Big MRI Data Dissemination and Retrieval in a Multi-Cloud Hospital Storage System

Antonino Galletta
Department of Engineering, University of Messina
Messina, Italy
angalletta@unime.it

Antonio Celesti
Scientific Research Organizational Unit, University of Messina
Messina, Italy
acelesti@unime.it

Francesco Tusa
Department of Electronic & Electrical Engineering, University College London
London, UK
francesco.tusa@ucl.ac.uk

Maria Fazio
Department of Engineering
University of Messina
Messina, Italy
mfazio@unime.it

Placido Bramanti
IRCCS Centro Neurolesi "Bonino Pulejo"
Messina, Italy
pbramanti@irccsme.it

Massimo Villari
Department of Engineering, University of Messina
IRCCS Centro Neurolesi "Bonino Pulejo"
Messina, Italy
mvillari@unime.it

ABSTRACT

Nowadays, we are observing an explosion in the proliferation of clinical data. In this context, a typical example of the well-known big data problem is represented by the huge amount of Magnetic Resonance Imaging (MRI) files that need to be stored and analysed. Although the Cloud computing technology can address such a demanding problem, data reliability, availability and privacy are three of the major concerns against the large scale adoption of Cloud storage systems in the healthcare context - this is why hospitals are reluctant to move the patients' data over the Cloud. In this paper, we focus on data reliability and availability and we discuss an approach that allows healthcare centres storing clinical data in a Multi-Cloud storage environment while guaranteeing patients' privacy. Experiments proved the feasibility of our approach.

KEYWORDS

MRI, big data, hospital information system, cloud computing

ACM Reference format:
Antonino Galletta, Antonio Celesti, Francesco Tusa, Maria Fazio, Placido Bramanti, and Massimo Villari. 2017. Big MRI Data Dissemination and Retrieval in a Multi-Cloud Hospital Storage System. In *Proceedings of DH '17, London, United Kingdom, July 02-05, 2017,* 5 pages.
DOI: http://dx.doi.org/10.1145/3079452.3079507

1 INTRODUCTION

Information and Communication Technologies (ICT) has recently been gaining popularity in the healthcare sector. As an example, every day a huge amount of Magnetic Resonance Tomography (MRT) are produced and stored as Digital Imaging and COmmunications in Medicine (DICOM) files. However, most of the traditional Hospital Information Systems (HIS) are based on old storage technologies and are not able to manage efficiently this huge amount of data. Cloud Computing and Cloud Storage service providers may represent a good solution to tackle that problem as they provide features such as flexible storage capacity, automatic and incremental backups, etc. Unfortunately, the realisation of a private Cloud for small medium hospitals may not economically be sustainable due to both purchase and management costs. In this paper we propose a cheaper multi-cloud storage system based on the integrated usage of several public Cloud storage providers. However, clinical centres are still reluctant to store clinical data over the Cloud due to reliability and privacy concerns, in fact Cloud providers might discontinue their services or suddenly disappear in the case of a cyber-attack.

The Cloud storage service is a very interesting topic that can potantially allow storing huge amounts of data. The business behind the Cloud storage service is demonstrated by the increasing proliferation of storage providers (e.g., Dropbox, Google Drive, pCloud, Amazon S3 and OneDrive). Dropbox was the first player experiencing with this business model, Google Drive provided a similar service that was followed by many other providers. In order to guarantee data reliability, a recent trend consists in replicating data in multiple different Cloud storage providers. The advantages of this new model are evident: if a provider is not able to deliver its service due to an hardware/software maintenance, data files can be retrieved from another Cloud Storage Service provider. This feature turns out being very important for HIS that have to guarantee specific legal requirements about data availability.

