EAI Endorsed Transactions
on Cloud Systems

Research Article EAI.EU

On the Adoption of Erasure Code for Cloud Storage by Major Distributed Storage Systems

Aatish Chiniah[1]* and Avinash Mungur[2]

[1] Department of Digital Technologies,
Faculty of Information, Communication and Digital Technologies
University of Mauritius, a.chiniah@uom.ac.mu

[2] Department of Information and Communication Technologies
Faculty of Information, Communication and Digital Technologies
University of Mauritius, a.mungur@uom.ac.mu

Abstract

INTRODUCTION: Traditional Cloud Systems are struggling to cope with the exponential growth of data in todays' distributed application environment. The amount of data online has continuously increased since 2003. From an estimated 5 Exabyte in 2003 to 988 Exabyte in 2010. Presently it is estimated that 5 Exabyte of data are produced daily. To cope with such astronomical load of data, Distributed Storage Systems such as Amazon, Google and Microsoft Azure are becoming the de-facto method for the storage of data. Replication is the method used for providing redundancy. However Erasure Coding is a worthy alternative.
OBJECTIVES: The purpose of this paper is to assess the most used distributed storage systems using different evaluation criteria and identifying how erasure code can be integrated into them.
METHODS: This paper provides a survey of well-known Distributed Storage Systems by using the CAP (Consistency, Availability and Partition Tolerance) Theorem. We go by presenting the solution according to the objectives set and trade-off acknowledged by the designers.
RESULTS: A comprehensive survey is presented using five evaluation criteria (design principle, data model, failure detection and recovery, consistency and security). Adoption of erasure code in Distributed Storage Systems is discussed and its advantages are deliberated. Several open challenges are also put forward.
CONCLUSION: This paper provides researchers in the field with a comprehensive review of Distributed Storage Systems and how the adoption of erasure codes will enhance their capabilities.

Keywords: Distributed Storage Systems (DSS), Erasure Code, Cloud Storage, Redundancy, CAP Theorem.

Received on 07 June 2021, accepted on 31 August 2021, published on 14 September 2021

doi: 10.4108/eai.14-9-2021.170955

1. Introduction

The world is experiencing an exponential growth of data [1] as everything is being digitalized: music, pictures, videos, online gaming, and not to mention Internet related data such as Facebook's posts, Instagram's pictures, tweeter's tweets and so on. The new generation of web applications requires the processing of terabytes or even petabytes of data. Such processing power can only be achieved by using distributed/parallel processing [2]. In

*Corresponding author. Email: a.chiniah@uom.ac.mu order to reap the maximum possible benefit out of distributed processing, the traditional storage model (Relational Databases) does not fit into the bill anymore. Distributed Storage Systems (DSS) have been implemented to provide a helping hand by improving availability, scalability and performance to meet the requirements of today's applications. That is the underlying architecture which powers the world's major web service providers such as Google, Amazon and Microsoft Azure.

There are several reasons for distributed processing. Firstly, applications should be scalable and should reap the benefit of multiple systems as well as multi-core CPU architectures that are ready available in commodity PCs nowadays. Secondly, website servers have to be globally distributed for low latency and failover. For example, as soon as a client logs in to Amazon, there will be thousands of processes that will be triggered to display the best possible service to the customer such as list of previous purchases, list of similar goods bought by other customer, who bought same kind of goods, list of preferences of customer, best deals according to location of customer and so on.

Distributed processing implies distributed data, and to obtain scalability, performance and availability from traditional relational database systems is quite impossible. Several researchers have suggested that this is an end of an architectural era [3]. However RDBMS systems still have their place especially in business applications.

The Cloud can be represented as a stack as shown in the Figure 1. In this paper we investigate the second layer from the top (circled in red). That is storage solutions that would allow the hosting of applications by developers. It is of prime importance to know the characteristics of the platform they are going to use. Since there are numerous distributed storage solutions available, our work has as objective to facilitate the task of developers to opt for the right platform that would meet the requirements of their application.

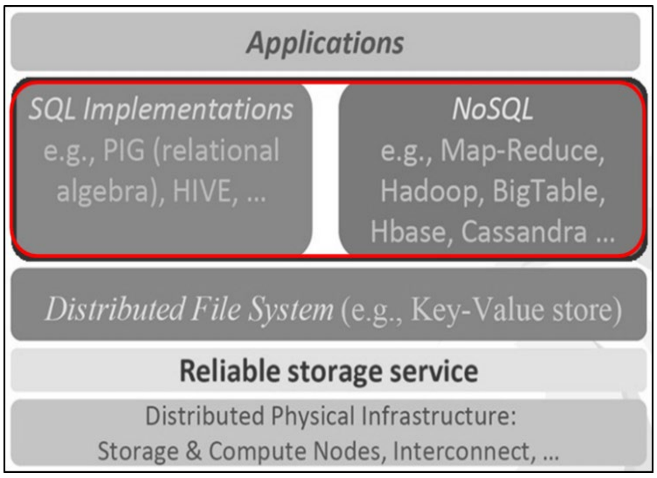

Applications
SQL Implementations e.g., PIG (relational algebra), HIVE, ...
NoSQL e.g., Map-Reduce, Hadoop, BigTable, Hbase, Cassandra ...
Distributed File System (e.g., Key-Value store)
Reliable storage service
Distributed Physical Infrastructure: Storage & Compute Nodes, Interconnect, ...

Figure 1. The new stack of Cloud Storage

In this paper, we survey six Distributed Storage Systems that have been deployed by the world's largest Cloud Service Providers using the CAP Theorem [4]. The CAP theorem states that any distributed systems can achieve two of the three goals at the expense of the third one. The three goals of CAP are Consistency, Availability and Partition Tolerance. The DSS surveyed include Amazon's Dynamo [5, 6], Facebook's Cassandra [6, 7], Haystack [8 - 10], Google's BigTable [6, 11], Yahoo!'s pNuts [12, 13] and Microsoft's Azure [14-17].

After surveying those different DSS against the CAP theorem, the concept of Erasure Code will be introduced, which is the next best alternative to replication for providing redundancy to guard against hardware failures (which many DSS lacks). Erasure Code splits the object to be stored into *n* blocks and creates *m* parity blocks. Any *n* number of blocks can be used to reconstruct the original object, and the system can tolerate up to *m* failures. As such Erasure Code provides higher storage efficiency at the expense of processing power and internal bandwidth. To assess the adoption of Erasure Code by those DSS, an understanding of the different cloud storage information coding implementations is required. To that end, different Erasure Codes are reviewed; namely Reed-Solomon, Hierarchical, Self-Repairing, Regenerating and Locally Repairable Codes.

