Fault-Tolerant Dynamic Deduplication for Utility Computing

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

***Abstract*— Utility computing is an increasingly important paradigm, whereby computing resources are provided on-demand as utilities. An important component of utility computing is storage; data volumes are growing rapidly, and mechanisms to mitigate this growth need to be developed. Data deduplication is a promising technique for drastically reducing the amount of data stored in such system systems; however, current approaches are static in nature, using an amount of redundancy fixed at design time. This is inappropriate for truly dynamic modern systems. We propose a real-time adaptive deduplication system for Cloud and Utility computing that monitors in real-time for changing system, user, and environmental behaviour in order to fulfill a balance between changing storage efficiency, performance, and fault tolerance requirements. We evaluate our system through simulation, with experimental results showing that our system is both efficient and scalable. We also perform experimentation to evaluate the fault tolerance of the system by measuring Mean Time to Repair (MTTR), and using these values to calculate availability of the system. The results show that higher replication levels result in higher system availability; however, the number of files in the system also effects recovery time. We show that the tradeoff between replication levels and recovery time when the system overloads needs further investigation.**

Keywords*— Utility Computing; Fault-tolerance; Cloud Computing; Cloud storage; Deduplication; Adaptive computing; Dependability;

I. INTRODUCTION

In modern day society, utilities such as water, gas, and electricity are deemed to be requirements for fulfilling routines in daily life. In this context, we define *utility* as an essential service that can be easily obtained by the general population. The idea of computing utility was realised as early as 1966, where it was envisioned that computing networks would mature to reach a point where the idea of 'computer utilities' was made a reality and worked in similar principle to electrical and telephone utilities; able to provision computing service such as computing resources, development platforms or applications to consumers.

The latest paradigm to emerge to realize the vision of utility computing provisioned as a service is *Cloud computing*, featuring loosely coupled systems typically deployed within datacenters capable of dynamically providing computing service and utility to consumers. There is currently no standard definition for Cloud computing, but there are a number of proposed definitions. A popular definition for Cloud computing is taken from the National Institute of Standards and Technology (NIST) [1], which states that Cloud computing is *a model for enabling ubiquitous, convenient, on-demand network access to a shared pool of configurable computing resources that can be rapidly provisioned and released with minimal management effort or service provider interaction*. Furthermore, Cloud computing from an implementation perspective can be defined as a "*parallel and distributed system consisting of interconnected and virtualized computers that are dynamically provisioned and presented as one or more unified computing resources base on service-level agreements established through negotiation between the service provider and consumers*" [2].

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.

One of the services provided by Cloud computing is that of storage – typically this is provided in the form of virtualized storage on demand to customers. This can be provided to meet a variety of needs [4] including archival/backup storage, elastic real-time storage, etc. The amount of storage required by cloud users and applications – and hence ultimately by providers - is rapidly increasing, digital data creation growing by around 50 percent per year. 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. In order to meet these challenges in the face of such demand, providers are increasingly looking for ways to reduce data storage volumes, in order to reduce capital expenditure and energy consumption. The concept of data deduplication is becoming an increasingly popular method for achieving such savings.

Data deduplication is a technique whose objective is to improve storage efficiency. In traditional deduplication systems, incoming data is analysed and compared with data already stored in the system – this can be done at either file-level or block-level (whereby individual "chunks" of data are hashed and compared against a central database). Only unique data is copied into the target system, with logical pointers are created for other copies instead of storing redundant data. Deduplication can reduce both storage space and network bandwidth [5]. However such techniques can result with a negative impact on system fault tolerance; as many files could refer to the same data chunk, any fault pertaining to the chunk (loss of availability, integrity, etc.) will impact all dependent files. Due to this problem, many deduplication-based approaches and techniques have been proposed to not only provide solutions to achieve storage efficiency, but also to improve fault tolerance. These techniques provide redundancy of data chunks after performing deduplication; this ensures that faults affecting a single data chunk will not cause any system failure.

