Lightweight Cloud Storage Systems: Analysis and Performance Evaluation

Samson Akintoye and Antoine Bagula
ISAT Laboratory, Department of Computer Science,
University of the Western Cape, Bellville, South Africa

***Abstract*—The increasing popularity of cloud storage has led many organizations that dealing with critical data (financial, medical, telecoms data) to consider moving data out of their data centers into the cloud storage. However, using a single cloud storage provider raises concern such as having a single point of failure. Furthermore, despite their popularity, vendor locks-ins cloud solutions are not a natural fit-for-all as they are plugged with issues related to electricity consumption and security and raise privacy concerns. Striping data across multiple clouds storage to provide fault tolerance and deploying lightweight cloud solutions to avoid vendor lock-in are the solutions to these problems. In this paper, we survey different multiple clouds storage systems: how they are designed and implemented; their strengths and weaknesses. We also conduct an analysis of different redundancy schemes and compare emulation and experimentation to assess the relevance of using emulation when cost and availability do not allow prototyping.**

Cloud Computing Cloud Storage, Regenerating codes, Erasure Codes, Replication

I. Introduction

Cloud computing is a model for hosting and delivering IT services over the internet. It provides users with a long list of benefits, such as on-demand self-service; broad, heterogeneous network access; resource pooling and rapid elasticity with measured services [20]. One of the important services offered in cloud computing is the cloud data storage, in which, subscribers do not have to store their own data on their servers, where instead their data will be stored on the cloud service provider's servers [29]. Cloud Storage is a service where data is remotely maintained, managed, and backed up. The service is available to users over the internet. It allows the user to store files online so that the user can access them from any location via the internet. The provider company makes them available to the user online by keeping the uploaded files on an external server. However, using a single cloud storage provider as a backup solution raises concerns such as having a single point of failure [4] and vendor lock-ins [11]. The alternative solution is to stripe data across different cloud providers [11], [10] and thus improve the fault-tolerance of cloud storage through diversity.

Data can be striped to multiple clouds with replication, erasure codes and generating codes to provide high availability from non-reliable devices. Replication is easy to understand and implement but far from being optimal with respect to the trade-off between storage and availability [19], [24] since it requires extremely high bandwidth and storage overhead. Erasure codes eliminate the overhead of strict replication. It divides an object into n fragments and recode them into m fragments, where $n > m$. The key property of erasure codes is that the original object can be reconstructed from any m fragments. Erasure codes performs well when some clouds experience short-term transient failures or foreseeable permanent failures but there some cases where permanent failures do occur and are not always foreseeable [22]. When a cloud fails permanently, it is replaced by a new, empty cloud. This replacement cloud is required to obtain the data that was stored previously in the failed cloud, by downloading data from the remaining clouds in the network. However, downloading the entire data is clearly wasteful of the network resources. It is important to reduce the repair traffic (i.e., the amount of data being transferred over the network during repair), and hence the monetary cost due to data migration [26]. To minimize repair traffic, regenerating codes [16] have been proposed for storing data redundantly in a distributed storage system (a collection of interconnected storage nodes). Regenerating codes are built on the concept of network coding [3], in the sense that nodes perform encoding operations and send encoded data. It requires less repair traffic than traditional erasure codes with the same fault tolerance level.

This paper revisits the issue of fault-tolerance in cloud computing to evaluate the performance of the different cloud storage systems described above when deployed on commodity hardware used in emerging lightweight community cloud infrastructures. The main contributions of this paper are threefold. Firstly, we review different multiple clouds storage systems and compare these systems in an emulated environment in terms of data integrity, confidentiality and availability. Secondly, building under the OpenStack platform , we conduct testbed experimentation using the same performance parameters to compare the same cloud storage algorithms on a cloud storage prototype built with commodity desktop hardware. Finally, we compare the emulated and experimental results to assess the relevance of using emulation when testbed experimentation is not an option because of cost acquisition and availability of equipment. The remainder of this paper is organized as follows. Section 2 presents data redundancy schemes while section 3 describes the key features of the studied multiple cloud storage systems. These systems are then analyzed in in section 4 and their performance evaluated in section 5. Our conclusions and avenues for future work are discussed in section 6.

II. Data redundancy schemes

Data redundancy is a technique of storing multiple instances of the same data on different cloud storage in order to avoid data unavailability such that if any part of the stored data is unavailable as a result of permanent failure in any of the cloud storage , the surviving clouds should be enough to reconstruct the original data. In this section, we review different schemes used to add redundancy to data in multiple cloud storage.

A. Replication.

Replication is the simplest and most commonly used form of striping data across the clouds storage to improve reliability against cloud failures [24], [17]. There are two type of replication schemes: whole-file and block level replication schemes.

1) Whole-file replication:: Suppose we make *n* copies of the file stored across many cloud storages. We need at least one of those *n* clouds storage to be available to recover the file in case of failure.