However, if on the one hand the adoption of a multi-Cloud storage system can allow hospitals to guarantee clinical data availability, on the other hand it can lead to new privacy issues. In fact, if hospitals actually adopt this model they would not have any warranties about the data privacy of their patients as malicious users can hack accounts in order to steal sensitive personal data. Our idea is to create an innovative Multi-Cloud Storage system for health data that will be able to solve the well-known data availability, reliability and privacy issues. Our system is aimed at improving the security of existing data replication systems such as secrets sharing [8] and Redundant Residue Number System (RRNS) [3]. Such an approach can be adopted by a HIS to split clinical data in a set of redundant chunks with a particular fault-tolerance degree, so that only a subset of those chunks can be enough to reconstruct the original data. In this way, if we store each chunk of a piece of clinical data in a different Cloud storage provider, the latter will not be able to reconstruct the original data. In this paper, we specifically focus on a mechanism that allows a HIS to securely track the Cloud storage providers where the chunks of each clinical data are stored in. Although those data will be scattered across different locations/sites, the HIS will still be able to retrieve the original information when required. Experiments conducted at the IRCCS "Bonino Pulejo", i.e., a clinical and research centre, prove that our approach is able to track the dissemination and retrieval of big clinical data in a Multi-Cloud storage environment by considering a MRI case study.

The rest of the paper is organized as follows. Section 2 describes related works. In Section 3 we analyse big MRI data, whereas a HIS using a Multi-Cloud storage environment is described in Section 4. Big clinical data dissemination and retrieval are discussed in Section 5. Experiments, considering big MRI data are discussed in Section 6. Conclusion and lights on the future are summarized in Section 7.

2 RELATED WORK

This scientific work focuses on the creation of a multicloud-based HIS which is currently a quite new topic as we did not find any papers within the same scope produced by the scientific community. Conversely, storage systems management is a widely debated topic that is recently gaining more interest with the advent of new cloud technologies. In [5], the authors describe a technique for optimizing the file partition considering a Network Storage Environment (NSE). The authors make XOR operations using splitting and merging tasks on files that have to be protected. The procedure is hard to be applicable to scalable scenarios because it requires particular kernel configurations. In [1], the authors claimed the improvement of file reliability by introducing redundancies into a large storage system in different ways such as erasure correcting codes used in RAID levels 5 and 6, and by introducing different data placement, failure detection and recovery disciplines inside a datacenter. Even in this case the files are divided in chunks. A similar technique is discussed in [9]. The authors present PRESIDIO, a framework able to detect similarity and reduce or eliminate redundancy when objects are stored. The aforementioned works are pretty theoretical and they are not applicable cloud computing. With regard to big data storage solution in cloud computing, in [7], the authors discuss how a cloud based storage system will provide ways for many organizations to handle increasing amounts of information. A file partitioning approach is described in [4]. In particular, the authors present BerryStore, a distributed object storage system designed for cloud service especially for the massive small files storing. With a distributed coordinated controller, the proposed system is able to provide scalability, concurrency, and fault-tolerance. Blocking files is one of the major critical aspects in big data storage. In order to solve the shortages of file storage and parallel computing support issues in fixed-blocking storage, in [10], the authors propose a smart-blocking file storage method. By setting up six grouping factors, the method can determine whether the file should be automatically blocked or not, depending on both the file size and the client bandwidth. A full-stack data storage aimed at clouds is reported in [6]. The authors investigated the cloud applications requirements for supporting Data-Intensive Applications at Infrastructure as a Service (IaaS) level. They considered *Cloud Storage Resource Management*, *Cloud Data Access*, *Metadata Management* and *Data Sewing*. The common idea of all presented works is the adoption of a single Cloud Storage provider, instead, in our system we use a system relying on several Cloud Storage providers.

