Webology (ISSN: 1735-188X)
Volume 18, Number 6, 2021

Enhancing Performance Of Multi-Cloud Storage Environment Using Modified Erasure Coding Technique

V. L. Padma Latha †, Dr. N. Sudhakar Reddy ††, and Dr. A. Suresh Babu †††

†Research Scholar, Department of CSE, SVCE Tirupati, JNTUA University, Ananthapur, India.

††Professor, Department of CSE, SVCE, Tirupati, Andhra Pradesh, India.

†††Professor, Department of CSE, JNTUA University, Ananthapur, India.

Abstract:

Cloud services aren't without flaws. Even giants in the cloud, such as Amazon Web Services, have outages. Measures to safeguard data from disruptions should be part of any successful cloud computing strategy. Data Availability in the Cloud refers to the capacity to provide data and services to users regardless of outages or other disasters. Planning for redundancy and data dissemination is the best way to ensure high data availability. Implementing redundancy and data dissemination may be done in a variety of ways. Replication and Erasure Coding Approaches are two of the most often used techniques. Many top cloud storage providers (CSPs) are now using these strategies to avoid data loss and to recover quickly from disk failures. This project focuses on leveraging OpenStack-Swift to create an object storage cloud that uses replica and erasure coding methods to deliver high data availability and efficient storage.

Keywords: Cloud Data Monitoring, Object based storage, Erasure coding

1. Introduction:

In the modern era, day to day the data is producing enormously. To store that data in a secured way, processing the big data become a big challenges for the service providers. With the storage of infinite amount of data, the service provides should consider one question that how to store the tremendous amount of data reliably by considering both data storage overhead and data integrity.

In the olden days the data can be stored in the personal devices, but while generating tremendous amount of data by the users, the bandwidth is decreasing and the storage cost is also increasing. So, the industrial companies such as google, amazon, etc., provide services to the users with higher bandwidth storage infrastructure for storing their data in pay as you go model. The major disadvantage with this method is whenever the disk fails in the provided data center or when the

outages occurs then the data will be lost and it cannot be retrieved again. So, to overcome this data loss, the loss of data can be prevented by storing the data as finite number of replicas in finite datacenters. Whenever, the data is lost in one data center then the data can be retrieved from the other centers automatically. With this, there is storage overhead because it requires more space to store multiple replicas of data, but data loss can overcome along with that it guarantees the integrity of the stored data.

To overcome the storage overhead and data security, the data which is to be stored in cloud has to be divided into segments and then parity bit is added to the data. Instead of storing the entire data as a replica in many disks, the parity bit is stored to overcome the storage overhead. Whenever the data is lost, based on the parity bit the data is automatically retrieved and prevents the data loss. But this method doesn't work effectively on a single cloud. The vendor lock-in issue will arise and also the cost of storing data is very high and data loss is also more for single cloud.