2) Block-level replication:: Let f be an input file, n be the total number of cloud storages, and k be the number of cloud storages required for data reconstruction. The input file f is divided into n chunks f.chunk0, ... , f.chunk$n$ such that each cloud storage will hold exactly one chunk.

Many distributed storage systems adopted replication as a redundancy scheme. For example PAST [18] adopts full-file replication, while CFS [15] divides files into chunks and then performs replication at the chunk level.

B. Erasure codes.

Erasure codes provide a better storage solution and ensure high data availability with less storage than replication. There are two types of erasure codes: a) *RAID-6 codes:* This is based on XOR operations and b) *Reed-Solomon (RS) codes:* This is based on linear operations on Galois Fields. In the latter, an input file f is encoded using a Vandermonde matrix to give $(n-k)$ parity chunks f.chunk$k$, ... , f.chunk$n$, where n is the total number of cloud storages and k is the number of cloud storages required for data reconstruction. The file f is split into k chunks f.chunk0 , ... , f.chunk$k-1$. Each node will hold exactly one chunk. Decoding is done by decoding the k chunks downloaded from the first k accessible cloud storage. Repairs are simply decodes (if data chunks are missing) followed by encodes (if code chunks are missing).

C. Regenerating Codes.

Regenerating codes are a class of erasure codes that optimally trade the bandwidth needed for repairing a failed node with the amount of data stored per node of the network. Regenerating codes apply network coding to storage systems to offer the best trade-off between network bandwidth repair cost C and storage cost M. The parameter set of regenerating codes over finite field F_q is given by $\{n, k, d, \alpha, \beta, B\}$. The corresponding codes are called $[n, k, d]$ regenerating codes having a parameter set (α, β, B). Two important operations of $[n, k, d]$ regenerating codes are as follows.

Data Reconstruction: Reconstruct the message of size B symbols by downloading $B = k\alpha$ symbols from any k nodes

Regeneration (repair): Repair a failed node of size α symbols by downloading $\gamma = d\beta$ symbols from any d nodes among $n-1$ remaining nodes

There are two classes of regenerating codes: Exact minimum-storage regenerating (EMSR) codes [28] and Functional minimum-storage regenerating (FMSR) codes [14]

1) Exact minimum-storage regenerating (EMSR) codes: In EMSR, a replaced node is required to store exactly the same data as was stored in the failed node. As a result of that, there is no change in the coefficients of a replace node under Exact minimum-storage regenerating codes. This eliminate additional communication overheads during the regeneration operation, and also avoids retuning of the reconstruction and regeneration operations. It has capability to maintain the code in systematic form [23], [25].

2) Functional minimum-storage regenerating (FMSR) codes: The notion of functional regeneration was introduced in [16] and implemented to repair a single cloud failure in [14]. The input file f is encoded to give $2n$ code chunks f.chunk0, ... , f.chunk$2n$. Each cloud storage will hold exactly two chunks. Decoding is done by decoding the $2k$ chunks downloaded from the first k accessible cloud storage (tolerates up to two failed cloud storage). Despite tolerating two failed nodes in decode, only single-cloud storage failures are supported in repair, which will download exactly one chunk each from the remaining $n-1$ accessible cloud storage. In the functional regeneration, the data stored at the replacement cloud storage may be different from the data stored in the corresponding failed cloud storage. This difference may incur the additional communication to inform all cloud storage of the replacement. Moreover, the reconstruction and regeneration need to be retuned for the new set of coefficients.

III. Multiple Cloud Storage Systems

In this section, we survey and review some proposed multiple clouds storage systems.

A. HAIL (High-Availability and Integrity Layer).

Bowers et al. [11] propose HAIL (High-Availability and Integrity Layer) that provide data integrity and availability across multiple cloud storage. It engages the use of PORs (Proofs of Retrievability) as building blocks by which storage resources can be verified and reallocated when failures are detected. It relies on a single trusted verifier that interacts with nodes to verify the integrity of stored data. In HAIL, a client distributes a file F with redundancy across n servers and keeps some small (constant) state locally.The goal of HAIL is to ensure resilience against a mobile adversary. The advantages of the protocol are: Strong file-intactness assurance; Low overhead; Strong adversarial model; Direct client-server communication; and Static/dynamic file protection. However, the protocol is built on erasure code which provides only fault tolerance; it does not address the recovery of the outsourced data on a failed cloud. It requires that the servers run some code and does not provide guarantee of confidentiality of the stored data.

