Article can be accessed online at http://www.publishingindia.com

A System Architecture for Data Storage in the Cloud

Maryam Hammami*, Hatem Bellaaj**

Abstract

The Cloud storage is the most important issue today. This is due to a rapidly changing needs and a huge mass of varied and important data to back up. In this paper, we describe a work in progress and propose a flexible system architecture for data storage in the Cloud. This system is centered on the Data Manager module. This module provides various functions such as the dispersion of data in fragments, encryption and storage of fragments... etc. This architecture proves to be very relevant. It ensures consistency between different components. On the other hand, it ensures the security and availability of data.

Keywords: Cloud Storage, Data Manager Module, Dispersion of Data, Encryption

Introduction

At the heart of all discussions and communication actors of IT, Cloud Computing occupies a large part of the IT scene in recent years. This technology is considered a new management model and use of computer systems.

Currently, the Cloud Computing is primarily used to treat the intensive workload and provide very high data storage capacity while ensuring their safety.

In this context, several research activities are focused on the proposal of effective systems for data storage in the Cloud. Indeed, traditional storage systems are unable to handle easily a huge amount of data. On the other hand, the implementation of infrastructure provision for data storage requires a fairly significant financial investment. In addition, this infrastructure must be maintained and controlled in order to ensure data availability.

Therefore, data storage systems in the Cloud allow escaping these problems, among others. They have storage models in which the data are maintained, managed, stored and available to the users through the Internet.

The objective of this work is to represent the architecture of our storage service in the Cloud while ensuring consistency between different components.

Through this system, we aim at: (1) ensuring the security of saved data; (2) making the read data operation very fast.

The rest of this article is organized as follows: In section 2, we introduce various storage systems into the Cloud conducted in the literature. In section 3, we give an overview of the architecture of our service, followed by details of the functions and the communications fluxes within the Data Manager module. In section 4, we represent some future work. Finally, we summarize this paper in section 5.

Related Work

In this part, we represent architectures and storage systems defined in the literature.

In (Seiger et al, 2011), the authors define a file storage system incorporating storage vendors in the cloud. The architecture of this system comprises a proxy server located in the intranet. This proxy keeps a cached copy of each file and divides it into fragments according to Erasure Coding (EC) technique.

* Faculty of Economics and Management Sciences,University of Sfax, Tunisia. Email: maryam1990.hammami@gmail.com
** Preparatory Institute of Engineering Studies, Sfax, University of Sfax, Tunisia. Email: bellaaj@yahoo.fr

The EC technique is to divide the data into a number of same-size fragments. In addition, it generates a fragment containing parity data to reconstruct the data in case of loss. This fragment is replicated as many times, depending on the configuration adopted in the EC system. In this way, the EC technique is able to recover the loss of a single data fragment from one of the parity fragments. The parity fragments can be rebuilt, also, either: (1) from fragments of data in case of loss of all parity fragments, (2) from one of the existing parity fragments (Weatherspoon et al, 2011).

The authors guarantee the security of data fragments through a proxy server. It encrypts and stores them in Cloud providers using a protocol adapter. This server ensures the homogenization between various storage services used to insure the coherence of the system and facilitate data recovery (Seiger et al, 2011).

Windows Azure provides reliable data storage through its own data fragmentation technique Local Reconstruction Codes (LRC). LRC is inspired by Erasure Coding technique. The motivation behind this technique is to minimize the number of data fragments for the reconstruction of parity fragments [3].

The formal definition of the LRC is represented by A (k, l, r). It divides k data fragments into l groups, with k/l data fragments in each group. It computes one local parity within each group. In addition, LRC computes r global parities from all the data fragments. Let n be the total number of fragments (data + parity). Then n = k + l + r (Huang et al, 2011).

LRC provides low storage overhead. Among all the fragments that can decode single data fragment failure from k/l fragments and tolerate up to r + 1 arbitrary fragment failures. It can achieve the Maximally Recoverable property (Huang et al, 2011), which means it can decode any failure pattern which is information-theoretically decodable. Then, LRC tolerates failures more than r +1 (up to l+r), provided those are information-theoretically decodable (Huang et al, 2011).

Yifang Luo and al. [5] represent their RAM Cloud Storage System (RCSS). It is based on the architecture of Hadoop Distributed File System (HDFS).