3 BIG MRI DATA

Nowadays, Magnetic Resonance is a technique widely adopted in medicine. It was introduced at the beginning of the 1980s. This tool is very useful, in fact it allows producing very detailed images of the brain or other parts of the body without the adoption of any X-Ray but using magnetic fields instead. Each exam usually produces several thousands of medical images. In order to simplify data transmission, data storing and data analysis the DICOM (Digital Imaging and Communications in Medicine), an ISO 12052:2006 standard that join medical images with a specific header, was introduced. The DICOM header contains several metadata (patient's name, date of birth, exam id, exam name, etc). It is composed by hundreds of TAGs, each of them having a precise meaning. TAGs are formed by 16 Bytes: 8 of them represent the TAG groupwhereas the others represent the specific element. We remark that the meaning of the second group of Bytes is specific for each TAG group. For instance, considering two different groups of TAGs: 0008 representing the *identifying group*, and 0010 representing *patient group*, the meaning of element 0020 in the former case is the study date whereas in the latter one is the patient ID.

- group=0008, element=0020 $\rightarrow$ study date;
- group=0010, element=0020 $\rightarrow$ patient ID.

We highlight that the size of each DICOM file is a few of kilo-Bytes, but the total size of an exam is greater than 1 Giga-Bytes because it include roughly 20000 images.

4 MULTI-CLOUD HOSPITAL STORAGE SYSTEM

In this Section, firstly we provide an overview of the current trend in Cloud Storage and propose a new model to use Cloud Storage in order to store data. Then we discuss on the Multi-Cloud Hospital Storage system. In our idea the proposed system uses several public Clouds Storage provider services (i.e. OneDrive, Google Drive, pCloud etc) in order to efficiently store and manage medical data. Data security and data privacy are very hot topics in the healthcare domain and storing medical data in the Cloud might expose the Hospital Information System to some threats. For what concerns security, a Cloud Storage provider may not temporarily be able to provide its service due to a hardware/software maintenance (or in a worst case, it could disappear without any notice). In respect of data Privacy, Cloud Storage providers do not assure that malicious users can be able to gain the access to data. In order to avoid the aforementioned issues, we propose an innovative system that securely stores information by spreading user data among different Cloud Storage providers - none of the Cloud providers will have available the minimum information required to reconstruct original data files.

The usage of Cloud storage providers is characterized by the possibility for customers to subscribe to many storage services even for free (e.g., pCloud, DropBox and Google Drive) and to manually manage data upload and download. For reliability reasons data can be replicated in different Cloud storage providers at the same time, however this solution does not solve security and privacy issues.

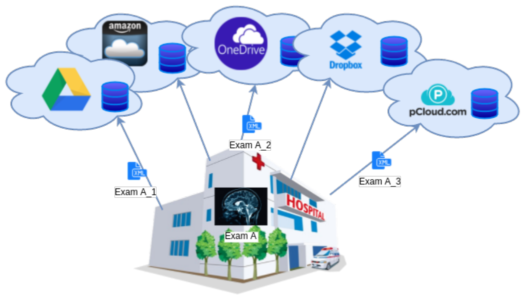

Figure 1: Storage Cloud Services distributed over the Internet.

Our approach introduces a software layer that abstracts heterogeneous Cloud storage providers and allows end-users to upload their files in an efficient way. Figure 1 shows an example on how the proposed approach works. The original file A is split in three chunks, A_1, A_2 and A_3 respectively in pCloud, OneDrive, and Dropbox. The end-user makes a choice about the level of redundancy of each file, in order to overcome failures in data retrieval or data loss. Pieces of file or chunks are wrapped into an XML structure, in order to increase the portability of the system. Any Cloud storage provider sees the XML file as a body containing a chunk of the original file encoded in BASE-64. The failure of the metadata map-file determines the loss of the whole file. To prevent this event and improve the reliability of the proposed solution, the map-file has to be stored in the Cloud, but information on chunks distribution has to be spread over two or more further partial metadata map-files, and deployed over two or more different independent trusted Cloud providers in order to carry out also medadata obfuscation. Since the trusted providers hold only partial metadata map-file, no one will be able, by itself, to reconstruct the whole metadata map-file of any particular user.