B. RACS (Redundant Array of Cloud Storage).

Abu-Libdeh et al. [2] Propose RACS (Redundant Array of Cloud Storage) a proxy based system that transparently stripes data across multiple cloud storage providers. The protocol allows customers to avoid vendor lock-in, reduce the cost of switching providers, and better tolerate provider failures. It retrieves data from the cloud that is about to fail and moves the data to the new cloud. It is designed as a proxy-based solution between the client application and a set of n repositories, which are cloud storage locations ideally hosted by different providers. RACS splits the incoming object into m data shares of equal size where $m < n$ are configurable parameters. It uses erasure coding to create additional $(n - m)$ redundant shares, for a total of n shares. Any subset of m shares is sufficient to reconstruct the original object. The disadvantages of the protocol are: it does not have capability of recover loss data when permanent cloud failure occurred and address data integrity and confidentiality challenges of cloud storage.

C. DEPSKY.

Bessani et al. [10] propose DEPSKY, a dependable and secure storage system to improve the availability, integrity and confidentiality of data stored in the cloud. It addresses four limitations of individual clouds storage: Loss of availability, Loss and corruption of data, Loss of privacy and Vendor lock-in by using Byzantine quorum system protocol, cryptography, secret sharing, erasure codes and the diversity that comes from using several clouds. DEPSKY is designed as a virtual storage cloud, which is accessed by its users by invoking operations in several individual clouds.

D. NCCloud.

Chen et al. [14] propose NCCloud, a proxy-based storage system for fault-tolerant multiple-cloud storage that achieves cost-effective repair for a permanent single-cloud failure. It built on top of a network-coding-based storage scheme called the functional minimum-storage regenerating (FMSR) codes, which maintain the same fault tolerance and data redundancy as in traditional erasure codes (e.g., RAID-6), but use less repair traffic which incur less monetary cost due to data transfer. NCCloud is a proxy based design that interconnects multiple cloud repositories as shown in Figure 1a. The proxy serves as an interface between client applications and the clouds. If a cloud experiences a permanent failure, the proxy activates the repair operation, as shown in Figure 1b.
The proxy reads the essential data chunks from other surviving clouds, reconstructs new data chucks, and writes these new chucks to a new cloud. It excludes the failed cloud in repair operation. However, the protocol does not guarantee the integrity and confidentiality of the data chunks stripped across multiple clouds.

E. ICStore (Intercloud Storage.)

Cachin et al. [12] address and improves the confidentiality, Integrity, Reliability and Consistency of data in multiple cloud storage providers. The solution distribute data to the intercloud after confidentiality and integrity test. The ICStore consist of ICStore client which coordinate multiple cloud storage services of the intercloud and ICStore client consist of three core layers: (i) Confidentiality, (ii) Integrity and (iii) Reliability and Consistency. The layered approach enables layers to be switched ”on” and ”off” to provide different levels of dependability. The major advantage of the solution is to protect outsourced data against downtime, loss or hacker attacks.

F. Scalia

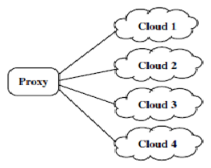
Papaioannou et al. [21] propose a cloud storage brokerage solution that continuously adapts the placement of data across multiple cloud storage providers with the optimization objective to minimize storage cost. It efficiently considers repositioning of only selected objects that may significantly lower the storage cost. Scalia can run directly at customer sites as an integrated hardware and software solution or can be deployed as hosted service across several datacenters. The advantage of the solution is that, it help cloud customer to avoid vendor lock-in and satisfy certain availability and durability constraints in a cost-effective way. However, the solution did not address latency overhead and scalability of prototype

G. Toward Secure and Dependable Storage Services in Cloud Computing

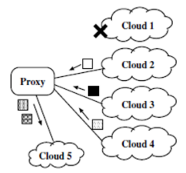
In the cloud storage system, users outsource their data in the cloud without burden of managing local hardwares and softwares. Thus, the integrity and availability of the data being stored on the distributed cloud servers must be guaranteed. One of the challenges is to detect unauthorized data modification and corruption as a result of server compromise and random Byzantine failures. To solve these problems, the Wang et al. [30] review basic tools from coding theory that is needed for data distribution across cloud servers and the homomorphic token is introduced to provide data integrity. The token computation function is selected to preserve the homomorphic properties, which can be perfectly integrated with the verification of erasure-coded data. Finally, the scheme is extended to third party auditing with only slight modification of the main design

H. A secured cost-effective multi-cloud storage in cloud computing

In order to secure, storage and access on outsource data in the cloud, Singh et al. [27] propose technique of multiple division to protect the client’s data. The scheme divides the data into multiple parts and stored in an redundant form across the cloud service providers simultaneously. The advantage of the scheme is that, as the number of part increase the security of data also increase because it is difficult for intruder to check all file to match the content.



(a) Normal Operation



(b) Repair Operation

Fig. 1. Normal/Repair Operations [14].

IV. Multiple Cloud Storage Systems Analysis

In this section, We analyse and compare multiple cloud storage systems as depicted in Table 1.