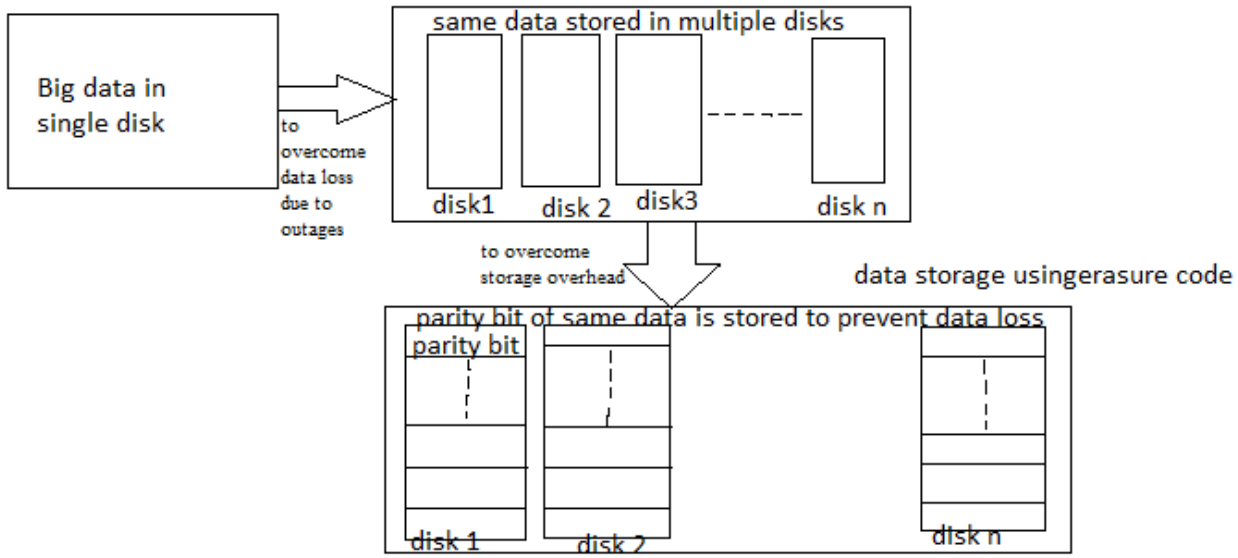

Figure 1: Data Storage in Multi-cloud Environment

To overcome all the issues which arise from the single cloud, a multi cloud storage system came into existence. In multi cloud storage, the data is stored in different data centers in different geographical location provided by the cloud service providers. With this the data loss is reduced and increases the availability of data. It avoids vendor lock-in and also increases data durability. The main objective in this system is reducing the storage cost and increasing the data availability. This can be achieved by erasure coding, Erasure coding gives better performance in storing the data either in object based storage or in distributed systems. It deals with the large data and also any application or systems that secures data from loss which caused by outages. To achieve better performance, cloud storage service providers like Amazon, Microsoft Azure and Google must provide the infrastructure which supports erasure coding. It works as backup to protect the static data and also it automatically avoids the high costs of replication.

With this erasure coding, the resources are utilized effectively to reduce the consumption of storage for multiple copies. It also handles disk failures and outages efficiently which reduced the data loss and offers high availability and durability of data.

2. Literature Survey:

Lixin Liang, Huan He, Jian Zhao, Chengjian Liu, Qiuming Luo, and Xiaowen Chu stated that the data which is storing divided into n data block where each block contains k data blocks and m parity bits. The n blocks of data are stored in n different data centers after performing the encoding to protect the data from hackers. Whenever, the data is missed or lost due to outages then the data is retrieved back by k data blocks and m parity bits along with that it performs decoding operation automatically and retrieves the data. This method provides high data availability and lessens the data storage capacity.

Maomeng Su, Lei Zhang, Yongwei Wu, Kang Chen, and Keqin Li proposed a new model called triones to place the data in multi cloud storage for effective and efficient performance on data availability, scalability and data loss etc., in order to achieve fault tolerance and avoiding vendor lock-in, an erasure coding is used to find the optimization of data placement in erasure coding. This works effectively for the simple data requirements and achieves more than 50% of reduction in latency.

The most important issue in multi-cloud storage is how to store the data cost effectively and with high availability, this problem is addressed by the authors named Pengwei Wang, Caihui Zhao, Wenqiang Liu Zhen Chen, Zhaohui Zhang. To achieve these multi objectives, the authors used non-dominant sorting genetic algorithm, which doesn't provide one solution easily for all the objectives, to find the better solution which satisfies all the objectives. This algorithm works well when all the objectives are satisfied for at least one solution.

A. Parallel and Distributed Storage Systems:

In recent years, parallel and distributed storage systems have seen significant evolution. New architectures and applications have quickly risen to the forefront of development. Cross-fertilization of parallel and distributed technologies with other rapidly growing technologies has resulted in these advances. Reviewing and evaluating these new advancements in comparison to new research efforts in the well-established fields of parelle and distribution is critical. Initially, network attached storage (NAS) and Network File Systems (NFS) provided additional storage devices over the network, allowing users to access them through a network connection. There have been several suggestions for scalability, robustness, and security [1]. Because a storage server may join or depart without the consent of a crucial authority, a storage system with a decentralized design offers high scalability. Making duplicates of each message and storing them on separate servers is a simple way to offer resilience against server failures. However, since Z clones result in Z times of growth, this strategy is costly. Using erasure codes to encrypt messages is one

technique to minimize the expansion rate [3]. Each storage server holds a codeword symbol, which is a vector of symbols used to encrypt a message. An erasure error of the stored codeword symbols occurs when a storage server fails. Distributed encoding is supported by random linear codes, which implies that each codeword symbol is generated individually. Each storage server linearly integrates the blocks with randomly generated coefficients to store a message of k blocks. The codeword symbol and coefficients are then saved. A user searches k storage servers for the stored codeword symbols and coefficients in order to retrieve the message. The linear system is then solved.