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 and performance 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 periods; similarly, the level of fault-tolerance provided will remain the same regardless of changing QoS requirements. Due to the drawback of such static schemes, which cannot cope with changing user behavior, deduplication in cloud storages requires a dynamic scheme which has the ability to adapt at real-time to changing access patterns and user behavior.

This paper extends the dynamic deduplication scheme for clouds first presented in [6] with a fault-tolerant mechanism for cloud storage, creating a dynamic data deduplication scheme that adapts in real-time to changing requirements for cloud storage, in order to fulfil a balance between storage efficiency and fault tolerance. Furthermore, we also aim 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 our adaptive system model; section IV illustrates our simulation of the proposed system model; Section V describes the results of our experimentation. Section VI discusses the future work, and section VII concludes the paper.

II. Background and Related Work

A. Deduplication in Cloud Storages

Data deduplication is a technique that aims to reduce the amount of space used for data storage. This is achieved by identifying redundant data using hash values to compare data chunks, storing only a fixed number of copies of these chunks (typically one), and creating logical pointers to these copies instead of storing the redundant data [7], [8]. By employing deduplication, data volume is reduced and hence disk space and network bandwidth can be reduced, resulting in reduced costs and energy consumption for running storage systems [5].

Data deduplication can be applied at nearly every point which data is stored or transmitted in cloud storage [5]. Many cloud providers offer disaster recovery [9] 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 [10], [11]. Moreover, in the live migration process, there is a need to transfer a large volume of duplicated memory image data [12]. 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 – therefore, deduplication can assist in migration [13]. 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 [14]. Additionally, Mandagere, et al., [14] state that deduplication algorithms reflect the performance of deduplicated storage in terms of fold factor, reconstruction bandwidth, metadata overhead, and resource usage. However, the mechanisms employed in this work are static, and do not consider the need of Cloud-based systems to adapt in real-time to changing behaviours and circumstances, in particular with regard to Quality of Service.

B. Quality of Service

In the context of cloud storage services, QoS properties refer to non-functional aspects or quality aspects of the services, such as performance, reliability, scalability, availability and security, which could be used as a differentiating point in the preference of customers [15]. QoS is typically defined in a Service-Level Agreement, and the QoS aspects that we will take into consideration in our work are *availability* and *performance*.

A *Service Level Agreement* is part of the contract between the service consumer and service provider and formally defines the level of service [16], whilst *availability* is one of the attributes of dependable system [17], and also a quality aspect of a service. Availability is probability that the system is ready to be used. The service should be available when it is invoked. *Performance* is one of the quality aspects of a service which represent the speed in which a request can be completed Performance can be measured in terms of throughput, response time, execution time, latency and transaction time. Initially we will measure the response time, which is the time that elapses from when a node sends a request for a file until it receives the complete file [18].

C. *Dependability Issues*

When performing deduplication, a portion of data chunks may be much more important than others (for example, data chunks that are referenced by many files may have more business value than those referenced by few files). Traditional deduplication approaches do not implement redundancy of data chunks; this means that deduplication may reduce the reliability of a storage system, as the loss of a few important chunks can lead to a loss of integrity in many files. As a result, critical chunks should be replicated more than the less important data chunks in order to improve reliability of the system. The authors in [19], 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. However, this again is a static approach, with no capability to adapt in real-time to changes in the system environment.

D. *Related Work*

We now examine the existing work performed on system architectures of deduplication for cloud backup services such as SAM [11], AA-Dedupe [20], CABdedupe [21], and SHHC [22].

The *SAM* [11] system architecture is composed of three subsystems: File Agent, Master Server and Storage Server. Clients subscribe to backup services, then File Agents are distributed and installed on their machines, while the service provider provides the Master Server and Storage Server in its data-centre to serve backup requests from clients.