A. Data Availability

All schemes reviewed facilitate the availability of data stored in multiple cloud storages. Different redundancy schemes are used by different systems to stripe data across multiple clouds storage providers for fault-tolerance and data redundancy. The diversity of multiple clouds improves the availability of data stored in the cloud storage.

B. Integrity of Data

[11],[12], [30], [27] and [10] use cryptographic protocols to protect the integrity of the stored data against malicious and mobile adversaries. However, RACS, Scalia and NCCloud proposed in [2], [21] and [14] do not provide any mechanism to detect data corruption violations in the cloud storage.

C. Data Confidentiality

The schemes [10], [12] and [21] address the security of data stored in the cloud storage. They use cryptographic protocol to secure access to the outsourced data in the cloud storage.

D. Repairing of a failed cloud

Only NCCloud proposed in [14] addresses a single cloud permanent failure and repair the lost data with the help of the other surviving clouds to preserve data redundancy. Other schemes do not provide any mechanism to repair the lost data of a failed cloud rather they retrieves data from the cloud that is about to fail and moves the data to the new cloud.

E. Redundancy Schemes

[11], [21], [2], [30] and [10] use erasure codes and [12] and [27] use replication technique to provide fault tolerance while NCCloud proposed in [14] is built on regenerating codes to provide both fault tolerance and storage repair. It is established that regenerating codes require less repair traffic (i.e., the amount of data being transferred over the network during repair) than traditional erasure codes with the same fault tolerance level [13].

V. Performance Evaluation

In this section, we emulate and implement NCCloud [14]. We also compare the results of emulation and implementation.

A. Emulation

NCCloud is mounted on a linux machine with an Inter(R) Core(TM) i5-4590 3.30Ghz CPU and 8GB RAM. We create five folders which represent cloud storage, one of them is a spare cloud storage used in repair. We carry out two emulations on the local cloud storage created. The first emulation compares Replication, RS and FMSR codes when n = 4 and k = 2 with varying file sizes, where n = number of nodes and k = fault tolerant. The second emulation compares Replication, RS and FMSR codes under different values for n and k with a fixed file size.

In the first emulation, We test the response times of the file upload, file download, and repair operations of Replication, RS and FMSR with n = 4 and k = 2. We use randomly generated files from 10MB to 50MB as the data set. We set the path of a chosen repository to a non-existent location to simulate a cloud storage failure in repair. Figure 2, plots the response times of all three file operations versus the file size and show that RS has less response time than FMSR codes and Replication has highest response time in file upload. For file download, RS has lowest response time follow by Replication codes, FMSR has highest response time. For instance in the case of n = 4 and k = 2, when uploading a 50MB file, RS takes 0.063s, FMSR codes take 0.078s and Replication takes 0.136s to upload. When downloading 50MB file, RS takes 0.402s and Replication takes 0.725s to download while FMSR takes 0.459s to download.

In the second emulation, we fix the file size at 1.2GB and test the response times of the three operations again under four different sets of configurations for n and k: n = 4, k = 2; n = 6, k = 4; n = 8, k = 6; and n = 10, k = 8. Figures 3, 4, 5 and 6 shows the response time results, RS codes have less response time than FMSR codes and Replication has highest response time in file upload. For file download, Replication has lowest response time follow by RS codes, FMSR has highest response time regardless of n and k. For instance in the case of n = 4 and k = 2, when uploading a 1.2GB file, RS codes take 17.3s, FMSR codes take 20.7s and Replication takes 36.0s to upload and encode. When downloading a 1.2GB file, RS codes take 10.1s and Replication takes 6.8s to download while FMSR codes take 9.3s to download and decode, no decoding

TABLE I
COMPARISON OF THE MULTIPLE CLOUDS STORAGE SYSTEMS.

Schemes	Availability	Integrity	Confidentiality	Repairing of a failed cloud	Redundancy Scheme
Bowers et al. [11]	Yes	Yes	No	No	Erasure Codes
Abu-Libdeh et al. [2]	Yes	No	No	No	Erasure Codes
Bessani et al. [10]	Yes	Yes	Yes	No	Erasure Code
Henry et al. [14]	Yes	No	No	Yes	Regenerating codes
Cachin et al. [12]	Yes	Yes	Yes	No	Replication
Papaioannou et al. [21]	Yes	No	No	No	Erasure Code
Wang et al. [30]	Yes	Yes	No	No	Erasure Code
Singh et al. [27]	Yes	Yes	Yes	No	Replication

is needed in the case of RS codes and Replication as the native chunks are available. The differences increase with n and k.

However, FMSR has less response time in repairing a single cloud failure. The main advantage of FMSR codes is that FMSR codes download less data during repair. For instance, in repairing a 50MB file with n = 4 and k = 2, FMSR codes takes 0.139s, RS codes takes 0.255s and Replication takes 0.726s.