HDFS is a distributed, scalable and portable file system developed by Hadoop. It is designed to store large files and distribute them quickly to users. The architecture of HDFS has two elements: (1) NameNode: It ensures the fragmentation of the contents of a file into fragments of 128 MB size and replicates them, independently, in DataNodes. In addition, it manages the file metadata (file location on the disks, access permissions, number of replicas of each fragment, …, etc.); (2) DataNode: It stores and recovers data blocks.

The architecture of RCSS is characterized by the introduction of the element RAM at DataNode. The motivation behind this improvement is to accelerate data read operations. RCSS will keep in RAM only the most used and accessed files. This based on the priority, assigned to each file during the storage operation, and on the available storage space in RAM. At NameNode, the system handles both the fragments, stored in the RAM, and the I/O operations. In addition, it selects a DataNode to host a file in RAM.

In (Hsu et al., 2014), the authors define a personalized storage system in the Cloud based on the combination of the LDAP database and the pNFS integration.

The LDAP (Lightweight Directory Access Protocol) is an Internet protocol that organizes data management and access to data access permissions. Also, it protects users' privacy, as well as, provides a simple method allows the user to be found the desired information (Huang et al, 2011).

The pNFS is NFSv4.1 (Network File System) version of the protocol. It allows the clients to read and write data, directly and in parallel, to and from the physical storage devices (pNFS).

The system works as follows: The client sends a registration message to the Primary LDAP server. The system saves the customer information in the user profile database and those of Cloud Node in Node profile database. On the other hand, the Diretory Server Module assigns a set of cloud Node for each user. In the file storage process, the pNFS Server fragments the file and notifies the customer to complete the storage process at Cloud Node. Then, this node maintains the metadata (list of users and their files) into a database. In addition, each Cloud Node is coordinated with other nodes in order to access to a data in case of node failure.

In (Papaioannou et al., 2012), the authors define data storage model with various Cloud providers named SCALIA. The purpose behind this system is to determine

the effective location of the data, already saved or to save, in one or more Cloud providers based on historical data access.

The SCALIA architecture has three layers: (1) Engine layer: It breaks the data into blocks of data and storage them in the best combination of Cloud providers. In addition, it elects a single motor to hold all the data accessed or modified during the last operation. (2) Cache layer: It stores copies of data accessed previously in order to minimize costs and response time for data read operations. (3) Database layer: It stores the metadata of each data (size, access policy, etc.) and keeps track of historical data access.

The SCALIA system adopts the Multi version concurrency control approach. This approach is to keep the old data to the obsolete state. If data is updated simultaneously by multiple data center, the system keeps the latest version or asks the user to choose the appropriate version.

System Design

Service Architecture

The architecture of our storage service includes mainly five modules. The Figure 1 shows the overall modules and the fluxes of data exchanged between them.

Workload Balancer Module

This module is directly related to the user interface. It receives requests from clients and forwards them to the appropriate module. If there is a consultation process, the request is sent, first, to the cache module. In case of non availability of data, the request will be redirected to the Data Manager Module. Otherwise, the request is forwarded to the Data Manager Module.

Data Manager Module

The data manager is the most important module in our architecture as it is in direct or indirect relationship with all other modules. It provides various features such as fragmentation, encryption and data storage.