B. Proxy Re-Encryption schemes

Proxy re-encryption (PRE) enables a semi-trusted proxy to transform a ciphertext meant for one user into a ciphertext intended for another user without revealing the plaintext. To transform a cipher text from one group to another, Chunbo Ma et al. suggested a group-based proxy re-encryption system. Any group member may decode the cipher messages encrypted to the group on their own. Mambo and Okamoto[14] and Blaze et al.[15] offer proxy re-encryption techniques. Using the re-encryption key RK AB, a proxy server may transform a cipher text encrypted with user A's public key (PUKA) to a new one encrypted with user B's public key (PUKB). Because the communications are originally encrypted by the owner, the server does not know the plaintext during conversion. Type-based proxy re-encryption techniques [4] provide more granularity in the re-encryption key provided right. In this sort of arrangement, a user may choose which types of communications he wants to send and with whom he wants to share them. Ateniese et al. [18] present key private proxy reencryption techniques. A proxy server will not be able to verify the recipient's identity given a re-encryption key in a proxy re-encryption scheme. Proxy re-encryption techniques like these provide more anonymity while dealing with proxy servers. Even while pairing operations are used in most proxy re-encryption schemes, there are proxy re-encryption methods that do not [19].

C. Integrity Checking Functionality

The function of integrity checking is another key feature of cloud storage. Once data is stored in the storage system, the user no longer has access to it. The user may wish to double-check that the data in the storage servers is correct. Provable data possession [20], [21] and evidence of storage [22], [23], [24] have been suggested. After that, [25] discusses the public auditability of stored data.

3. Proposed Model

3.1 Object Storage based Erasure Coding Model

Data is divided into fragments, enlarged and encoded with redundant data bits, and saved across a number of various places or storage medium using erasure coding (EC). The original data is split into k parts. In addition, the encoding process generates further m pieces, which operate as

redundancy for the original data. As a result, "n = k + m" pieces are encoded as the result. The coding system may be represented using a tuple, with the 'k' pieces being data fragments and the'm' fragments being parity fragments (k, m). The complete source data might be successfully recovered with enough data and parity pieces. This (k, m) code can correctly rebuild data from any 'k' fragment subset. Because a storage system has so many disks and nodes, spreading fragments throughout the whole set of disks might effectively assure data availability in the event of disk failure.

The Python Erasure Coding Library is known to operate with Python and offers a simple Python interface for implementing erasure codes. The library makes use of liberasurecode, a C-based erasure code library, to get the greatest feasible efficiency. Erasure Coding backends are supported by PyECLib. In OpenStack Swift, this library is used for Erasure Coding. OpenStack is a cloud computing platform that is free and open source. A set of components operating on top of OpenStack offer a range of functionality, such as Nova for computation, Neutron for networking, Swift for object storage, and Glance for image service. As a frontend and for connecting with other components, each component exposes a set of Application programming interfaces (APIs) to users. Swift is the name of the OpenStack Object Storage component. It uses clusters of computers to offer redundant and scalable distributed data storage. Swift, like other OpenStack components, exposes a set of RESTful APIs for accessing data objects. Standard Hypertext Transfer Protocol (HTTP) and Hypertext Transfer Protocol Secure (HTTPS) calls may be used to create, edit, and remove records.

The data saved in object storage is maintained as distinct objects. Each object has data, metadata, and a unique identifier in addition to the data itself. Object-based storage may be thought of as a combination of file and block storage that offers the best of both worlds: a high degree of abstraction, as well as rapid performance and scalability. From top to bottom, the Swift object storage system is arranged in a three-level hierarchy: Accounts, Containers, and Objects. Objects are saved on the local disk as binary files with metadata. An account lists and maintains the references of all containers, whereas a container lists and keeps the references of all objects in that container. Each object is identified by an access path that looks like this: /account/container/object/account/container/object/account/container/object/account/container/object/account/container/object/ac It's worth noting that the hierarchical structure differs from how

the things are physically stored. The rings, on the other hand, keep track of the real locations.