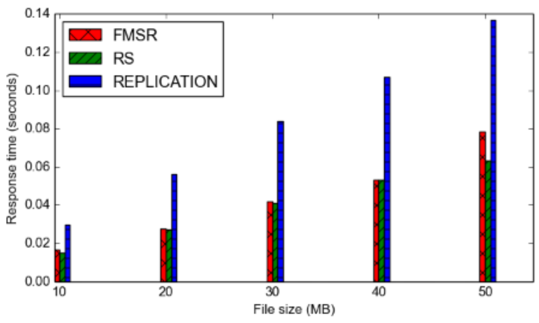
B. Experimentation

Multiple cloud storage is implemented by integrating NC-Cloud on openstack swift [1], openstack is an open-source software for cloud computing platform, usually deployed as an infrastructure-as-a-service (IaaS). We connect five storage nodes to proxy node via private network and proxy node connects to the internet as shown in figure 7. Proxy node runs the following services:

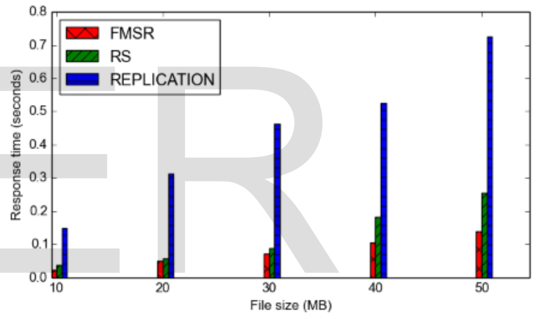
- Proxy services : This receive and process HTTP requests from NCCloud
- Keystone: Provides an authentication and authorization service for OpenStack services
- Mysql database : Provides scalable and reliable Cloud Database-as-a-Service functionality

Storage nodes runs account, container, and object services. Also contains the SQLite databases. We create $(n + 1)$ containers on Swift, so each container represent a cloud repository, one of them is a spare cloud repository used in repair.

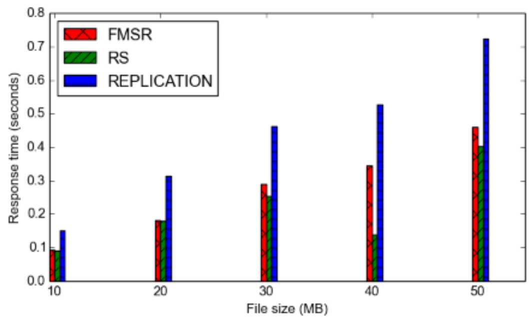
We carry out experiments to compare Replication, RS and FMSR codes when n = 4 and k = 2 with varying file sizes where n = number of nodes and k = fault tolerant. we test the response times of the file upload, file download, and repair operations of Replication, RS and FMSR with n = 4 and k = 2. We use randomly generated files from 10MB to 50MB as the data set. We set the path of a chosen repository to a non-existent location to simulate a cloud storage failure in repair. Figure 8, plots the response times of all three file operations versus the file size and show that RS has less response time than FMSR codes and Replication has highest response time in file upload. For file download, RS has lowest response time follow by Replication codes, FMSR has highest response time. For instance in the case of n = 4 and k = 2, when uploading a 50MB file,RS takes 33.17s, FMSR codes take 35.02s and Replication takes 65.26s to upload to upload. When downloading 50MB file, RS takes 6.55s, FMSR codes take 7.27s and Replication takes 7.57s to download. FMSR has less response time in repairing a single cloud failure. For instance, in repairing a 50MB file with n = 4 and k = 2, FMSR codes takes 14.01s, RS codes takes 14.53s and Replication takes 21.14s.