This paper is organized as follows; Section 2 discusses about the previous surveys that have been published on DSS. In Section 3, the six different DSS with focus on the objectives of each are elaborated. Section 4 summaries the different characteristics of the six DSS. Erasure Code schemes are introduced in Section 5. In Section 6, issues which have been identified during the surveying process are put forward and Section 7 proposes the different solutions to be adopted for the integration of Erasure Code in DSS. Finally Section 8 concludes this paper.

2. Previous Work

In this section, the previous surveys published in the area of Distributed Storage Systems are presented. Several reviews and surveys have been undertaken in this area. Four surveys have been chosen based on the different DSS they have reviewed, or characteristics of DSS they have surveyed.

The first review on DSS was published in an article entitled – The Evolving Field of Distributed Storage – [18] in the IEEE Internet Computing Magazine in its September-October 2001 issue. The article introduced the issue that DSS had to deal with, such as, shared content access, availability, survivability, interoperability, search, caching, load balancing and scalability. The article reviewed two sets of DSS. The first set of DSS (Past, Intermemory and Farsite) performed data archival through the usage of Replication. The second set (Napster, Gnutella, Mojo

Nation and Freenet) of DSShas characteristics of Peer-to-Peer systems. This review article provides a good description of the characteristics required by today's DSS, and explicitly points out deficiencies in the systems reviewed.

The second survey paper – "A Survey of Distributed Storage Systems" – [19], was performed in the year 2004. The paper is a survey of the design and implementation of distribution storage systems. Again the author lay emphasis on the required characteristics of DSS, namely, Local Transparency, Permanent Storage, Consistency, Availability, Performance and Security. This paper examines four Distributed File Systems (DFS), namely Sun's Network File System (NFS), Andrew File System (AFS), Coda File System and Google File System (GFS). Among those we find that only GFS is commercially deployed at present. The paper also reviewed two DSS that operate using Distributed Hash Table, namely, Tapestry and Chord. Of all the DSS reviewed only GFS is widely used nowadays.

"Distributed Storage Systems for Structured Data – a Survey" [20], is another review of DSS performed in the year 2007. The author first surveyed two relational database turned into DSS, namely MySQL Cluster and xFS (Serverless Network File System). Both were not able to meet some fundamental characteristics of DSS. The second part of the survey discusses three systems C-Store, DDS (Distributed Data Structures) and Google's Bigtable which address the scalability problem. They are discussed in terms of API, Partition of data, Consistency guarantees, Persistency and Availability, Performance Optimizations and Complexity. Out of three DSS surveyed, only Bigtable and xFS have been deployed widely.

Another survey on DSS – "Survey of Storage and Fault Tolerance Strategies Used in Cloud Computing" [21], has been published in 2010. The authors surveyed xFS, Amazon S3, Dynamo, GFS, Bigtable and Azure. They compared those systems in terms of Failure Model, Replication, Data Access, Integrity, Consistency guarantees, Metadata, Data placement and Security. Issues identified are failures due to usage of commodity components, location of data and replication.

Another good survey on DSS is "A Taxonomy of Distributed Storage Systems" – [22] and was published in 2008 and revised in 2012. In this survey, the authors presented the areas where research is more needed to improve DSS. The areas identified are: System Function, Storage Architecture, Operating Environment, Usage Patterns, Consistency, Security, Autonomic Management, Federation and Routing and Network Overlays.

Lately the term "Distributed Storage System" has been less frequently employed and has slowly been replaced by the term "Cloud Storage" in order to make reference storage systems for huge amount of data. As such we also reviewed surveys related to Cloud Storage.

One of the first surveys on Cloud Storage [23], "A Survey on Cloud Storage" was performed in 2011, and introduced the concepts of cloud computing as well as discussed about GFS and HDFS as platforms for cloud storage.

A simple but enlightening survey [24], "Overview of Cloud Storage and Architecture" completed in 2018, highlights the architecture of a cloud system, but most importantly discussed on the pros and cons of such systems.

A more recent survey on Cloud Storage [25] which covers more on data placement and file system is the article entitled "A survey on data storage and placement methodologies for Cloud-Big Data ecosystem" done in 2019. Systems such as IBM [26], VoltDB [27], MongoDB [28], Couchbase [29], Cassandra [30], HBase [31], and Ceph [32, 33] were reviewed.

And the latest survey on Cloud Storage [34] is entitled "Issues and challenges in Cloud Storage Architecture: A Survey", whereby the authors discussed about the challenges and issues in terms of data security and data management. For data security, the following issues were found: Integrity, Confidentiality, Access, Authentication & Authorization and Breaches of Data. As for data management issues, the following issues were discussed: Data Dynamics, Data Segregation, Virtualization Vulnerabilities, Backup issues, Availability and Data Locality.

Apart from reviewing some of the surveys on Distributed Storage Systems and Cloud Storage Systems, we also reviewed some other popular DSS as mentioned in Section 1. We compared the different systems in terms of CAP theorem that is in terms of Consistency, Availability and Partitioned-Tolerance and also provided a summary of the different characteristics of each of the surveyed DSS.

Deployment of different Erasure Codes in DSS has not been discussed in literatures so far. The factors acting as barriers and motivators have not been presented in a single presentation. There are several surveys presenting the theories of the different erasure codes, such as [35], [36] and [37]. In this work, the different Erasure Codes are presented under different classifications.

3. Cloud Computing

Different definitions are provided by different literatures for Cloud Computing [38]. Below are some popular examples:

- Cloud Computing is the delivery of computing services—including servers, storage, databases, networking, software, analytics, and intelligence—over the Internet (“the cloud”) to offer faster innovation, flexible resources, and economies of scale [39].

- The practice of using a network of remote servers hosted on the Internet to store, manage, and process data, rather than a local server or a personal computer [40].

Cloud Computing offers various advantages such as pay as you use, scalable services, virtualisation and CDN. All these factors makes cloud computing become more and more transparent in different organisations.

In recent years, research in Cloud Computing have taken different directions such as Edge Computing [41, 42], Fog Computing [43, 44], Mobile Cloud Computing [45] and QoS in Cloud Computing [46-50].

4. Distributed Storage Systems

The primary objective of this paper is to present the available solutions in such a way that developers of cloud applications can easily identify a suitable one that would match the requirements of their own applications. So first of all, we provide a brief introduction of the different types of distributed storage systems and then we differentiate them according to their characteristics as shown in Table 2.

4.1. Types of DSSs

Dynamo
In order to match the requirements of the one of the largest e-commerce operations in the world, Amazon has developed a series of Distributed Storage Systems. One of them is Dynamo [5, 6]. It is a highly-available key-value store, that is uses a primary-key only interface. The key is specified to have access to the data. This is possible as the data objects are relative small (< 1MB).