Most of the existing solutions that use deduplication technology primarily focus on the reduction of backup time while ignoring restoration time. The authors of CABdedupe [21] propose a performance booster for both cloud backup and cloud restore operations, through 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 [20] aims to reduce the computational overhead, increase throughput and transfer efficiency, while SHHC [22] 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 whose architectures are designed for the scalability issue, for example; Extreme Binning [23], and Droplet [24].

Extreme Binning is used to build a distributed file backup system. The architecture of such a 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 is 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. The 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 limitation of all of these mechanisms when applied to the Cloud computing paradigm is that the nature of data in cloud storage is dynamic [25], [26]. 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. In order to adequately address the needs of Cloud users and providers, it is crucial to support this dynamicity; this requires an architecture and mechanism for monitoring the real-time behaviours of the Cloud system and adapting dynamically to changing needs. The following section introduces the system model for our proposed dynamic Cloud deduplication system.

III. SYSTEM MODEL FOR ADAPTIVE DEDUPLICATION

A. *Overall Architecture*

The work presented in this paper builds on the dynamic deduplication system first presented in [6]. The system model for this work is shown in figure 1, and features a client-side deduplication using whole file hashing. The hashing process of our scheme is performed on the client-side, and connects to one of a number of *Deduplicator* components, based on current system load (the number of these components is arbitrary and scaleable). The deduplicator component used then identifies duplication in the client's file by calculating a hash of the file and comparing this with existing hash values stored in the *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 the hash value is already contained in the database, the number of references for the file will be increased.

Some systems may keep a static number of copies of each file (or data chunk); however, as the system environment changes – for example, data chunks that are referenced by a large number of files - more replicas may be necessary to avoid late timing faults and thus ensure appropriate QoS is maintained. To solve this issue, some existing works introduce a level of redundancy into deduplication systems - however, in addition to their lack of dynamism, identifying a correct level of redundancy by the number of references to a data chunk is a poor measurement because files with fewer references may be critical files.

In our system model, after identifying the duplication, the *Redundancy Manager* then calculates an optimal number of copies for the file based on its number of references and level of QoS necessary, as associated with the policy defined in the *SLA Manager*.

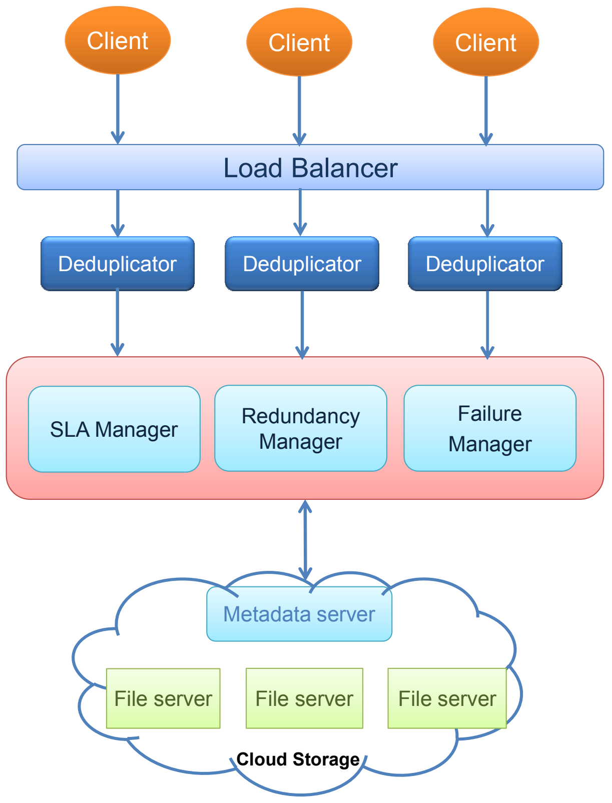

Figure 1: The System Model for our Adaptive Real-Time Deduplicator for Cloud Systems

The numbers of copies of deduplicated data are dynamically changed at run-time 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.

