoch, and S. Rao, "CloudSpider: Combining Replication with Scheduling for Optimizing Live Migration of Virtual Machines across Wide Area Networks," in Cluster, Cloud and Grid Computing (CCGrid), 2011 11th IEEE/ACM International Symposium on, 2011, pp. 13-22.

[13] N. Mandagere, P. Zhou, M. A. Smith, and S. Uttamchandani, "Demystifying data deduplication," presented at the Proceedings of the ACM/IFIP/USENIX Middleware '08 Conference Companion, Leuven, Belgium, 2008.

[14] D. Bhagwat, K. Pollack, D. D. E. Long, T. Schwarz, E. L. Miller, and J. F. Paris, "Providing High Reliability in a Minimum Redundancy Archival Storage System," in Modeling, Analysis, and Simulation of Computer and Telecommunication Systems, 2006. MASCOTS 2006. 14th IEEE International Symposium on, 2006, pp. 413-421.

[15] F. Yinjin, J. Hong, X. Nong, T. Lei, and L. Fang, "AA-Dedupe: An Application-Aware Source Deduplication Approach for Cloud Backup Services in the Personal Computing Environment," in Cluster Computing (CLUSTER), 2011 IEEE International Conference on, 2011, pp. 112-120.

[16] T. Yujuan, J. Hong, F. Dan, T. Lei, and Y. Zhichao, "CABdedupe: A Causality-Based Deduplication Performance Booster for Cloud Backup Services," in Parallel & Distributed Processing Symposium (IPDPS), 2011 IEEE International, 2011, pp. 1266-1277.

[17] X. Lei, H. Jian, S. Mkandawire, and J. Hong, "SHHC: A Scalable Hybrid Hash Cluster for Cloud Backup Services in Data Centers," in Distributed Computing Systems Workshops (ICDCSW), 2011 31st International Conference on, 2011, pp. 61-65.

[18] D. Bhagwat, K. Eshghi, D. D. E. 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, 2009, pp. 1-9.

[19] Z. Yang, W. Yongwei, and Y. Guangwen, "Droplet: A Distributed Solution of Data Deduplication," in Grid Computing (GRID), 2012 ACM/IEEE 13th International Conference on, 2012, pp. 114-121.

[20] W. Cong, W. Qian, R. Kui, C. Ning, and L. Wenjing, "Toward Secure and Dependable Storage Services in Cloud Computing," Services Computing, IEEE Transactions on, vol. 5, pp. 220-232, 2012.

[21] K. Yang and X. Jia, "An Efficient and Secure Dynamic Auditing Protocol for Data Storage in Cloud Computing," Parallel and Distributed Systems, IEEE Transactions on, vol. PP, pp. 1-1, 2012.

[22] R. N. Calheiros, R. Ranjan, A. Beloglazov, C. A. F. De Rose, and R. Buyya, "CloudSim: a toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms," Software: Practice and Experience, vol. 41, pp. 23-50, 2011.

[23] C. Debains, P. A.-T. Togores, and F. Karakusoglu, "Reliability of Data-Intensive Distributed File System: A Simulation Approach," 2010.

[24] X. Jin, H. Yiming, L. Guojie, T. Rongfeng, and F. Zhihua, "Metadata Distribution and Consistency Techniques for Large-Scale Cluster File Systems," Parallel and Distributed Systems, IEEE Transactions on, vol. 22, pp. 803-816, 2011.

[25] O. Parisot, A. Schlechter, P. Bauler, and F. Feltz, "Flexible Integration of Eventually Consistent Distributed Storage with Strongly Consistent Databases," in Network Cloud Computing and Applications (NCCA), 2012 Second Symposium on, 2012, pp. 65-7