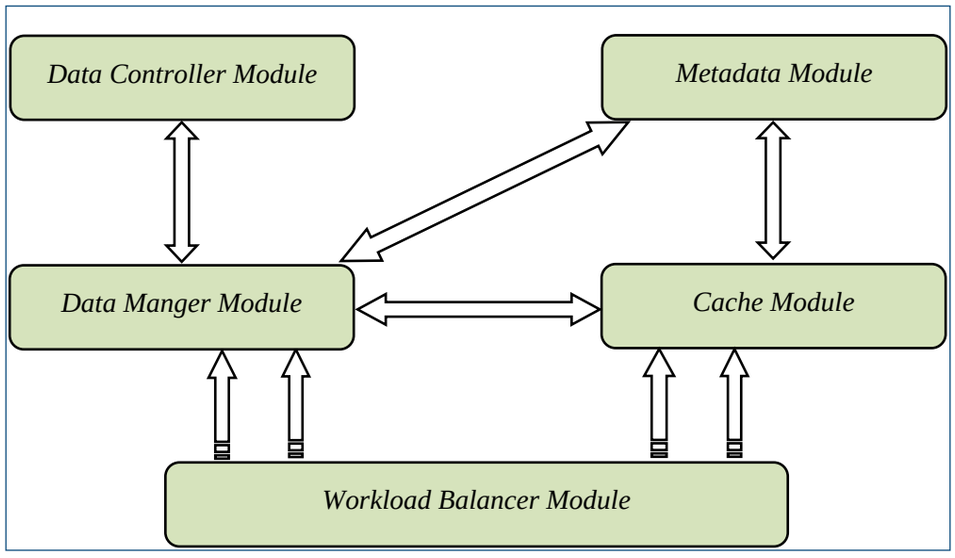
Data Manager Module

The data manager is the most important module in our architecture as it is in direct or indirect relationship with all other modules. It provides various features such as fragmentation, encryption and data storage.

Metadata Module

The Metadata Module stores all associated metadata for each data (name, type, Storage Service_id, encryption

Figure 1: Architecture of the Proposed Storage Service in the Cloud



key and frequency of access) in a MySQL database. This database can be shared and replicated as many times as necessary to achieve the performance and the expected levels of availability.

Cache Module

This module is integrated in order to accelerate the response time for data read operations. It keeps only a copy of frequently accessed data based on the retrained value from Metadata module.

Data Controller Module

This module interacts with data manager module, periodically in order to trigger the control process. This process is based on a comparison between two values calculated according to a specific function that will be the objective of further work.

Data Manager Module

The Data Manager Module has two basic components: a location service and multiple storage services (see Figure 2).

Storage Service

The Data Manager module consists of several storage services. Each of them has an identifier so that the location service can index it.

The storage service maps the data storage procedure. It performs various tasks to achieve a secure data storage.

Architecture

The figure 2 shows that the proposed architecture contains 3 layers: Partitioning layer, Cryptography layer, Storage layer.

Partitioning layer

The partitioning layer provides in the first place, file fragmentation into fragments according to the Local Reconstruction codes technique. This technique is chosen, among others, based on the properties of LRC which are mentioned in [3]. Our attention is granted to the ability of the LRC to recover data lost.

Furthermore, this layer keeps and manages all metadata corresponding to the location of stored fragments in each storage services. Indeed, this can make our storage service very flexible. It can index to all stored data and reconstructs a fragment in case of its loss.

Cryptography Layer

The role of this layer is to ensure data security. It encrypts data fragments using Blowfish encryption technique. This choice is based on evaluation and experimental results, between different techniques, conducted in (Mandal, 2012).

Figure 2: Architecture of the Data Manager Module

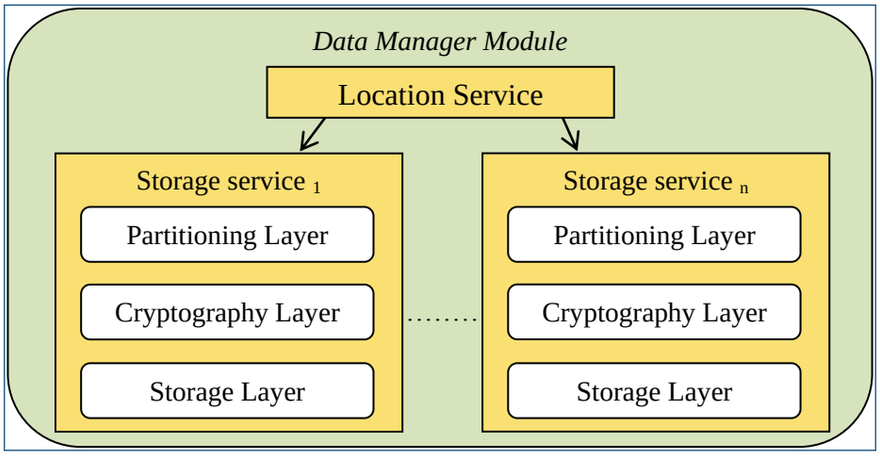
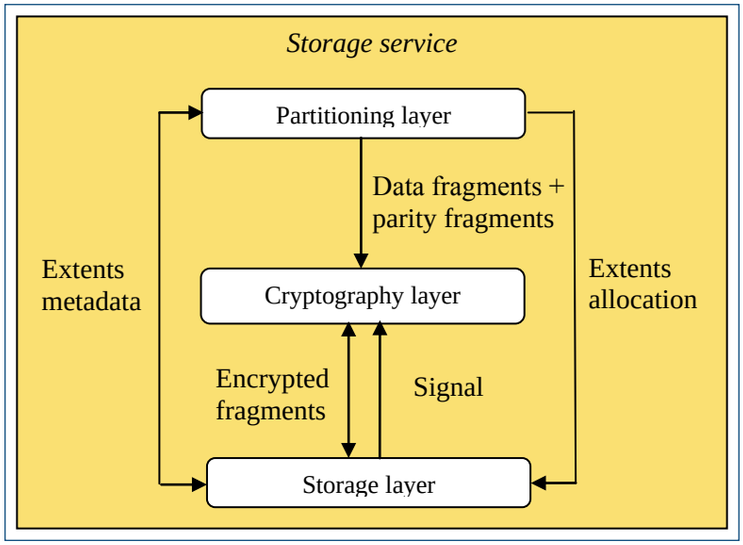