(a) File upload



(b) File repair



(c) File download

Fig. 2. Emulation

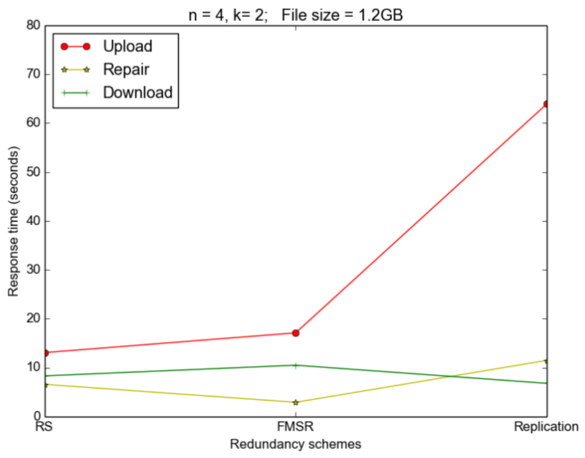


Fig. 3. n = 4, k = 2

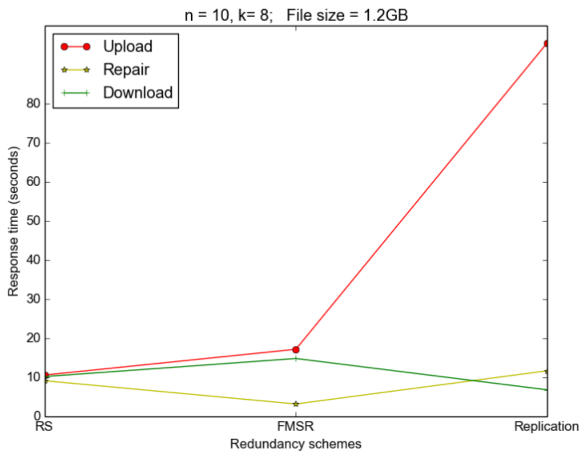


Fig. 6. n = 10, k =8

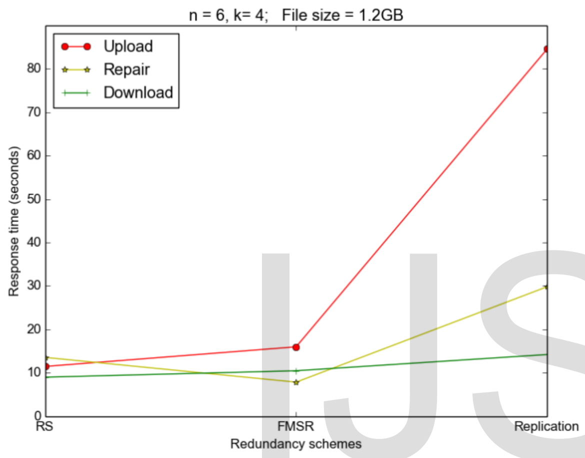


Fig. 4. n = 6, k = 4

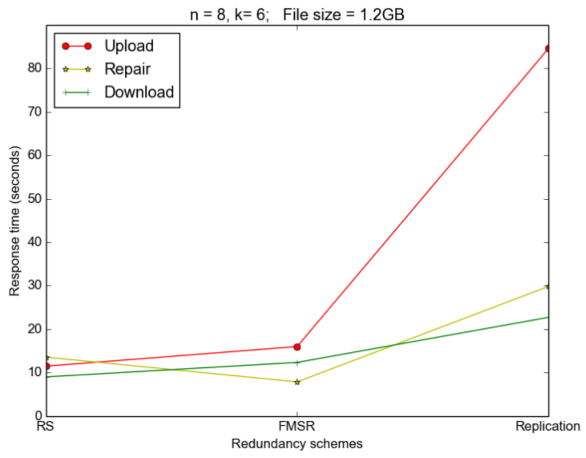


Fig. 5. n = 8, k = 6

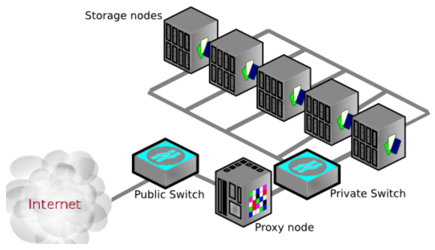


Fig. 7. Implementation of Multiple Clouds Storage using Openstack Swift

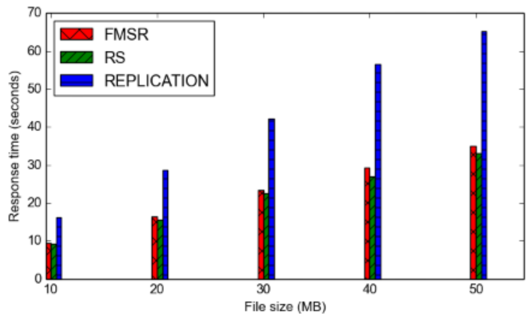
C. Emulation versus Experimentation

We compare response times of the emulation and experimentation/implementation for three operations; file upload, file repair and file down where n = 4, k = 2. As expected, the results reveal that emulation leads to shorter response time than experimentation in file upload, repair and download, regardless of file sizes as shown in figures 2 and 8. This is due to the absence of latency since the emulation is run on a single machine on a private network during uploading and downloading data to storage nodes. For instance, when uploading a 50MB file where $n = 4$ and $k = 2$, in case of emulation as shown in table 2, RS takes 0.063s, FMSR codes take 0.078s and Replication takes 0.136s to upload while in case of experimentation, RS takes 33.17s, FMSR codes take 35.02s and Replication takes 65.26s to upload.