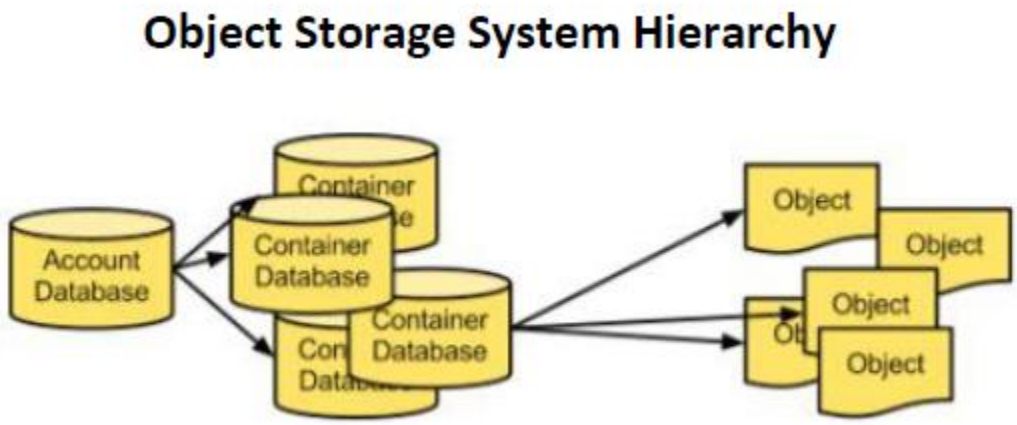

Figure 2. Object Storage hierarchy

The rings in the cluster are used by Swift to locate all entities, including accounts, containers, and objects, since the data is spread throughout the cluster. Swift's ring structures function like this: each ring has a list of all the nodes in the cluster's devices. In order to map the devices in the list, consistent hashing is used to build partitions. The hash value of the path to the item is generated using the same approach, and then used as the index to designate the relevant partition when an object has to be saved or read.

Using a proxy server, the Swift component's APIs are made available to the user. An entity-finding service processes all incoming requests, and then directs them to their respective entities in a completely transparent manner.

3.2 Proposed Data Storage Architecture

A number of servers are used to run the Openstack fast production system (nodes). Typically, controller nodes and fast nodes make up the majority of the network (we can even have other nodes for other services). Swift nodes provide object storage, whereas controller nodes provide proxy services. Object storage is made possible by the integration of Swift nodes with controller nodes.

A. Replication

When a file is posted to Swift, the proxy node buffers it. There are many copies of the file on each of the hard drives in each of the fast nodes.

B. Erasure Coding

There are various code pathways specific to EC policies since the Proxy Server treats Erasure coding differently from replication. These code paths include subclassing and simple conditionals.

The following page provides a high-level summary of how an item moves through the system. Fragments of data are broken down into smaller parts and encoded in a manner that allows the transmission of the data, even if some of the coded fragments are lost. To put it another way, if you send 14 pieces and only 13 are received, one of them is said to be "erased" because of this. The term "erasure" distinguishes EC from other methods in that it refers to the process of dealing with failures rather than the detection of faults. A key feature of EC is the ability to customize the maximum number of erasures it will accept according on the specifics of the application.