Figure 3: Communication Fluxes within the Storage Service



Storage Layer

This layer storages all encrypted fragments. It keeps the contents of each fragment to a file. Subsequently, it randomly assigned to each disk the fragment to save according to an allocation method.

The Storage layer allocates for each fragment one or more Extent(s). An Extent is a contiguous set of disk blocks allocated to a File. It is defined by the following values: the disk_id, the address of the first block and the length of the Extent. This allocation method is defined in (Andrea et al., 2010).

Interaction between Layers

The figure 3 shows that the storage layer interacts with the two upper layers.

Interaction with Cryptography Layer

The Storage layer and Cryptography layer interact through two fluxes:

- Flux 1: The Storage layer sends a signal that informs that storage operation was realized. Upon receipt of this flux, cryptography layer transmits the encryption key to the location service.
- Flux 2: This flux is bidirectional. It transmits encrypted fragments.

Interaction with Partitioning Layer

The communication between the storage layer and the partitioning layer is achieved by:

- Flux 1: It contains an allocation order of Extents sent by partitioning layer. This flux is transmitted, by default, when receiving a data storage request. For the update operation, the partitioning layer controls this flux. It transmits it, again, when all the allocated extents are already used.
- Flux 2: This flux transmits the Extent(s) metadata appropriate to each fragment. Upon receipt of this flux, the partitioning layer saves these metadata.

Location Service

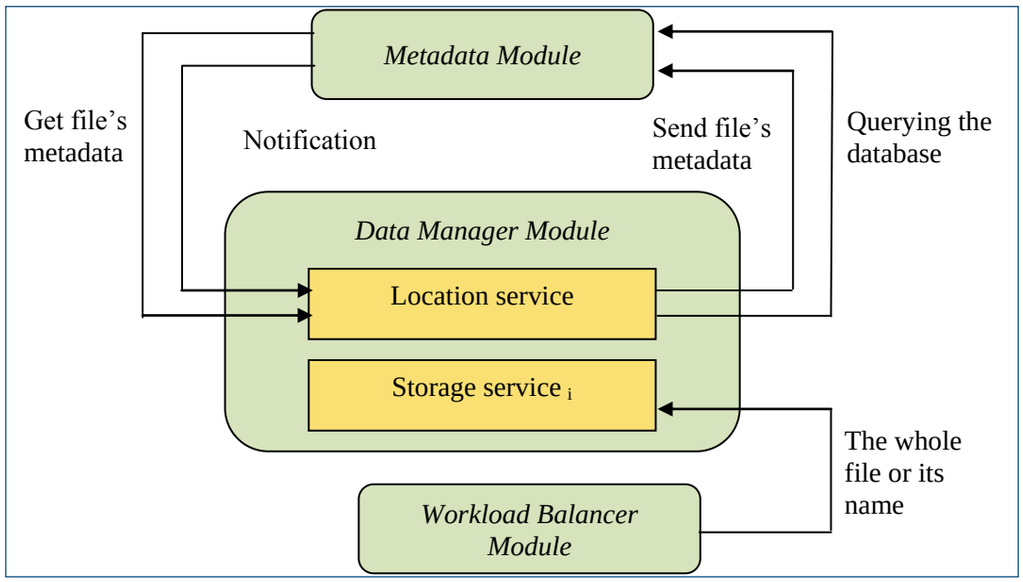
The location service manages all storage services within the data manager. It ensures the distribution or indexing the received operations to the designated storage service.