BigTable
Google’s own data store is known as BigTable [6, 11]. It is a structured distributed storage system which is scalable to a very large capacity (in the range of petabytes spread across several datacentres using commodity servers). BigTable has been deployed in several projects in Google, including web indexing, Google Earth, Google Analytics and Google Finance.

Cassandra
Cassandra [6, 7] is an open sourced decentralized structured distributed storage implemented in Java by Facebook. It was initially designed to support the Facebook’s inbox searching problem and has also been deployed for other application. In fact Cassandra can be considered to a hybrid of Google’s BigTable and Amazon’s Dynamo. Cassandra adopted the salient features from both of those DSS, and resulted in a system that could run on cheap commodity hardware and could handle high write throughput without sacrificing read efficiency.

PNUTS
As all other major web service provider, Yahoo! has been following the footsteps of Google and Facebook by deploying PNUTS [12, 13], which is a distributed database system for Yahoo!’s web application. It has been designed to provide storage for Flickr. As Dynamo, it is a key-value lookup. It provides a hosted, centrally managed with relaxed consistency guarantees as all other DSS do.

Haystack
Haystack [8 - 10] is used as a distributed storage system for storing photos for Facebook. It comes as a replacement for the previous system which was a network attached storage appliances over NFS. Since Haystack is primarily used for storing photos, it is said to be an object storage system.

Azure
Windows Azure Storage (WAS) [14-17] is Microsoft’s cloud storage system that has the ability to provide strong consistency, availability and partition tolerance. It has been deployed since November 2008 and has been used for applications such as social networking search, serving video, music and game content.

4.2. Characteristics of Distributed Storage Systems

Traditionally, a database for web services such as MySQL [51] provides ACID-guarantees for reliability, whereby ACID is for Atomic, Consistent, Isolated and Durable. However these principles are applicable for a single node system only. In the year 2000, Eric Brewer introduced the CAP theorem [52] which states that it is impossible for a distributed service to provide consistency, availability and

Table 1. Comparison of DSSs

Distributed Storage Systems	CAP Theorem		
	Consistency	Availability	Partitioned-Tolerance
Dynamo	-	+	+
BigTable	+	±	+
Cassandra	Adjustable	+	+
PNUTS	-	+	+
Haystack	±	+	+
Azure	+	+	+

partition-tolerance at the same time. As such we present the DSS in terms of these three characteristics to distinguish among them. The characteristics are presented in the Table 1.

Fault-Tolerance is a de-facto characteristic of any Distributed Storage System, as failures are common in such systems. However how those failures are overcome varies from system to system. Consistency and Availability are decisive factors to choose between. In most systems one of them is sacrificed to some extent.

Consistency

Dynamo

While designing Dynamo, specific algorithm has been devised to maintain consistency. Though strict consistency was not the priority, update/conflict resolution is performed during the reads in order to ensure that writes are never rejected. Dynamo targets the design space of an "always writeable" data store. As such consistency is provided by performing object versioning [5]. The consistency among replicas during updates is preserved by a quorum-like technique. Other design principles adopted are Incremental Scalability, Symmetry (same set of responsibilities), Decentralization and Heterogeneity (work distribution according to node capabilities).

BigTable

Consistency is maintained by using a versioning mechanism Thus BigTable allows several versions of the same data as they will be indexed by different timestamps. BigTable uses garbage-collection to keep track of cell versions.

Cassandra

Cassandra uses features from Dynamo and BigTable to ensure consistency. The features adopted from Dynamo are consistent hashing for key generation, Gossip-Based membership algorithm and replication model. From BigTable, the structured consistency model has been adopted.

PNUTS

The consistency models of PNUTS outcast its predecessors. The idea is to provide "per-record timeline consistency". That is PNUTS totally orders the updates that need to be applied in such a way that an update is only processed once all previous updates have been made. Yahoo! uses a message service known as YMB (Yahoo! Message Broker) as compared to the gossip service used in other DSS. However the consistency of PNUTS is still considered to be weak.

Haystack

Since Haystack uses synchronous writes and append-only semantics, consistency does not arise as an issue. Therefore Haystack can be considered to have a fair level of consistency.

Azure

Azure has several features which would achieve a strong level of Consistency. Firstly, Azure is the only DSS working in a layering manner. Each layer has specific responsibilities. As such Azure has been able to achieve the CAP requirements without compromising any of the three characteristics. [14] The Partition Layer provides transaction ordering leading to strong consistency. Separate replication engines (Intra-Stamp and Inter-Stamp) provide synchronous and asynchronous replication and also enlarged namespace also leading to strong consistency.

Design Principle (Availability)

Dynamo

The design principle adopted by Dynamo is the ACID (Atomicity, Consistency, Isolation and Durability) properties. However Dynamo had to sacrifice the "C", consistency, to be able to achieve higher efficiency, which is provided using commodity hardware infrastructure and applying stringent SLA (Service Level Agreement) that states latency requirements of 99.9th percentile of the distribution. Dynamo uses versioning mechanism thoroughly and application-assisted conflict resolution in order to provide an ideal platform for developers. Also Dynamo has been built as a pure peer-to-peer architecture.

BigTable

BigTable has been designed to handle very large files generally measuring in the petabyte range. In fact BigTable is a combination of technologies. It uses services provided by GFS and Chubby. It employs ideas from Log-Structured Merge Tree and uses Distributed Hash Table for content location. These features provide availability to some extent.

Cassandra

The design of Cassandra has been based on the CAP theorem as opposed to ACID in Dynamo.. Cassandra aimed for Availability and Partitioning tolerance as consistency is achieved by using HBase.

PNUTS

PNUTS has been designed using four principles: (i) Asynchrony - the system provides high performance at large scale by using asynchrony, weak consistency and loose coupling; (ii) Automated replication and failure recovery has been put into the design to ensure high availability; (iii) The system has been designed to ensure that it is easy to use, operate and scale. Ease of use implies that the complexity of the system is hidden from the user, and only simple UI is provided. Ease of operation refers to the functions such as self-management and self-tuning, which are in-built into the system. Ease of scaling means that the addition of extra nodes should not affect the overall performance of the system; (iv) Multiple rich access methods are provided, including multiple types of primary tables and secondary indexing.

Haystack
While designing Haystack, four main goals were set: (i) High throughput and low latency - This has been achieved by requiring at most one disk operation per read, which has been made possible by shifting the metadata into main memory instead of using traditional databases; (ii) Fault-Tolerant - Haystack uses replication as the main method for Fault-Tolerance. As soon as a node has failed, another machine is added, the data is replicated to it, from the replicas scattered geographically; (iii) Cost-Effective - The author claimed that "each usable terabyte costs ~28% less and processes ~4x more reads per second than the previous system" [8]; (iv) Simple - Simple and straight forward design and implementation of the system.