In order to address fault-tolerance issue, apply the re-replication function from HDFS simulator [27] to *Failure Manager*, the new component in our simulation system. This function provides failure monitoring and detection tools including failure recovery process. Our proposed system model as shown in figure 1 is further composed of:

- *Load Balancer*: after hashing the 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 the 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, and a number of *File Servers* to store actual files and their copies.
- *Redundancy Manager*: a component to identify the initial number of copies according to the level of QoS and the policy defined in SLA Manager, and monitor the changing level of QoS.
- *SLA Manager*: a component defines the level of service policy.
- *Failure Manager*: a component to monitor and detect failure and recover failure.

IV. Simulation Environment

In order to test the effectiveness of this scheme, we conducted a number of simulation-based experiments. CloudSim [28] and HDFS Simulator [27] are both Java-based toolkits that have different purposes for simulation. *CloudSim* is used for modelling and simulating cloud computing environments and infrastructure, and is intended to be used for experimenting with various scheduling and allocation algorithms, while *HDFS Simulator* is a simulation of the replication mechanism in the Hadoop Distributed File System, popularly used in Cloud Storage systems.

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. Therefore, HDFS Simulator is more applicable to our work, as it already provides replication mechanisms. Although the replication degree of these mechanisms is pre-defined to be a static value, it is possible modify the source code in order to introduce replication dynamicity. Moreover, we can perform experiments by simulating events like changing the level of Quality of Service. 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 the metadata server, and File servers store the copies of files.

We simulate three file system events: 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. All three events cause the system to monitor the current set of QoS requirements in the system, and cause the system to adapt in real-time to the changing requirements.

A. Upload

When uploading files, the deduplicator requests a hash value of the uploaded file from a client, and then checks for any duplicates with the same existing hash value in the 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 the 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 an existing file, the metadata of the file will be updated to reflect a new reference being added, 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 the 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.

In order to evaluate availability of our proposed system, we also simulated failure events by applying an experiment from the HDFS simulator [27]. We generated a simulation file to be used as a failure scheduler; the failure scheduler defined the scheduled failing times. Mean Time to Failure (MTTF) was chosen to be 3 seconds. Then we used this failure scheduler to simulate failure events to our system. These events are monitored, detected, and recovered by the Failure Manager component. From the simulation of failure monitoring, detection, and recovery, we can measure the values of Mean Time to Repair, and use these values to calculate availability according to the formula [29] presented in (1):

$$\text{Availability} = \text{MTTF} / (\text{MTTF}+\text{MTTR}) \qquad (1)$$

V. Experimental Results

Based on the discussed events, we performed a number of experiments on the simulation of our proposed model. The experiments were performed using one, five, and ten deduplicators respectively.

All the files used in the experiments were created with stochastic contents and properties. There are various sizes of