This service is responsible, also, to ensure the communication between the components of our storage service through two categories of fluxes: of intra-communication flux and inter-communication flux.

Inter-Communication Fluxes

Inter-communication fluxes represent exchanges of information between the data Manager Module, in

Figure 4: Inter-Communication Fluxes with the Data Manager Module



particular at the Location Service, and the other modules.

The Figure 4 illustrates the exchanged of fluxes between the Workload Balancer module and the Metadata module.

Communication with the Workload Balancer Module

The communication takes place in a one-way direction from the workload balancer to the location service. In this case, the workload balancer charges to send:

- either the whole file issued upon receipt of a backup or update operations to the location service. Then, the location service refers to a storage service and keeps, temporally, the service identifying in memory until it receives the file's metadata.
- either the file name that the client wishes to consult. In this case, the location service keeps, also, in memory the file name and prepares an interrogation request to issue the Metadata module.

Communication with Metadata Module

These Communications are conducted in one way or another. The location service sends two fluxes to the Metadata module:

- Flux 1: It transfers the metadata of each file in order to save them in the database.
- Flux 2: It queries the database, according to the file name, for operations of consulting or updating.

In the other part, the metadata module interacts according to the received flux and transmits a response to the location service:

- Reply 1: This is a notification indicating that the metadata is stored (corresponds to flux1). In this case, the location service deletes all data stored temporarily.
- Reply 2: This flux contains metadata needed to index the stored data in the storage service (corresponds to flux2).

Intra-Communications Fluxes

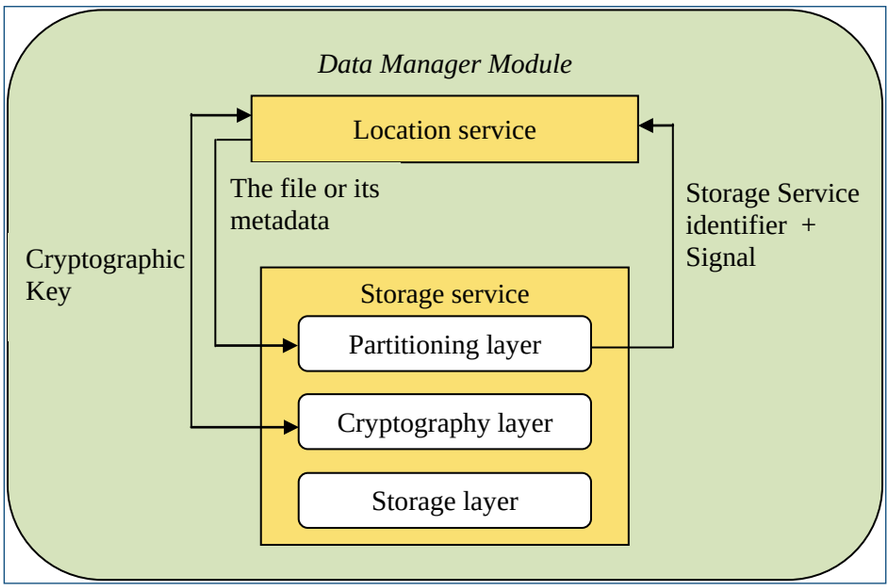
This category represents the flux of communication within the Data Manager Module (see Figure 5).

Communication with the Partitioning Layer

The location service is interacting with the partitioning layer through two fluxes:

- Flux 1: The location service sends the file or its metadata to the partitioning layer depending on the nature of the operation. For a backup or update operation, this layer remains in the waiting state until receiving an acknowledgment from the partitioning

Figure 5: Intra-Communication Fluxes within the Data Manager Module



layer.