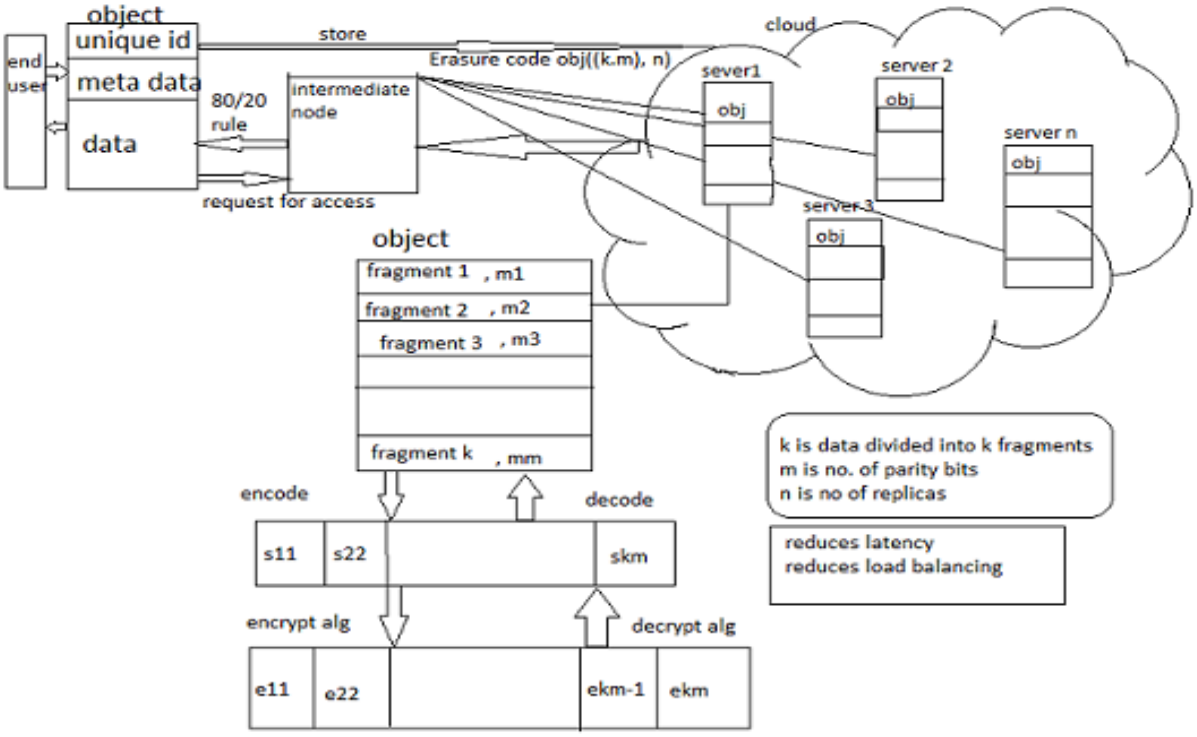

Figure 3. Proposed model for data monitoring in cloud

The procedure shown in the diagram above demonstrates the erasure-coded storage of an object. The proxy buffers incoming objects into segments. Erase codes are used to create fragments at the proxy level. We refer to this as a "fragment archive" since each new fragment is added to the on-disk files that make up this archive. Given our segmenting and fragmenting, this strategy allows us to reduce the amount of on-disk files. ".data" files are used to store fragment archives on the Swift nodes' drives. Retrieval is done in the other direction.

4. Performance Analysis

There can be no more than "N-1" disk failures if you use N-way replication to store N copies on N separate drives. Any one of the three disks that have the original data may be used to restore or retrieve the data. As far as disk failure goes, erasure coding can handle it, but it does it at a

significantly higher storage efficiency. As a result of error coding, the storage capacity is maximized. Coded blocks of the data object are distributed among the 'n' disks in Erasure coding. The encoded data must be decoded before retrieval may take place. The encoding and decoding techniques, it seems, cause an increase in access latency. Erasure Coding, on the other hand, has a far lower access latency than Replica. Replica and Erasure Coding schemes are evaluated and described in terms of performance measures that are specific to each context. Redundancy systems for data are fully dependent on these measures. They seem to be.

• Optimum Use of Space

• Latency in obtaining information

• Restore I/O functionality

• Fix Bandwidth Issues

The usage of storage policy is dependent on a variety of factors, including the size of the objects, the design requirements, and the frequency of access. We use erasure codes if we wish to maximize storage efficiency. We utilize replication strategy if we desire fast access latency. Different storage policies must be utilized for each performance statistic to get the most out of it.

• Small files and frequently viewed files need less delay in accessing. So, the utilization of replication is necessary.

• Large and seldom accessed files need more efficient storage. Hence, the need for erasure coding.

In order to achieve our primary goal of increasing data accessibility while maintaining a small storage footprint, we must adhere to the two principles outlined above. It is much faster and more energy-efficient than simple replication or erasure coding. We use OpenStack-swift to build a storage system that adheres to both of these criteria.