Azure
Windows Azure System (WAS) has the following key design goal: (i) Strong Consistency – As compared to all other DSS, WAS offers the strongest consistency, and does so through the use of a layering system and applying both synchronous and asynchronous updates; (ii) Global and Scalable Namespace/Storage – WAS uses a namespace similar to a URL, which is employed to identify the Domain, the Account name, the Partition name and the Object name; (iii) Disaster Recovery – WAS uses replicas as well as erasure codes for redundancy. Erasure Code provides the ability to distribute parts of an object with redundant information that can be used for recovery in cases of disaster. Only a minimum of those parts are required to reconstruct the original object; (iv) Multi-tenancy and cost of storage – slashing the cost of storage has been possible by allowing several clients to share the same physical node to serve heterogonous application data.

Partitioned-Tolerance (Mechanism for Fault-Tolerance)

Dynamo
Data objects are partitioned and replicated using consistent hashing. For redundancy purposes, each key range along with their values is stored over N machines. When the data object is required, a minimum of R (or W) machines is needed to return (or store) the data object respectively. This ensures consistency between the data objects. Failures and membership alterations are identified using a lightweight gossip-based protocol. Nodes communicate with each other every 10 seconds and keep track of the status of the membership. If a node is unreachable, other nodes will simply assume that it is down and continue communicating with other nodes. Consistency among replicas is achieved during update by applying a quorum-like technique with a decentralized synchronization protocol. "Merkle trees" are used so that only the 'diff' of the tree is exchanged in the synchronization mechanism.

BigTable
Replication of data with versioning mechanism is used by BigTable to guard against failure. Given that files are stored in chunks and metadata stored tablet servers, even if some files or tablet servers are lost, they can be reconstructed using the master. However BigTable's dependency on Chubby locking service being up and running turns out to be a single point of failure and leads to downtime if Chubby is not working.

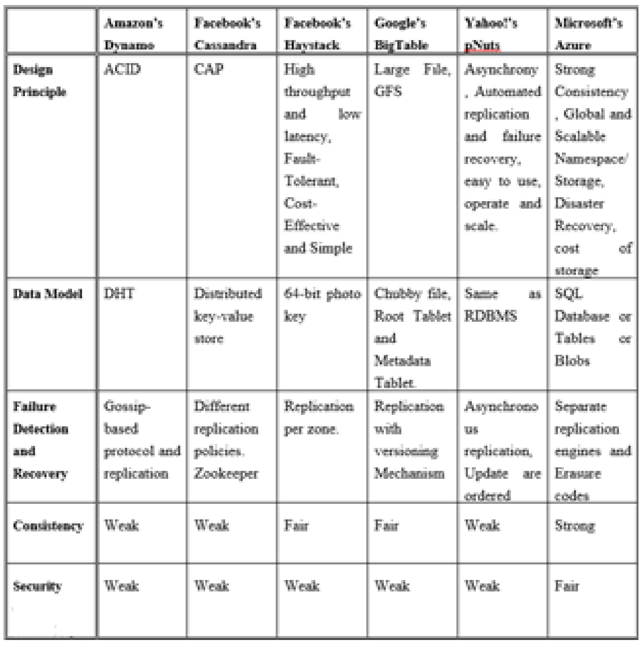
Cassandra
Cassandra uses different replication policies such as "Rack Aware", Rack Unaware" and "Data Center Aware". A leader is chosen among the nodes in the system, using a system called ZooKeeper. Any new node that joins in, will contact the leader, and will be assigned their responsibilities. The leader also ensures an optimal load balance among all nodes in the ring. Cassandra is described as "eventually consistent" because it doesn't guarantee that all replicas will always have the same data. [53]

PNUTS
Yahoo!'s PNUTS is a DSS that uses a centralized storage system and asynchronous replication to ensure low write latency while providing geographic replication. Updates are totally ordered to all replicas and follow a per-record timeline consistency. PNUTS has multi-level redundancy applicable to data, components as well as for metadata and leverages on its consistency model to be able to guard against hardware or network failures.

Haystack
Any photo is replicated to each of the physical volumes mapped to the assigned logical volume. Haystack Directory provides this mapping. So fault-tolerance is achieved by replicating each photo in geographically distinct locations. It is a fairly simplistic method of adding new machines to compensate failed machines and making more copies of the photos. The only single point of failure is the index file. Though it can be reconstructed by restarting the whole system.

Azure
WAS also has two replication engines, namely, Intra-Stamp Replication and the Inter-Stamp Replication. The Intra-Stamp Replication is found in the stream layer and it provides synchronous replication and makes sure that all other replicas in the stamp are in sync. Its main objective is to provide redundancy in case of rack failure. Inter-Stamp Replication works in the partition layer and it provides asynchronous replication and makes sure that replicas in other stamps are in sync. Its objective is to provide redundancy for objects, and to ease disaster recovery. Another solution for cost reduction used in WAS, is the adoption of erasure codes for archival of Blobs. It uses Reed-Solomon codes for erasure coding algorithm. This mechanism has allowed the reduction in storage space from replicas to 1.3 – 1.5x the original data. Erasure codes also increases the durability of the data stored.

Table 2. Summary of features of DSSs



	Amazon's Dynamo	Facebook's Cassandra	Facebook's Haystack	Google's BigTable	Yahoo!'s pNuts	Microsoft's Azure
Design Principle	ACID	CAP	High throughput and low latency, Fault-Tolerant, Cost-Effective and Simple	Large File, GFS	Asynchrony, Automated replication and failure recovery, easy to use, operate and scale.	Strong Consistency, Global and Scalable Namespace/ Storage, Disaster Recovery, cost of storage
Data Model	DHT	Distributed key-value store	64-bit photo key	Chubby file, Root Tablet and Metadata Tablet.	Same as RDBMS	SQL Database or Tables or Blobs
Failure Detection and Recovery	Gossip-based protocol and replication	Different replication policies. Zookeeper	Replication per zone.	Replication with versioning Mechanism	Asynchronous replication, Update are ordered	Separate replication engines and Erasure codes
Consistency	Weak	Weak	Fair	Fair	Weak	Strong
Security	Weak	Weak	Weak	Weak	Weak	Fair

5. Erasure Code

Following from section 4, it is found that the DSS will benefit enormously through the introduction of redundancy. The latter can be provided through the use of Erasure Code (EC).

In Erasure-Coded Storage Systems, a data object is divided into m blocks and they are encoded to generate the n redundant blocks ($m<=n$). Using the properties of erasure codes, the system is able to reconstruct the original data by collecting m out of the n encoded blocks. The data redundancy factor r is defined as n/m, and since m is always less than n, the data can be reconstructed given that the number of storage node failures does not exceed n-m. Examples of such erasure codes include Reed-Solomon codes, Hierarchical codes, Regenerating codes and Self-Repairing codes.