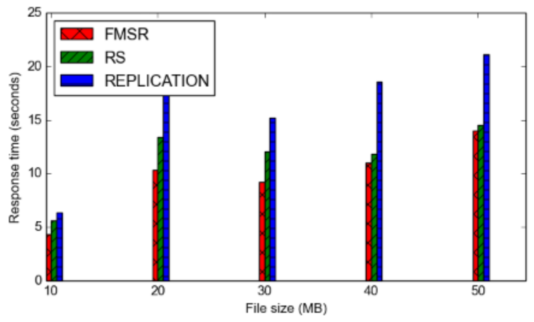
For file download as shown in table 2, in case of emulation, RS takes 0.402s and Replication takes 0.725s to download while FMSR takes 0.459s to download while in case of experimentation, RS takes 6.55s, FMSR codes take 7.27s and Replication takes 7.57s to download.

TABLE II
RESPONSE TIME: UPLOAD/DOWNLOAD 50MB

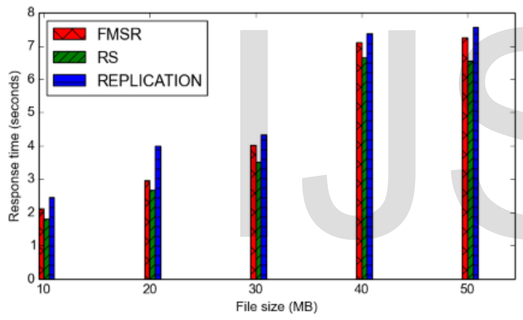
Upload 50MB	Emulation (secs)	Implementation (secs)
RS	0.063	33.17
FMSR	0.078	35.02
Replication	0.136	65.26
Download 50MB	**Emulation (secs)**	**Implementation (secs)**
RS	0.402	6.55
FMSR	0.459	7.27
Replication	0.725	7.57



(a) File upload



(b) File repair



(c) File download

Fig. 8. Implementation

However, when looking at the performance patters of both methods, the results reveal that they are in agreement as they lead to similar relative performance values. This confirms that when designing and evaluating the performance of lightweight cloud storage systems, emulation results can be used with confidence to predict experimental results when cost and availability preclude the use of testbed experimentation/prototyping.

VI. Conclusions

In this paper, we have reviewed and surveyed different multiple clouds storage systems proposed to provide fault tolerance, data integrity and confidentiality. The schemes [11], [2], [10], [21] and [30] use erasure codes to provide availability of the outsourced data in the cloud storage. [12] and [27] use replication technique to provide data redundancy in the cloud. NCCloud [14] uses regenerating codes to provide fault tolerance and repair a permanent single-cloud failure in multiple clouds storage system.

Furthermore, we performed comparative experiments for data redundancy schemes for multiple cloud storage: Replication, RS and FMSR. FMSR codes have less response time in repairing a single cloud failure compare to RS and Replication.

We also compare the response time of the emulation and implementation for three operations: file upload, repair and file download, the results show that emulation has lower response time than implementation, this is due to the latency between the proxy server and storage servers.

The management of the cloud communication infrastructure is a key parameter that may require redesigning existent network management techniques to enable efficient data migration between cloud nodes. Multipath routing techniques such as presented in [9], [6] will be redesigned to support QoS by having different forms of cloud data propagated over different paths form a source to a destination. The cost-based traffic engineering techniques proposed in [8], [7] will also be redesigned to balance traffic over the cloud communication platform to increase throughput and reduce communication delays. Deploying a wireless network [31], [5] to scale the cloud infrastructure over long distances in the rural settings of the developing world is another key issue that needs to be addressed as future research work.

References

[1] Openstack object storage. http://www.openstack.org/projects/storage/.
[2] H. Abu-Libdeh, L. Princehouse, and H. Weatherspoon. Racs: A case for cloud storage diversity. *Proc. of the 1st ACM Symposium on Cloud Computing*, pages 1204–1216, June 2010.
[3] R. Ahlswede, N. Cai, S. R. Li, and R. W. Yeung. Network information flow. *IEEE Trans. on Information Theory*, 46(4):1204–1216, July 2000.
[4] M. Armbrust, A. Fox, R. Griffith, A. D. Joseph, R. Katz, A. Konwinski, G. Lee, D. Patterson, A. Rabkin, I. Stoica, , and M. Zaharia. A view of cloud computing. *Communications of the ACM*, 53(4):50–58, 2010.
[5] A. Bagula, M. Zennaro, G. Inggs, S. Scott, and D. Gascon. Ubiquitous sensor networking for development (usn4d): An application to pollution monitoring. *In Sensors*, 12(1):391–414, 2012.
[6] A. B. Bagula. Hybrid traffic engineering: the least path interference algorithm. *In Proceedings of the 2004 annual research conference of the South African institute of computer scientists and information technologists on IT research in developing countries*, pages 89–96, 2004.
[7] A. B. Bagula. Hybrid routing in next generation ip networks. *In Computer Communications*, 29(7):879–892, 2006.
[8] A. B. Bagula. On achieveing bandwidth-aware lsp//spl lambda/sp multiplexing/separation in multi-layer networks. *IEEE Journal on Selected Areas in Communications*, 25(5):987–1000, 2007.
[9] A. B. Bagula. Modelling and implementation of qos in wireless sensor networks: a multiconstrained traffic engineering model. *EURASIP Journal on Wireless Communications and Networking*, (1), 2010.
[10] A. Bessani, M. Correia, B. Quaresma, F. Andre, , and P. Sousa. Depsky: Dependable and secure storage in a cloud-of-clouds. *In Proc. of ACM EuroSys*, 2011.
[11] K. D. Bowers, A. Juels, and A. Oprea. Hail: A high-availability and integrity layer for cloud storage. *In Proc. of the 16th ACM Conference on Computer and Communication Security (CCS)*, pages 187–198, November 2009.
[12] C. Cachin, R. Haas, and M. Vukoli. Dependable storage in the intercloud. *IBM Research Report*, 2010.
[13] B. Chen, R. Curtmola, G. Ateniese, and R. Burns. Remote data checking for network coding-based distributed storage systems. *In Proc. of ACM CCSW*, 2010.
[14] H. C. H. Chen, Y. Hu, P. P. C. Lee, and Y. Tang. Nccloud:a network coding based storage system in a cloud of clouds. January 2014.