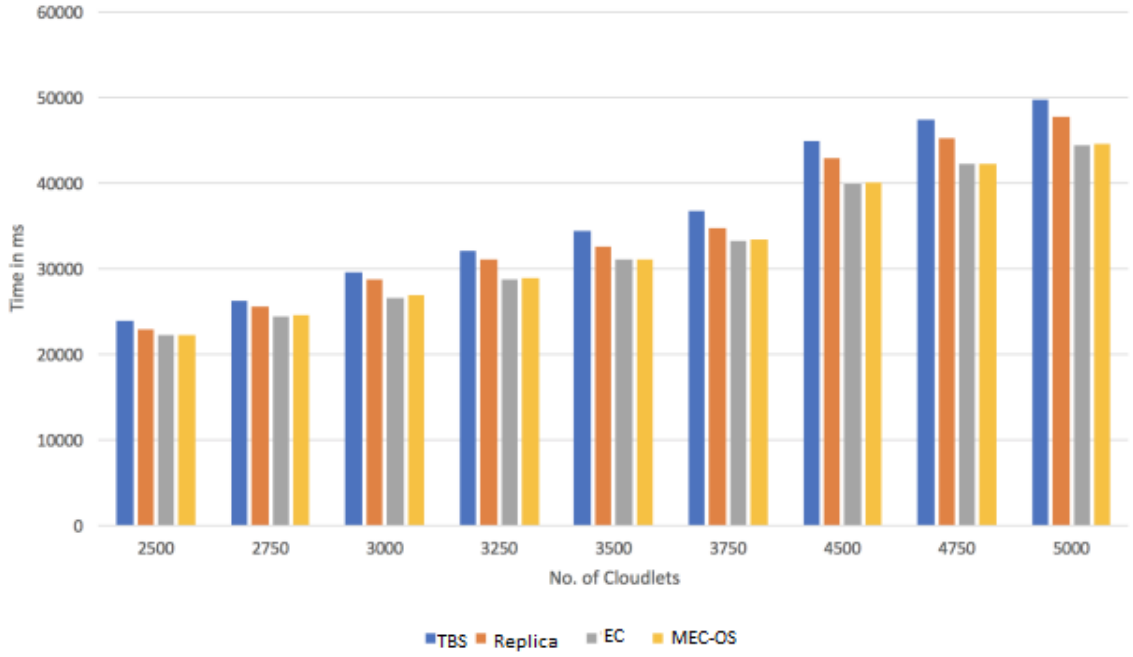

Figure 4: Execution Cost of TBS, Replica, Erasure Code and MEC-OS

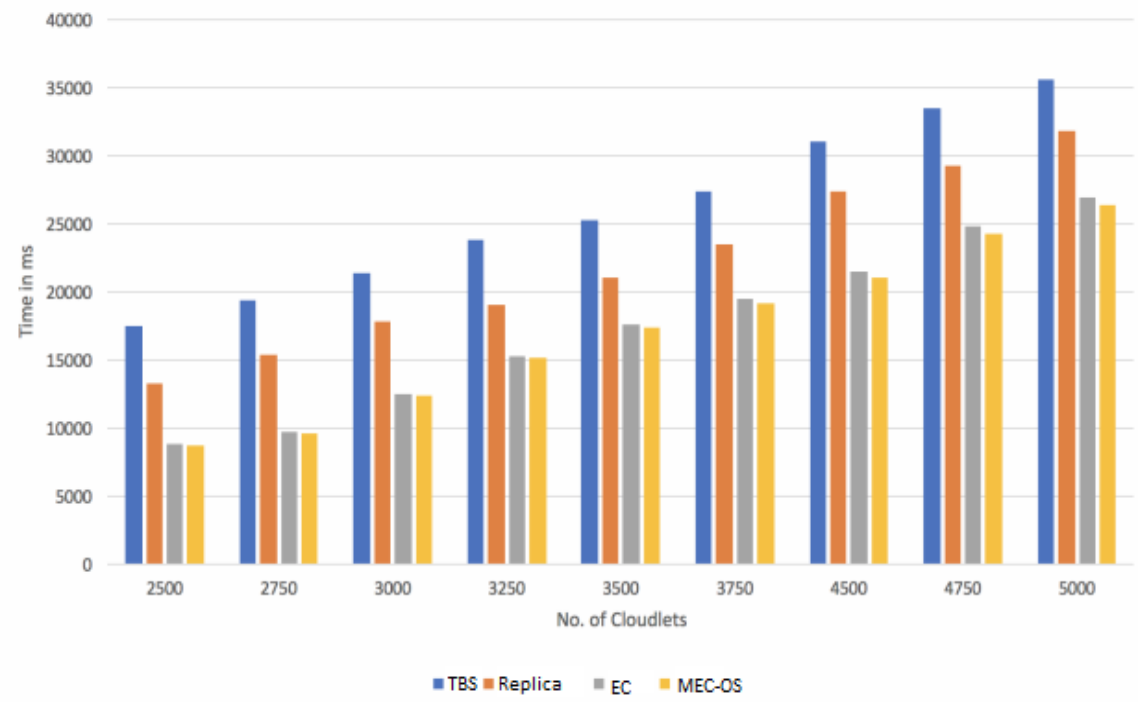

Figure 5: Total waiting time (in ms) of TBS, Replica, Erasure code and MEC-OS

5. Conclusion

In three virtual machines, we created an object storage cloud system that uses both replication and erasure coding. As a result, we were able to increase cloud data availability while using less storage. We can implement this in the actual production system and see a significant reduction in the amount of storage space required to maintain high availability. It is possible to boost the amount of data availability by creating a copy of the erasure coded information. In erasure codes, there are a variety of coding methods that may be used depending on the situation. To date, the system has been implemented using RS codes. A policy may be set up such that it can utilize PyECLib, which has codes for many different coding methods, so we can even employ these various coding techniques. Thus, a high-availability cloud may be built.

6. References

[1] J. Kubiatowicz, D. Bindel, Y. Chen, P. Eaton, D. Geels, R. Gumadi, S. Rhea, H. Weatherspoon, W. Weimer, C. Wells and B. Zhao," Oceanstore: An Architecture for Global-Scale Persistent Storage," Proc. NinthInt' l Conf. Architectural support for programming languages and operating systems(ASPLOS), pp.190-201,2000.

[2] P. Druschel and A. Rowstron,"PAST: A Large-Scale, Persistent Peer-to-Peer Storage Utility," Proc. Eighth Workshop Hottopics in Operating System (HotOS VIII), pp. 75-80,2001.

[3] Adya, W. J. Bolosky, M. Castro, G. Cermak, R. Chaiken, J. R. Douceur, J. Howell, J. R. Lorch, M. Theimer, and R.Wattenhofer," Farsite:Federated, Available, and Reliable Storage for an Incompletely Trusted Environment," Proc.Fifth Symp. Operating