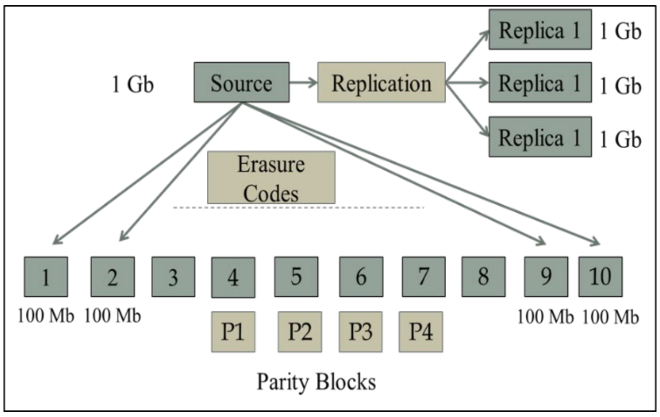


Figure 2. Erasure Code vs Replication

From Figure 2, it is obvious that erasure codes bring a storage capacity saving of about 60–70% compared to replication, having the same amount of redundancy. The issue with erasure codes is that together with the benefits of reducing the storage space needed for the same amount of redundancy, it also brings along additional bandwidth requirements, storage overheads as well as computational loads when creating the blocks and repairing them (Repair Problem). Finding the optimal erasure code in terms of less processing and bandwidth requirement, is still an open problem.

Erasure Codes can be classified as per the specified categories:

1. The Repair Problem (Maximum Distance Separable (MDS) Codes):
   a. Reed-Solomon Codes
   b. Regenerating Codes
   c. Self-Repairing Codes
2. Reducing Bandwidth (Minimum Bandwidth Regenerating (MBR) Codes):
   a. Locally Repairable Codes
   b. Local Reconstruction Codes
   c. Local Regeneration Codes
3. Reducing Storage Overheads (Minimum Storage Regenerating (MSR) Codes)
   a. Exact-Repair Minimum-Storage-Regenerating
   b. Functional Regenerating Codes
   c. Functionally Minimum Storage Regenerating Code – Data Integrity Protection (FMSR-DPI)

Reed-Solomon Codes
Reed-Solomon code [54] is implemented is two parts: the encoder and decoder. The encoder takes a block of digital data and adds extra redundant bits, which are computed using the Reed-Solomon code. The decoder processes each block and attempts to correct errors and recover the original data. If an error occurs during transmission or storage, the error can be detected and corrected depending on the characteristics of the Reed-Solomon.

During the encoding process, data is divided into sets of k data symbols of fixed size s and some amount of parity symbols are generated and added to derive a codeword of the size n. Parity symbols derived will be $n – k$. It also means that up to $n – k$ symbols can be corrected if the $n – k$ data symbols are erased.

Regenerating Codes
Regenerating codes [55] address the issue of rebuilding (also called repairing) lost encoded fragments by contacting only a subset of fragments thus reducing bandwidth requirements.

In traditional RS codes [54], each fragment is considered as a single symbol which is encoded together with other fragments and stored on different nodes. Regenerating code assumes that each fragments consists of α symbols. That is one node will not only store one encoded fragment by several stored as one. During the repair process (a failed node), a random set of d residual nodes (known as helper nodes) are contacted and downloading of $\beta <= \alpha$ symbols from each node. The repair bandwidth, total amount of data downloaded amounts to $\gamma = d\beta$.

Regenerating codes are characterised by [n, k, d] where d specifies the amount of nodes to be contacted for repair. With these new characteristics, came the issue of finding the trade-off between storage and repair bandwidth.

Local Reconstruction Codes
In LRC [56], the placement of blocks and rack awareness is of prime importance. LRC diminishes the amount of network traffic that needs to be transmitted when reading blocks for reconstruction of the object. By placing fragments closer to each other reduces transmission time, thus the reconstruction and repair time is also minimized. Another novelty in LRC is the usage of local parity. A group of blocks are encoded to generate one local parity and then other groups perform the same routine. Finally global parities are created by encoding all the blocks of the associated object. This is done by maintaining the same level of fault-tolerance. The highlight of this method is geared towards repairing single failure.

LRCs are significant for deployments where not only the repair bandwidth, but also the number of nodes to be in communication with, during repair matters. The number of nodes needed to be contacted during repair is often referred to as the repair locality.

Hierarchical codes
Most Erasure Codes such as Reed-Solomon Codes or MBR codes, are meant for solving the computational issues of building blocks. However they do not address the issue of limited bandwidth. Two erasure codes that aim at finding a flexible solution that allows the reduction of the network traffic while maintaining the benefits of traditional erasure codes are Hierarchical [57] and Self-Repairing Codes [58]. However both solutions come with an additional storage costs, which is not a major issue in P2P systems, as storage is amply available.

Self-Repairing Codes
Self-Repairing Codes is a very interesting erasure codes. In [58], the codes are formulated with the acronym of PSRC which stands for Projective Self-Repairing Codes and PSRC is derived from a projective geometric construction. The fundamental difference between Self-Repairing and Hierarchical Codes is in its construction, and satisfying some cardinal properties.

Self-Repairing codes has two main objectives: (i) Minimize the absolute amount of data transfer needed to recreate the lost data from one node and (ii) Minimize the number of nodes to be contacted for repairing one node failure. As such, in Self-Repairing codes, an object is also divided into blocks and some redundant blocks are created to add availability in case repair is required.

6. Issues in DSSs

Having surveyed the Distributed Storage Systems, several issues have been identified and they are, cost of storing replicated data, data location/migration and security. The most pertaining of them is Security to protect the data stored.

Security

Dynamo, BigTable, Cassandra and Haystack
Most DSS have been designed to run in a trusted environment and to serve as a storage system for other services such as social network or e-commerce. Also each operation is meant to be atomic, hence provide some sort of secureness. DSS like Dynamo, BigTable, Cassandra and Haystack operates using those design principles. Haystack does have limitations on usage of cookies and providing minimal security. Encryption could have been implemented in BigTable as its data is structured. Security in those DSS have been omitted to provide faster access to the Big Data.

PNUTS
Security has been one of the requirements set in the design of Yahoo!'s PNUTS. A simple example of the low level of security in PNUTS, is that PNUTS does not always return the most current version of the data requested as it uses asynchronous updates. However little information is available on the application of security measures.

Azure
WAS is the DSS having security implanted at the various layers of its system. Access is granted by using a key generated by Azure for each user. However if the key is hijacked, the system will be compromised. WAS has a security mechanism to guard against faults such as "disk and node errors, power failures, network issues, bit-flip and random hardware failures" – [8].