The traditional approach used to increase fault tolerance in data storage is to replicate the whole data. Thus, if we need a 3 degree of redundancy for file A (that means we can recover A even if 3 files are lost), we need to deploy 4 replicas of A in different Cloud storage providers. Let consider the following two parameters: p is the minimum number of modules necessary to reconstruct a file and r is the desired redundancy degree. For example, let consider a generic file that can be split in p residue-segments, for example 5. In order to have 3 degree of redundancy, if we set $r = 3$ and we can recover A even if 3 residue-segment are lost. Further details regarding RRSN are available in [2].

Our idea is to apply this approach to allow a HIS managing a huge amount of data generated by current medical devices. Thus, we designed several specific micro-services thay are able to exchange data each others and with a replicated central database system as well as using Cloud Storage and its features as medical exam repository. In this way the HIS can be cheaper and more powerful, in fact, Clouds offers unlimited resources according to the pay per use paradigm. In order to achieve this goal, a HIS should include:

- a replicated central database system able to store all patient's data (personal data and the history of all medical exams);
- a specific micro-service able to store personal data inside the database system (patient registration service);
- a micro-service able to split medical exams in chunks (data split service);
- a micro-service able to spread and retrieve data chunks coming from data split service (data dissemination and retrieval service);
- a micro-service able to recompose data chunks produced by data split service and coming from data dissemination and retrieval service (data re-composition service).

5 SECURE DATA DISSEMINATION AND RETRIEVAL

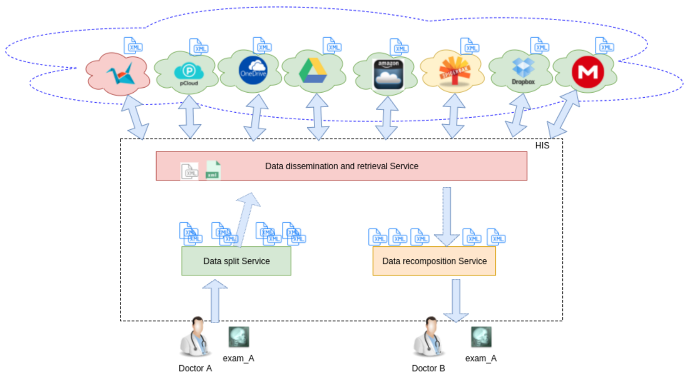
HIS
Data dissemination and retrieval Service
Data split Service
Data recomposition Service
Doctor A
exam_A
Doctor B
exam_A

Figure 2: Representation of the RRNS encoding/decoding.

After having introduced the general concepts regarding our idea, in the following we are going to analyse how data is processed during both the upload and download phases. Figure 2 depicts how these tasks are carried out: Doctor A uploads a medical exam to the data split Service, which divides it into chunks and encloses them into XML files according to the selected redundancy. After that, the XML files are sent to the data spread and retrieval Service, which distributes them to different cloud service providers and stores the chunk location in two metadata files through an Obfuscation method. Whenever Doctor B wants to reconstruct an exam, he selects metadata files related to it and sends them to the data spread and retrieval Service. The latter recovers the original chunks location based on the metadata files and downloads them accordingly. As showed in Figure 2 some Cloud storage service providers may not (temporarily) be able to offer their service (e.g., the red and the yellow Clouds). In this case our service will download chunks from the available providers only (i.e., the green Clouds). In particular, instead of waiting for the transmission of a monolithic block from the Cloud provider, our service can download different residue-segments in parallel from different operators, allowing a more efficient bandwidth occupation. This same method is used by the *Torrent* protocol for increasing the speed of file downloading over the Internet. We remark that our system will download p fragments only to reduce bandwidth occupation, where p is the minimum number of chunks required in the re-composition phase. These chunks are sent to the data re-composition Service in order to build the original exam. At this point Doctor B is able to download the required exam.