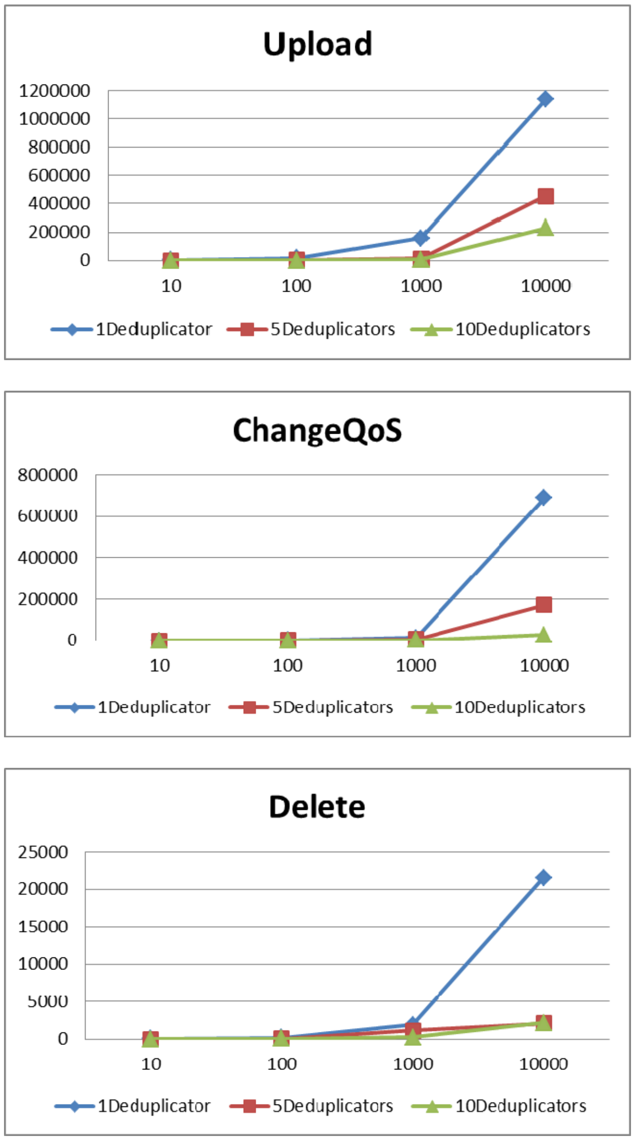

Figure 2: Experimental results

files used in this experiment: 100 KB, 150 KB, 200 KB, 250 KB, 300 KB, 500 KB, 800, 1 MB, 2 MB. We tested upload, update, and delete events on ten files, a hundred files, a thousand files, and ten thousand files respectively. For testing the changing level of QoS, each file was randomly assigned a level of QoS between 1 and 5; this mapped to the required level of redundancy of each file. Files with higher level of QoS will be replicated more than the lower ones.

When a single deduplicator was 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, we compared the time taken from 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.

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 SAVED 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.

We also perform experiments on the fault tolerance mechanism, which is the extended component in our proposed system. The experiment was conducted by scheduling failure times into our system as described in section IV, with the simulated failure events being monitored by the Failure Manager. When failure events are detected, the files in the failing node are be re-replicated to another alive node. The value of the Mean Time to Repair (MTTR) are measured and we used these values to calculate availability. We carried out 50 experiments for each set of values, and then get the mean of those results. The replication level went from 3 to 10. We plot the results from replica=3 to replica=10 when the number of files from 10,000, 20,000, 50,000 and 100,000 files on Figure 3.

From the results we can see that, the availability of the system getting higher when the replication level increased. This is because, if a failure occurs in a node, there are still more replicas in other nodes which are available. However when the replication level reach 8, 9, 10, the difference in availability are not significant. We may need to find out the balance between the replication level and the availability.

When the number of files increases to 50,000 and

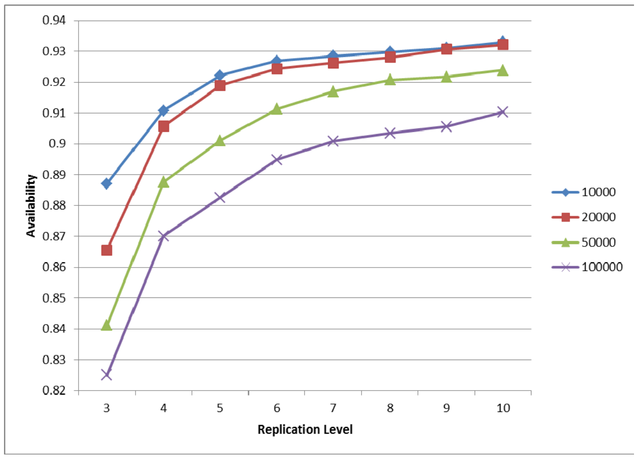

Figure 3: Availability results related to number of replicas under simulated failure events

100,000, availability values of the same replication level significantly decreased. This is because as the number of files in a node increase, the number of replicas also increase. Then when a node fails, there are more files which need to be re-replicated. This is considered as a limitation of the simulation system when it reaches a great load in the system. This also indicated that we still need to improve our system and simulation environment. Moreover, the tradeoff between replication level and availability need to be compared.