Cost of Storing Replicated Data

Apart from Azure which uses Erasure Coding for storing archive data, the other five DSS uses only replication for providing storage redundancy to cater for hardware failure primarily. Replication consumes 300% more storage space compare to Erasure Code that requires around 60% – 80% depending on policies used. Another drawback of replication is in terms of bandwidth consumption during the replication process, which is very greedy. The benefits of replication compared to Erasure Coding are that, replication requires less processing and less data nodes are involved. With modern Data Centres equipped with thousands of servers, both these disadvantages of Erasure Coding are now insignificant.

Data Location/Migration

Microsoft Azure and Google's BigTable, have data centers all around the world Therefore Data Location and Migration is not a major issue when it comes to CDN (Content Delivery Network) or Local Replication. However, for the other four DSS, Data Location/Migration could lead to latencies for users further away from the data centers.

7. Adoption of Erasure Code in DSSs

In this section, the factors that are preventing the full adoption of Erasure Code for cloud storage by major DSSs are discussed.

Security

From Table 2 and Section 6, security seems to be the issue that is predominant in most DSSs. Though a lot has been done by cloud providers to tighten up security, security breaches are still prevalent. Can erasure code be a solution to that issue? Given that the data is spilt, encoded and dispersed, a compromised node will not mean a compromised data. Added to that, if the dispersed data are further encrypted, this would increase the complexity for any breach. There have been several attempts in [59], [60] and [61] to implement Secure Erasure Code.

Live Data

Up to now, most deployments of erasure code storage [35], [61] are being used mainly for archiving purposes. There are two barriers preventing the usage of Erasure Code for live data. First, is the fact that the time taken for decoding is still quite high to be acceptable for live data. Secondly, erasure coded data is difficult to update. An update in a block would imply encoding all parity blocks. Update or mutation can happen beyond a single block, and this further entails additional processing. Authors of [62], [63], [64] and [65] have had significant enhancements to implement an updatable erasure coded DSS.

Support for NoSQL Databases

A large of amount of data being stored in the cloud, is through NoSQL databases. Those include transactional data, or data capture from IoT devices. Database Archiving is used to reduce the amount of data stored in the primary tables, but stored adjacently, to be recovered if ever needed. Replication is still used as redundancy method for archived database. Using erasure code for this purpose could enhance storage capacity usage.

Application Layer Control

Up to now, implementation of the erasure code systems have been limited to libraries, accessible through commands only. WebHDFS [66] offers an API to implement web interface to interact with Hadoop systems. However it does not support Erasure Coding functionalities existent in Hadoop version 3.0. [67]

Energy Saving Capabilities

This feature is not a barrier but rather a motivator. Erasure Coded DSS could be designed in such a way that they become energy saving (since parity blocks are only used when there is a failure). Those servers storing parity blocks could be placed in stand-by modes, thus using less energy. In a similar approach, there could be hierarchy of servers storing different levels of parity and placed in different energy saving modes. The authors of [68] did mention that erasure code can have energy saving potentials. Finally in [69], an attempt has been made to have energy efficient erasure code.

8. Conclusion

The increasing amount of data and incapacity of traditional databases to handle those quantities of data has led to a new type of data store named as Distributed Storage Systems. In this work, we have surveyed six Distributed Storage Systems which are widely used as data storage in cloud computing settings. Most of the DSS have been built for a single purpose and they have been performing relatively well, though some had to sacrifice either on Consistency or Availability, but never on performance. Only BigTable and Azure have been deployed in several services.

Erasure Code is tipped to be the next best alternative to providing redundancy of stored data in the Cloud. Till date a lot of fundamental research have been undertaken in order to make erasure code more attractive for Cloud Service Providers and more performing for users. Its adoption is gradually happening as some of the world leading cloud providers (Facebook, Microsoft Azure, and IBM) are integrating Erasure Code in their system. Issues such as security, live update, and support for NoSQL and application layer control have been identified as barriers for the adoption of erasure code by DSS.

Acknowledgements.

We thank Associate Professor Anwitaman Datta, NTU, Singapore, for his scientific insight on the subject matter and constant support.

References

[1] Kennealy P. How Much Data is Created Every Minute? - The EUI Library Blog [Internet]. The EUI Library Blog. 2021 [cited 2 May 2021]. Available from: https://blogs.eui.eu/library/how-much-data-is-created-every-minute/

[2] Marinescu, Dan C. Cloud computing: theory and practice. Morgan Kaufmann, 2017.

[3] Stonebraker M, Madden S, Abadi DJ, Harizopoulos S, Hachem N, Helland P. The end of an architectural era: It's