In order to track the location of the uploaded residue-segments, for each file a metadata map-file is created. In this section we first provide some details about the structure of that file, and then we discuss the obfuscation technique that allows storing this information in a safe way. The metadata map-file must be accessible only from certain enabled doctors that are allowed to rebuild the original file. Listing 1 shows an example of possible metadata map-file.

Listing 1: Example of possible metadata map-file.

```
<OWNER>ownerInfo</OWNER>
<SEGMENTS>...</SEGMENTS>
<FILE>
    [...]
    <CHUNK num="11">Path/to/the/StorageProviderX/
            94090e1381a1700fb8c34a0069bc6533.xml</CHUNK>
    [...]
</FILE>
```

The first element of the file, *OWNER*, specifies the owner information. The *SEGMENTS* element includes the number of necessary segments required to reconstruct the file (i.e., the value of p). The *FILE* element contains a variable number of *CHUNK* elements. The *CHUNK* tag has the attribute *num*, which refers to the residue-segment sequence number, its content represents a combination of the path associated to the front-end application, the Cloud storage provider for that chunk, and the name of the XML file containing the data. Information stored within the above XML document will allow building up the original file during the decoding process. Depending on the number of available providers and the number of XML chunks, providers can be in charge of storing one or more chunks.

Listing 2: Example of the servicelist map-file.

```
<servicelist>
  <PROVIDER>Path/to/the/StorageProviderX/<PROVIDER>
  <PROVIDER>Path/to/the/StorageProviderY/<PROVIDER>
  [...]
</servicelist>
```

Listing 3: Structure of a trusted file.

```
 [...]
<OWNER>ownerInfo</OWNER>
<SEGMENTS>...</SEGMENTS>
<FILE>
 [...]
<CHUNK num="11">
    <CHUNK_REF>94090e1381a1700fb8c34a0069bc6533.xml</CHUNK_REF>
    <UUID_REF>a72ebba5d9b695c39e6d2193c3cb8057</UUID_REF>
</CHUNK>
 [...]
</FILE>
 [...]
```

It is straightforward foreseeing that the metadata map-file represents a key point of the whole process: its accidental lost or unavailability definitely leads to data loss as retrieving chunks and rebuilding the file becomes impossible. Thus, keeping the map-file in the local file system is not ideal. To improve the reliability of the metadata map-file storing, the data split and retrieve service also stores the map-file into several Cloud providers. To preserve data confidentiality, the map-file has to be split into different partial medatada map-files to be distributed across different independent trusted Cloud providers. This mechanism can be achieved using well known security techniques, in particular combining asymmetric and/or symmetric encryption with the MD5 message-digest algorithm. In order to clarify those ideas, in the following we discuss a methodology to split the metadata map-file into two partial metadata map-files using the MD5. In this example, partial metadata map-files are called *servicelist* and *trusted*. The *servicelist* file is an XML document containing the list of storage providers wherewith hospital holds an a priori agreement. Listing 2 shows an example of a servicelist map-file. The structure of a *trusted* file is shown in Listing 3. The first two elements of the file are equal to the ones specified in the map-file. The *CHUNK* tag within the *FILE* element has the attribute *num*, which refers to the fragment order and the child elements *CHUNK_REF* and *UUID_REF* respectively identifies the name of the XML file containing the data associated to that chunk and the MD5 digest of the fragment location. The *UUID_REF* does not contain the actual service provider in order to obfuscate this information to the storage provider where the file will be uploaded. In fact, the unique identifier is obtained by applying the MD5 to the couple (chunk , provider) in order to prevent brute force attacks aiming at identifying the actual paths associated to the chunks. Uploading the two partial metadata files instead of the