- Flux 2: This flux transmits from the partitioning layer to the location service when the service receives a data storage request. It includes two types of information :

- a signal in which it indicates that the operation is performed without interruption. Upon receiving this information, the Location Service changes its pending state to the success state.
- storage service identifier: This information is added to others in the memory of the Location Service. Then, it transmits them to Metadata module.

Communication with Cryptography Layer

The location service interacts with the cryptography layer for:

- receiving the key generated by the encryption algorithm. This key will be saved in the Metadata module with other file metadata.
- transmitting the appropriate key to access the required file. Using this key, cryptography layer can decrypt the data fragments.

Future Work

In this paper, we describe a work in progress and propose a flexible system architecture for data storage in the Cloud without giving up data security. Our future work is oriented towards the construction and improvement of our storage service based on the following points:

- Define a structure to manage extents metadata within the portioning layer as well as file's metadata in the Metadata module.
- Increase the data security level by defining a data control mechanism to be implemented in the data control module. This ensures the integrity of stored data.
- Define a mechanism to manage all copies of data which are stored in Cache module.
- Integrate a system to authenticate users before accessing to their data. This increases the security of our storage service.
- Allow the data owner to share his data with others. Indeed, the user attributes access privileges to each user. This can protect data against their disclosure or modifications by unauthorized third parties

Conclusion

This paper defines the architecture of our storage service in the Cloud. We describe briefly the functionality of each module. In addition, we have given more importance to the Data Manager module. Indeed, we have clarified the adopted techniques and fluxes of information exchanged in each layer of this module.

Through our service, we could: (1) ensure data security using the Blowfish cryptography technique; (2) insure the availability of stored data by the Local Reconstruction Codes data fragmentation technique; (3) accelerate data read operations through the Cache Module.

Acknowledgements

The authors would like to thank everyone, just everyone!

References

Andrea, C., & Remzi, H. (2010). *File Allocation*. Working Paper 10.Wisconsim, US: University of Madison.

Calder, B., Wang, J., Ogus, A., Nilakantan, N., Skjolsvold, A., McKelvie, S., Xu, Y., Srivastav, S., Wu, J., Simitci, H., Haridas, J., Uddaraju, C., Khatri, H., Edwards, A., Bedekar, V., Mainali, S., Abbasi, R ., Agarwal, A., Fahim, M., Ikram ul Haq, M., Bhardwaj, D., Dayanand, S., Adusumilli, A., McNett, M., Sankaran, S., Manivannan, K., & Rigas, L. (2011). *Windows Azure Storage: A Highly Available Cloud Storage Service with Strong Consistency*. Paper presented at the 23th ACM Symposium on Operating Systems Principals in New York, USA.

Hsu, C., Luo, G., & Yuan, S. (2014). Personalized cloud storage system: A combination of LDAP distributed file system. *Genetic and Evolutionary Computing, Advances in Intelligent Systems and Computing*. 23(8), 99-408.

Huang, C., Simitci, H., Xu, Y., Ogus, A., Calder, B., Gopalan, P., Li, J., & Yekhanin, S. (2011). *Erasure coding in windows azure storage*. Paper presented at the Advanced Computing System Association in BOSTON, MA.

Luo, Y., Luo, S., Guan, J., & Zhou, S. (2013). A RAM Cloud storage system based on HDFS: Architecture, implementation and evaluation. *System and Software Elsevier*, 86(3), 744-750.

Mandal, P. (2012). Evaluation of performance of the symmetric key algorithms: DES, 3DES, AES and Blowfish. *Global Research in Computer Science*. 3(18), 67-70.

Papaioannou, T., Bonvin, N., & Aberer, K. (2012). *Scalia: An Adaptive Scheme for Efficient Multi-Cloud Storage*. Paper presented at High Performance Computing, Networking: Storage and Analysis in USA.

pNFS. http://www.pnfs.com

Seiger, R., Groß, S., & Schill, A. (2011). *Sec CSIE: A secure cloud storage integrator for enterprises*. Paper presented at the 13th conference on commerce and enterprise computing (CEC) in Luxembourg.

Weatherspoon, H., & Kubiatowicz, J. (2002). *Erasure coding vs. replication: A quantitative comparison*. Paper presented at the first International Workshop on Peer-to-Peer Systems in London, UK.