time for a complete rewrite. InMaking Databases Work: the Pragmatic Wisdom of Michael Stonebraker 2018 Dec 1 (pp. 463-489).
[4] Gilbert S, Lynch N. Perspectives on the CAP Theorem. Computer. 2012 Jan 3;45(2):30-6.
[5] DeCandia G, Hastorun D, Jampani M, Kakulapati G, Lakshman A, Pilchin A, Sivasubramanian S, Vosshall P, Vogels W. Dynamo: Amazon's highly available key-value store. ACM SIGOPS operating systems review. 2007 Oct 14;41(6):205-20.
[6] Kalid, S., Syed, A., Mohammad, A. and Halgamuge, M.N., 2017, March. Big-data NoSQL databases: A comparison and analysis of “Big-Table”,“DynamoDB”, and “Cassandra”. In 2017 IEEE 2nd International Conference on Big Data Analysis (ICBDA) (pp. 89-93). IEEE.
[7] Lakshman A, Malik P. Cassandra: a decentralized structured storage system. ACM SIGOPS Operating Systems Review. 2010 Apr 14;44(2):35-40.
[8] Beaver D, Kumar S, Li HC, Sobel J, Vajgel P. Finding a Needle in Haystack: Facebook's Photo Storage. InOSDI 2010 Oct 4 (Vol. 10, No. 2010, pp. 1-8).
[9] [8] Liu, F., Li, J., Wang, Y. and Li, L., 2019, October. Kubestorage: A Cloud Native Storage Engine for Massive Small Files. In 2019 6th International Conference on Behavioral, Economic and Socio-Cultural Computing (BESC) (pp. 1-4). IEEE.
[10] Li, W., 2021. Finding Needles in a Haystack: Recognizing Emotions Just from Your Heart. IEEE Transactions on Affective Computing, (01), pp.1-1.
[11] Chang F, Dean J, Ghemawat S, Hsieh WC, Wallach DA, Burrows M, Chandra T, Fikes A, Gruber RE. Bigtable: A distributed storage system for structured data. ACM Transactions on Computer Systems (TOCS). 2008 Jun 1;26(2):1-26.
[12] Cooper BF, Ramakrishnan R, Srivastava U, Silberstein A, Bohannon P, Jacobsen HA, Puz N, Weaver D, Yerneni R. PNUTS: Yahoo!'s hosted data serving platform. Proceedings of the VLDB Endowment. 2008 Aug 1;1(2):1277-88.
[13] Silberstein, A., Chen, J., Lomax, D., McMillan, B., Mortazavi, M., Narayan, P.P.S., Ramakrishnan, R. and Sears, R., 2011. Pnuts in flight: Web-scale data serving at yahoo. IEEE Internet Computing, 16(1), pp.13-23.
[14] Calder B, Wang J, Ogus A, Nilakantan N, Skjolsvold A, McKelvie S, Xu Y, Srivastav S, Wu J, Simitci H, Haridas J. Windows azure storage: a highly available cloud storage service with strong consistency. InProceedings of the Twenty-Third ACM Symposium on Operating Systems Principles 2011 Oct 23 (pp. 143-157).
[15] Copeland, M., Soh, J., Puca, A., Manning, M. and Gollob, D., 2015. Microsoft azure and cloud computing. In Microsoft Azure (pp. 3-26). Apress, Berkeley, CA.
[16] Kotas, C., Naughton, T. and Imam, N., 2018, January. A comparison of Amazon Web Services and Microsoft Azure cloud platforms for high performance computing. In 2018 IEEE International Conference on Consumer Electronics (ICCE) (pp. 1-4). IEEE.
[17] Ucuz, D., 2020, June. Comparison of the IoT Platform Vendors, Microsoft Azure, Amazon Web Services, and Google Cloud, from Users’ Perspectives. In 2020 8th International Symposium on Digital Forensics and Security (ISDFS) (pp. 1-4). IEEE.
[18] Yianilos PN, Sobti S. The evolving field of distributed storage. IEEE Internet Computing. 2001 Sep;5(5):35-9.
[19] Hong G. A Survey of Distributed Storage Systems, Supercomputing Frontiers and Innovations 2004, vol. 5, no. 1.
[20] Wei Xu and Mao Ye. Distributed Storage Systems for Structured Data - a Survey , 2007
[21] Ericson K, Pallickara S. Survey of storage and fault tolerance strategies used in cloud computing. In Handbook of Cloud Computing, Springer, Boston, MA. 2010 (pp. 137-158). .
[22] Placek M. and Buyya R. A taxonomy of distributed storage systems. Technical report 2006. University of Melbourne. Distributed Systems and Grid Computing Laboratory. doi=10.1.1.79.6708
[23] Ju, Jiehui, Jiyi Wu, Jianqing Fu and Zhijie Lin. “A Survey on Cloud Storage.” J. Comput. 6 (2011): 1764-1771.
[24] Liu, Allan and Yu, Ting, Overview of Cloud Storage And Architecture (December 12, 2018). INTERNATIONAL JOURNAL OF SCIENTIFIC & TECHNOLOGY RESEARCH, Available at SSRN: https://ssrn.com/abstract=3649074
[25] Mazumdar, S., Seybold, D., Kritikos, K. et al. A survey on data storage and placement methodologies for Cloud-Big Data ecosystem. J Big Data 6, 15 (2019). https://doi.org/10.1186/s40537-019-0178-3
[26] Leonard, B., Jia, H., Verstrepen, J., Lehtinen, J., Lauber, L., Jelinko, P. and Gucer, V., 2019. IBM Cloud Object Storage Concepts and Architecture System Edition. 2nd ed. IBM.
[27] Docs.voltdb.com. 2021. 1.3. How VoltDB Works. [online] Available at: <https://docs.voltdb.com/UsingVoltDB/IntroHowVoltDB Works.php> [Accessed 3 August 2021].
[28] S. Padhy and G. M. M. Kumaran, "A Quantitative Performance Analysis between Mongodb and Oracle NoSQL," 2019 6th International Conference on Computing for Sustainable Global Development (INDIACom), 2019, pp. 387-391.
[29] C. -M. Iurian, I. -A. Ivanciu and V. Dobrota, "Couchbase Server in Microsoft Azure Cloud: A Docker Container Approach," 2020 International Symposium on Electronics and Telecommunications (ISETC), 2020, pp. 1-4, doi: 10.1109/ISETC50328.2020.9301052.
[30] S. Kalid, A. Syed, A. Mohammad and M. N. Halgamuge, "Big-data NoSQL databases: A comparison and analysis of “Big-Table”, “DynamoDB”, and “Cassandra”," 2017 IEEE 2nd International Conference on Big Data Analysis (ICBDA), 2017, pp. 89-93, doi: 10.1109/ICBDA.2017.8078782.
[31] P. Wang, F. Xu, M. Ma and L. Duan, "Efficient Spatial Big Data Storage and Query in HBase," 2019 IEEE International Conference on Smart Cloud (SmartCloud), 2019, pp. 149-155, doi: 10.1109/SmartCloud.2019.00035.
[32] D. Gudu, M. Hardt and A. Streit, "Evaluating the performance and scalability of the Ceph distributed storage system," 2014 IEEE International Conference on Big Data (Big Data), 2014, pp. 177-182, doi: 10.1109/BigData.2014.7004229.
[33] X. Zhang, Y. Wang, Q. Wang and X. Zhao, "A New Approach to Double I/O Performance for Ceph Distributed File System in Cloud Computing," 2019 2nd International Conference on Data Intelligence and Security (ICDIS), 2019, pp. 68-75, doi: 10.1109/ICDIS.2019.00018.
[34] Ghani, A., Badshah, A., Jan, S., Alshdadi, A.A. and Daud, A., 2020. Issues and challenges in cloud storage architecture: a survey. arXiv preprint arXiv:2004.06809.
[35] Balaji SB, Krishnan MN, Vajha M, Ramkumar V, Sasidharan B, Kumar PV. Erasure coding for distributed