VI. Future Work

A. Performance Evaluation

For future work, we plan to evaluate the performance of the proposed system. Performance is one of the quality aspects of a service which represent the speed in which a request can be completed Performance can be measured in terms of throughput, response time, execution time, latency and transaction time. Initially we plan to measure the response time, which is the time that elapses from when a node sends a request for a file until it receives the complete file. This evaluation could be conducted by simulating file request and transfer events, and measuring the response time when demand of files is changing. 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 are suddenly high, additional copies will be created dynamically at real-time, and they will be removed when the access rate is back to normal. We can then calculate the average of response times for the length of the simulation.

B. Overall Evaluation

The main aim of our research is to improve availability and performance in cloud storage systems through dynamic deduplication, and to archive a good balance between storage efficiency and fault tolerance requirements. In order to archive this aim we need to balance trade-off between availability, performance and storage efficiency. Moreover, our proposed system must be evaluated against an existing static deduplication scheme, and it must ensure that the benefit of dynamic deduplication and redundancy is higher than the cost of them.

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 fault-tolerant 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 when adding more deduplicators and can handle with scalability problem.

We also perform the experiment in order to evaluate fault tolerance of the system by measuring Mean Time to Repair (MTTR), and using these values to calculate availability of the system. The results show that higher replication level makes the system higher availability. However the number of files in the system effect the recovery time, which we need to figure out the tradeoff between replication level and recovery time when the system overloads.

We still need to improve our system to achieve lower recovery time. Because if we try to reduce recovery time or Mean Time to Repair (MTTR) to be close to zero, then our system can achieve a higher availability. We also plan to monitor the changing of users' demand of files. Furthermore, we plan to evaluate performance of the system and evaluate our system against a static deduplication system.

References

[1] R. Buyya, C. S. Yeo, and S.r Venugopal, "Market-oriented cloud computing: Vision,hype, and reality for delivering it services as computing utilities."

[2] N. Leavitt "Is Cloud Really Ready for Prime Time?" Vol 42, 2009

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

[6] W. Leesakul, P. Townend, and J. Xu, "Dynamic Data Deduplication in Cloud Storage," in *Service Oriented System Engineering (SOSE), 2014 IEEE 8th International Symposium on*, 2014, pp. 320-325.

[7] 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.

[8] 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.

[9] 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.

[10] 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.

[11] 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.

[12] 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.

[13] 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.

[14] 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.

[15] Q. Yu, X. Liu, A. Bouguettaya, and B. Medjahed, "Deploying and managing Web services: issues, solutions, and directions," *The VLDB Journal,* vol. 17, pp. 537-572, 2008.

[16] B. Philip, L. Grace, and M. Paulo, "Service Level Agreements in Service-Oriented Architecture Environments," *Software Engineering Institute, Carnegie Mellon University,* vol. CMU/SEI-2008-TN-021, 2008.

[17] A. Avizienis, J. C. Laprie, B. Randell, and C. Landwehr, "Basic concepts and taxonomy of dependable and secure computing," *Dependable and Secure Computing, IEEE Transactions on,* vol. 1, pp. 11-33, 2004.

[18] Y. Dai, L. Yang, and B. Zhang, "QoS-driven self-healing web service composition based on performance prediction," *J. Comput. Sci. Technol.,* vol. 24, pp. 250-261, 2009.

[19] 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.

[20] 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.

[21] 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.

[22] 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.

[23] 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.

[24] 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.

[25] 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.

[26] Y. Kan and J. Xiaohua, "An Efficient and Secure Dynamic Auditing Protocol for Data Storage in Cloud Computing," *Parallel and Distributed Systems, IEEE Transactions on,* vol. 24, pp. 1717-1726, 2013.

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

[28] 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.

[29] G. Candea, A. B. Brown, A. Fox, and D. Patterson, "Recovery-oriented computing: building multitier dependability," *Computer,* vol. 37, pp. 60-67, 2004.