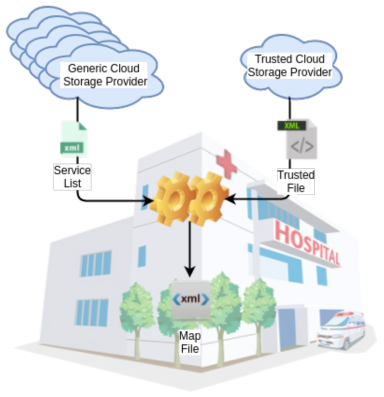
Generic Cloud Storage Provider
Trusted Cloud Storage Provider
xml
XML
Service List
Trusted File
HOSPITAL
xml
Map File

Figure 3: ***map*** **file reconstruction.**

actual whole metadata map-file to different trusted Cloud providers, guarantees knowledge about the file partitioning to the end-user only. This task is highlighted in Figure 3. Starting from *serviceList* and *trusted* files, in order to reconstruct the metadata map-file, for each *CHUNK* element, the data split retrieval service concatenates the content of elements *PROVIDER* contained in the *servicelist* file to *CHUNK_REF* content, calculates the MD5 of the new string and compares it to value of the *UUID_REF* element. We have two possible cases: strings are equal or not. The equality of two values means that the system has found the correct chunk location and can write this information into the metadata-map file, vice versa, if two values are different the system iterates over other providers present in *servicelist* file. In the next Section, we specifically focus on evaluating how the obfuscation algorithm works considering different degree of redundancy and multiple Cloud storage providers.

6 EXPERIMENTS

In this Section, we discuss the data retrieval performances of our system in four different configurations, respectively using: 6, 8, 10 and 12 different Cloud storage providers. In our experiments, we store a chunk per each Cloud storage provider. In particular using the Wrapper described in the previous Section, we simulated 6 datasets respectively composed by 1, 10, 100, 1000, 10000 and 100000 File wrappers. In our scenario we stored serviceList wrapper in a local machine, trusted wrappers instead are stored in a remote machine, in this way we simulated the presence of a third trusted part able to store our information. The system was deployed in two different blades, compute machine is composed by CPU: Intel(R) Core(TM) i7-6700 CPU @ 3.40GHz, RAM 16GB, OS: Ubuntu server 16.04 LTS 64 BIT. Remote machine instead is composed by CPU: Intel(R) Core(TM) i3-6100 CPU @ 3.70GHz, RAM 16GB, OS: Ubuntu server 16.04 LTS 64 BIT. Machines are connected by Virtual Private Network. After that we spread chunks on different Cloud storage providers, starting from servicelist and trusted wrappers we evaluate the time to reconstruct map wrapper. Listing 4 shows the pseudo-code of the reconstruction task.

Listing 4: Pseudo-code of the reconstruction task.

```
...
listServices = readFromServiceList();
listObfuscatedChunks = readFromTrusted();

foreach obfuscatedChunk in listObfuscatedChunks:
        clearChunk = reconstruct(listServices ,obfuscatedChunk);
    map.add(clearChunk);
...
return map;
```

For each dataset, we respectively performed the same operations 30 consecutive times, so as to obtain mean execution times and corresponding confidence intervals at 95%. The histogram of Figure 4 compares the execution times to reconstruct 10000 map wrapper files considering 6, 8, 10 and 12 providers that corresponds to 6, 8, 10, 12 chunks. Execution time is acceptable. In particular, we observe that increasing the number of considered providers we get an acceptable response time. In fact, the execution time of the configuration including 12 providers, present a delay of roughly 9000ms compared to the configuratoin including 6 providers. This means that increasing the number of replication does not have a meaningful impact on performances. This results prove that the proposed approach is suitable for HIS managing big MRI data over a Multi-Cloud storage system.