storage: An overview. Science China Information Sciences. 2018 Oct;61(10):1-45.
[36] Dimakis AG, Godfrey PB, Wainwright MJ, Ramchandran K. Network coding for distributed storage systems. InIEEE INFOCOM 2007-26th IEEE International Conference on Computer Communications 2007 May 6 (pp. 2000-2008). IEEE.
[37] Datta A, Oggier F. An overview of codes tailor-made for better repairability in networked distributed storage systems. Acm Sigact News. 2013 Mar 6;44(1):89-105.
[38] Kulkarni, V., 2021. 7 Definitions of Cloud Computing! – ESDS BLOG. [online] ESDS BLOG. Available at: <https://www.esds.co.in/blog/7-definitions-of-cloud-computing/> [Accessed 29 July 2021].
[39] Lip Phang, C., 2015. Web coding bible. 1st ed. Malaysia: IngramSpark.
[40] Yarger, R., Reese, G., King, T., Takami, Y. and Terada, M., 1999. MySQL & mSQL. O'Reilly and Associates, Inc.
[41] Shi, W., Cao, J., Zhang, Q., Li, Y. and Xu, L., 2016. Edge computing: Vision and challenges. IEEE internet of things journal, 3(5), pp.637-646.
[42] Satyanarayanan, M., 2017. The emergence of edge computing. Computer, 50(1), pp.30-39.
[43] Laghari, Asif Ali, Awais Khan Jumani, and Rashid Ali Laghari. "Review and State of Art of Fog Computing." Archives of Computational Methods in Engineering (2021): 1-13.
[44] Kumar, V., Laghari, A. A., Karim, S., Shakir, M., & Brohi, A. A. (2019). Comparison of fog computing & cloud computing. Int. J. Math. Sci. Comput, 1, 31-41.
[45] Fernando, N., Loke, S.W. and Rahayu, W., 2013. Mobile cloud computing: A survey. Future generation computer systems, 29(1), pp.84-106.
[46] Laghari, A. A., He, H., Khan, A., Kumar, N., & Kharel, R. (2018). Quality of experience framework for cloud computing (QoC). IEEE Access, 6, 64876-64890.
[47] Karim, Sajida, Hui He, Arif Hussain Magsi, and Rashid Ali Laghari. "Quality of service (QoS): measurements of image formats in social cloud computing." Multimedia Tools and Applications (2020): 1-26.
[48] Halepoto, I. A., Memon, M. S., & Parveen, S. (2017). Analysis of quality of experience frameworks for cloud computing. IJCSNS, 17(12), 228.
[49] Laghari, A. A., He, H., Karim, S., Shah, H. A., & Karn, N. K. (2017). Quality of experience assessment of video quality in social clouds. Wireless Communications and Mobile Computing, 2017.
[50] Laghari, Asif Ali, Hui He, Muhammad Shafiq, and Asiya Khan. "Assessing effect of Cloud distance on end user's Quality of Experience (QoE)." In 2016 2nd IEEE international conference on computer and communications (ICCC), pp. 500-505. IEEE, 2016.
[51] Brewer EA. Towards robust distributed systems. InPODC 2000 Jul 16 (Vol. 7, No. 10.1145, pp. 343477-343502).
[52] Westoby L. How Apache Cassandra™ Balances Consistency, Availability, and Performance [Internet]. How Apache Cassandra™ Balances Consistency, Availability, and Performance | Datastax. 2021 [cited 11 March 2021]. Available from: https://www.datastax.com/blog/how-apache-cassandratm-balances-consistency-availability-and-performance
[53] Reed IS, Solomon G. Polynomial codes over certain finite fields. Journal of the society for industrial and applied mathematics. 1960 Jun;8(2):300-4.
[54] Dimakis AG, Godfrey PB, Wainwright MJ, Ramchandran K. Network coding for distributed storage systems. InIEEE INFOCOM 2007-26th IEEE International Conference on Computer Communications 2007 May 6 (pp. 2000-2008). IEEE.
[55] Huang C, Simitci H, Xu Y, Ogus A, Calder B, Gopalan P, Li J, Yekhanin S. Erasure coding in windows azure storage. In2012 {USENIX} Annual Technical Conference ({USENIX}{ATC} 12) 2012 (pp. 15-26).
[56] Duminuco A, Biersack EW. Hierarchical codes: A flexible trade-off for erasure codes in peer-to-peer storage systems. Peer-to-peer Networking and Applications. 2010 Mar;3(1):52-66.
[57] Oggier F, Datta A. Self-repairing homomorphic codes for distributed storage systems. In2011 Proceedings IEEE INFOCOM 2011 Apr 10 (pp. 1215-1223). IEEE.
[58] Lin HY, Tzeng WG. A secure erasure code-based cloud storage system with secure data forwarding. IEEE transactions on parallel and distributed systems. 2011 Oct 6;23(6):995-1003.
[59] Wang C, Wang Q, Ren K, Cao N, Lou W. Toward secure and dependable storage services in cloud computing. IEEE transactions on Services Computing. 2011 May 12;5(2):220-32.
[60] Lin HY, Tzeng WG. A secure decentralized erasure code for distributed networked storage. IEEE transactions on Parallel and Distributed Systems. 2010 Feb 5;21(11):1586-94.
[61] Kubiatowicz J, Bindel D, Chen Y, Czerwinski S, Eaton P, Geels D, Gummadi R, Rhea S, Weatherspoon H, Weimer W, Wells C. Oceanstore: An architecture for global-scale persistent storage. ACM SIGOPS Operating Systems Review. 2000 Nov 12;34(5):190-201.
[62] Anthapadmanabhan NP, Soljanin E, Vishwanath S. Update-efficient codes for erasure correction. In2010 48th Annual Allerton Conference on Communication, Control, and Computing (Allerton) 2010 Sep (pp. 376-382). IEEE.
[63] Dong Y. Coop-U: A Cooperative Update Scheme for Erasure-Coded Storage Systems. Arabian Journal for Science and Engineering. 2018 Dec;43(12):7385-96..
[64] Pei X, Wang Y, Ma X, Xu F. Efficient in-place update with grouped and pipelined data transmission in erasure-coded storage systems. Future Generation Computer Systems. 2017 Apr 1;69:24-40.
[65] Harshan J, Datta A, Oggier F. Differential Erasure Codes for Efficient Archival of Versioned Data in Cloud Storage Systems. InTransactions on Large-Scale Data-and Knowledge-Centered Systems XXX 2016 (pp. 23-65). Springer, Berlin, Heidelberg.
[66] Hadoop 3.0. WebHDFS REST API [Internet]. Hadoop.apache.org. 2021 [cited 17 April 2021]. Available from: https://hadoop.apache.org/docs/r1.0.4/webhdfs.html
[67] Hadoop. Apache Hadoop 3.0.0 [Internet]. Hadoop.apache.org. 2021 [cited 14 April 2021]. Available from: https://hadoop.apache.org/docs/r3.0.0/
[68] Zhang Z, Deshpande A, Ma X, Thereska E, Narayanan D. Does erasure coding have a role to play in my data center?. Microsoft research. 2010. MSR-TR-2010, 52.
[69] Greenan KM, Long DD, Miller EL, Schwarz TJ, Wylie JJ. A Spin-Up Saved Is Energy Earned: Achieving Power-Efficient, Erasure-Coded Storage. InHotDep 2008.