[4] Haeberlen, A. Mislove, and P. Druschel, "Glacier: HighlyDurable, Decentralized Storage Despite Massive Correlated Failures,"Proc. Second Symp.Networked Systems Design and implementation(NSDI),pp.143-158,2005.

[5] Z.Wilcox-O'Hearn and B.Warner, "Tahoe: The LeastAuthority Filesystem."Proc. Fourth ACM Int'l Workshop Storage Security and Survivability (StorageSS), pp.21-26,2008.

[6] H.-Y.Lin and W.-G.Tzeng, "A Secure Decentralized Erasure Code for Distributed Network Storage," IEEE Trans. Parallel and Distributed Systems, vol.21, no. 11, pp. 1586-1594, Nov.2010.

[7] D. R. Brownbridge, L. F. Marshall, and B. Randell, "The Newcastle Connection or Unixes of the World Unite!,"Software Practice and Experience, vol. 12, no. 12,pp. 1147-1162,1982.

[8] R.Sandberg, D.Goldberg, S.Kleiman, D.Walsh, and B.Lyon,"Design and Implementation of the Sun Network Filesystem," Proc.USENIX Assoc. Conf.1985.

[9] M.Kallahalla, E.Riedel, R.Swaminathan, Q.Wang, and K.Fu, "Plutus: Scalable Secure File Sharing on Untrusted Storage." Proc. Second USENIX Conf. File and Storage Technologies(FAST),pp. 29-42,2003.

[10] S.C.Rhea, P.R.Eaton, D.Geels, H.Weatherspoon, B.Y.Zhao, and J.Kubiatowicz, "Pond: The Oceanstore Prototype," Proc.Second USENIX Conf, File and storage Technologies(FAST). Pp. 1- 14,2003.

[11] R.Bhagwan, K.Tati, Y.-C.Cheng, S.Savage, and G.M.Voelker,"Total Recall: System Support for Automated Availability Management," Proc. First Symp. Networked Systems Design and Implementation (NSDI), pp. 337-350,2004.

A. G. Dimakis, V. Prabhakaran, and K.Ramchandran, "Ubiquitous Access to Distributed Data in Large-Scale Sensor Networks through Decentralized Erasure Codes," Proc. Fourth Int'l Symp. Information Processing in Sensor Networks (IPSN), pp. 111-117,2005.

[12] G. Dimakis, V. Prabhakaran, and K. Ramchandran, "Decentralized Erasure codes