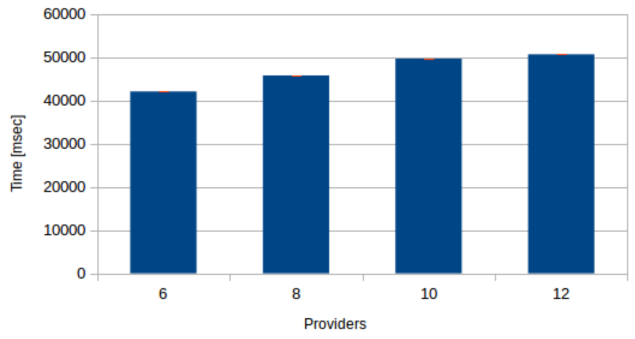

Figure 4: Execution time for 10000 wrapper files.

7 CONCLUSION AND FUTURE WORK

In this paper, we discussed an approach that allows a HIS to manage big clinical data in a Multi-Cloud storage system. In particular, we discussed a service able to spread and retrieve chunks of clinical data among different Cloud Storage providers. In order to map the chunk distribution, we introduced three different XML wrapper files adopting an obfuscation algorithm. This scientific work is the first initiative adopting a Multi-Cloud storage solution for HIS and for this reason we did not find any solution to compare performance of our system. However, experiments showed that our system response time presents a linear trend. The execution time grows up with the increasing number of considered Cloud storage providers. In future works, we plan to improve security functionalities using an encryption algorithm on XML wrappers. Moreover, we plan to switch from XML to JSON wrappers in order to push down parsing and processing times and in order to store information inside a NoSql document database.

ACKNOWLEDGEMENTS

This work was partially supported by the FP7 Project frontierCities under grant agreement no. 632853.

REFERENCES

[1] Bhagwat, D., Pollack, K., Long, D. D. E., Schwarz, T., Miller, E. L., and Paris, J.-F. Providing high reliability in a minimum redundancy archival storage system. In *Proceedings of the 14th IEEE International Symposium on Modeling, Analysis, and Simulation* (Washington, DC, USA, 2006), MASCOTS '06, IEEE Computer Society, pp. 413–421.
[2] Celesti, A., Fazio, M., Villari, M., and Puliafito, A. Adding long-term availability, obfuscation, and encryption to multi-cloud storage systems. *Journal of Network and Computer Applications 59* (2016), 208 – 218.
[3] Chang, C. H., Molahosseini, A. S., Zarandi, A. A. E., and Tay, T. F. Residue number systems: A new paradigm to datapath optimization for low-power and high-performance digital signal processing applications. *IEEE Circuits and Systems Magazine 15*, 4 (2015), 26–44.
[4] Fan, K., Zhao, L., Shen, X., Li, H., and Yang, Y. Smart-blocking file storage method in cloud computing. In *2012 1st IEEE International Conference on Communications in China (ICCC)* (2012), pp. 57–62.
[5] Hai-Jia, W., Peng, L., and Wei-wei, C. The optimization theory of file partition in network storage environment. In *Grid and Cooperative Computing (GCC), 2010 9th International Conference on* (2010), pp. 30–33.
[6] Huo, Y., Wang, H., Hu, L., and Yang, H. A cloud storage architecture model for data-intensive applications. In *Computer and Management (CAMAN), 2011 International Conference on* (2011), pp. 1–4.
[7] Leavitt, N. Storage challenge: Where will all that big data go? *Computer 46*, 9 (September 2013), 22–25.
[8] Traverso, G., Demirel, D., Habib, S. M., and Buchmann, J. As3: Adaptive social secret sharing for distributed storage systems. In *2016 14th Annual Conference on Privacy, Security and Trust (PST)* (2016), pp. 528–535.
[9] You, L. L., Pollack, K. T., Long, D. D. E., and Gopinath, K. Presidio: A framework for efficient archival data storage. *Trans. Storage 7*, 2 (July 2011), 6:1–6:60.
[10] Zhang, Y., Liu, W., and Song, J. A novel solution of distributed file storage for cloud service. In *Computer Software and Applications Conference Workshops (COMPSACW), 2012 IEEE 36th Annual* (2012), pp. 26–31.