[15] F. Dabek, M. F. Kaashoek, D. Karger, R. Morris, and I. Stoica. Wide-area cooperative storage with cfs. *In ACM Symposium on Operating Systems Principles (SOSP)*, 2001.
[16] A. G. Dimakis, P. B. Godfrey, Y. Wu, M. Wainwright, and K. Ramchandran. Network coding for distributed storage systems. *IEEE Trans. on Information Theory*, 56(9):4539–4551, September 2010.
[17] A. G. Dimakis, Y. Wu, C. Sub, and K. Ramchandran. A survey on network codes for distributed storage. *Proceedings of the IEEE*, 99(3):476–489, March 2011.
[18] P. Druschel and A. Rowstron. Past: A large-scale, persistent peer-to-peer storage utility. *In USENIX Workshop on Hot Topics in Operating Systems (HotOS)*, 2001.
[19] W. K. Lin, D. M. Chiu, and Y. B. Lee. Erasure code replication revisited. *In P2P*, 2004.
[20] P. Mell and T. Grance. Draft nist working definition of cloud computing. *http://csrc.nist.gov/groups/SNS/cloud-computing/index.html*, June 2009.
[21] T. G. Papaioannou, N. Bonvin, and K. Aberer. Scalia: An adaptive scheme for efficient multi-cloud storage. *In Proceedings of International Conference for high performance computing and networking*, November 2012.
[22] C. Preimesberger. Many data centers unprepared for disasters: Industry group. *http://www.eweek.com/c/a/ITManagement/Many-Data-Centers-Unprepared-for-Disasters- Industry-Group-772367/*, March 2011.
[23] K. V. Rashmi, N. B. Shah, and P. V. Kumar. Optimal exact-regenerating codes for distributed storage at the msr and mbr points via a product-matrix construction. *IEEE Trans. Inf. Theory*, 57(8):5227–5239, 2011.
[24] R. Rodrigues and B. Liskov. High availability in dhts: Erasure coding vs. replication. *In IPTPS*, 2005.
[25] N. B. Shah, K. V. Rashmi, P. V. Kumar, and K. Ramchandran. Distributed storage codes with repair-by-transfer and nonachievability of interior points on the storage-bandwidth tradeoff. *IEEE Trans. Inf. Theory*, 58(3):1837–1852, 2012.
[26] Y. Singh, F. Kandah, and W. Zhang. A secured cost-effective multi-cloud storage in cloud computing. *Computer Communications Workshops (INFOCOM WKSHPS), IEEE Conference*, pages 619–624, April 2011.
[27] Y. Singh, F. Kandah, and W. Zhang. A secured cost-effective multi-cloud storage in cloud computing. *Computer Communications Workshops (INFOCOM WKSHPS), 2011 IEEE Conference on*, pages 619 –624, april 2011.
[28] C. Sub and K. Ramchandran. Exact-repair mds code construction using interference alignment. *IEEE Trans. on Information Theory*, 57(3):1425–1442, March 2011.
[29] M. Vukolizc. The byzantine empire in the intercloud. *ACM SIGACT News*, 41:105–111, September 2010.
[30] C. Wang, Q. Wang, K. Ren, and N. C. an d W. Lou. Towared secure and dependable storage in cloud computing. *Journal of IEEE TRANSACTIONS ON SERVICES COMPUTING*, 5(2), June 2012.
[31] M. Zennaro, A. Bagula, D. Gascon, and A. Noveleta. Long distance wireless sensor networks:simulation vs reality. *In Proceedings of the 4th ACM Workshop on Networked Systems for Developing Regions*, (12), 2012.