Cloud Storage Reliability for Big Data Applications: A State of the Art Survey

Rekha Nachiappan [a], Bahman Javadi [a], Rodrigo Calherios [a], Kenan Matawie [a]

[a] *School of Computing, Engineering and Mathematics, Western Sydney University, Australia*

Abstract

Cloud storage systems are now mature enough to handle a massive volume of heterogeneous and rapidly changing data, which is known as Big Data. However, failures are inevitable in cloud storage systems as they are composed of large scale hardware components. Improving fault tolerance in cloud storage systems for Big Data applications is a significant challenge. Replication and Erasure coding are the most important data reliability techniques employed in cloud storage systems. Both techniques have their own trade-off in various parameters such as durability, availability, storage overhead, network bandwidth and traffic, energy consumption and recovery performance. This survey explores the challenges involved in employing both techniques in cloud storage systems for Big Data applications with respect to the aforementioned parameters. In this paper, we also introduce a conceptual hybrid technique to further improve reliability, latency, bandwidth usage, and storage efficiency of Big Data applications on cloud computing.

Keywords: Fault tolerance; Big Data applications; Cloud storage; Replication; Erasure coding; Data Reliability.

1. Introduction

In the contemporary society of Big Data, the data volume is growing faster than storage capacity (Gantz and Reinsel, 2012). Each week, Facebook requires extra 60TB of storage just for new photos (Beaver et al., 2010). YouTube users upload over 400 hours of video every minute and it requires 1 Petabyte of new storage every day (Baesens, 2014). According to the Inter-

national Data Corporation (IDC)'s sixth annual study, until 2020 the digital data will double every two years (Gantz and Reinsel, 2012). Cloud computing offers a cost-effective way to support Big Data and analytics applications that can drive business value (Groenfeldt, 2012). By 2020, approximately 40% of the data in the digital universe will be stored or processed by cloud computing providers (Gantz and Reinsel, 2012). Cloud storage provides reasonable scalability for storing Big Data and it helps to handle the steady growth of data variety, volume and velocity (Chen and Zhang, 2014).

As Cloud storage is built up on commodity servers and disk drives (Ford et al., 2010), it is subject to failures that can compromise operation of applications relying on it. For example, In 2009, Facebook temporarily lost over 10% of its stored photographs because of a hard drive failure (Gunawi et al., 2010). Amazon Simple Storage Service (S3) encountered a data corruption problem caused by a load balancer bug (Wang et al., 2015). Amazon Web Services (AWS) suffered major disruptions due to a DynamoDB failure (AWS, 2016). At Facebook, in a production cluster of 3000 nodes, it is typical to have 20 or more node failures (Sathiamoorthy et al., 2013). As failures are the norm in cloud storage systems, improving data reliability while maintaining the system performance during data recovery is one of the most important challenges in deploying Big Data applications on cloud computing.

Data failure in cloud storage is handled by various data redundancy techniques. The most common redundancy techniques are replication and erasure coding. Replication is a simple data redundancy mechanism. The same data is copied and stored in several locations on the storage systems. If the requested data is not available in one disk, it is served from the next available disk (Plank, 2013). Erasure coding is a more complex data redundancy mechanism. Parity data is created and stored along with the original data, such that if the requested data is not available, it can be reconstructed from parity data. Storage overheads for erasure coding is much smaller than for replication, hence it reduces the hardware needs for data storage and provides significant cost and energy savings in data centres (Huang et al., 2012). However, data reconstruction upon failure involves high reconstruction cost and network traffic.

This is the main reason why cloud service providers are interested in moving towards erasure coding to improve reliability and reduce operational cost of systems. Facebook increased storage efficiency from 2.1x to 3.6x using erasure coding with multiple Petabytes of savings (Muralidhar et al., 2014). Microsoft Azure reduced storage overhead from 3x to 1.33x using erasure

coding which provided over 50% cost savings (Huang et al., 2012).
A study on the Facebook warehouse cluster (Rashmi et al., 2013) revealed that more than 50 machine-unavailability events were triggered per day. Data reconstruction due to those unavailability events increases network traffic. Facebook implemented Reed-Solomon code to only 8% of the data. As this requires 10x more network repair overhead per bit compared to replication, it is estimated that if 50% of data were replaced with Reed-Solomon, repair network traffic might saturate the cluster network links (Sathiamoorthy et al., 2013).
Another issue with the use of error correction techniques is that repair traffic increases latency. Storage systems consume up to 40% of a data centre's total energy (Harnik et al., 2009) and energy efficiency of storage systems is influenced by read/write latency (Kumar et al., 2014). Hence reducing the latency involved in repair may conserve considerable amount of energy. As mentioned earlier, Erasure coding offers better storage efficiency, reliability,and availability, but reconstruction of lost data increases network traffic and latency.
This paper addresses ongoing research and challenges in improving data reliability of Big Data Applications in cloud computing using replication and erasure coding. As both techniques have their own advantages and disadvantages, we discuss how researchers are striving to bring the benefits of one technique to another. We also propose a hybrid technique in the form of an ensemble of replication and erasure coding to benefit from the strengths of both techniques. The proposed hybrid technique is intended to optimize the important parameters of cloud storage system and improve reliability and performance while reducing storage overhead.
The rest of this paper is organized as follows. In section 2, we discuss types of storage systems and file systems used in cloud storage for Big Data applications data failures and data reliability. In sections 3 and 4, we discuss the state of art and challenges involved in erasure coding and replication, respectively. In section 5 we present a comprehensive comparison between replication and erasure coding. In section 6, we present the state of art in cloud storage reliability for Big Data applications, challenges and conceptual architecture of proposed hybrid technique. Finally, in section 7 we conclude and provide a basis for future developments in this area of research.

2. Background

In this section, we briefly discuss types of storage systems and file systems used in cloud storage systems for Big Data applications. Following that, the analysis on data failures and data reliability is presented.

2.1. Cloud Storage and Big Data Applications

Cloud storage systems consist of a number of storage devices connected by the network. It is typically composed of Network Attached Storage (NAS) or Storage Area Network (SAN) type of distributed storage with the feature of storage virtualization (Zeng et al., 2009). Storage virtualization is the technique of abstracting the physical storage from applications and mapping the logical storage into physical storage. The network of storage devices can be treated as a single storage device and users can access information regardless of physical locations and storage modes.

Based on how the data is accessed and interfaced by the client, cloud storage systems can be classified as file storage, block storage, and object storage (Mesnier et al., 2003)

File Storage: In file storage, files are organized hierarchically. The information about the file is stored as a metadata in a storage system. The files can be accessed by specifying the path to the individual file. It provides the higher level of storage abstraction to applications and it enables secured data transfer among different platforms. It achieves good performance in Local Area Network (LAN) if the number of files and metadata are limited. File server maintains metadata and authorize I/O to share files among multiple clients. However, file server contention affects data retrieval performance.

Block Storage: In block storage, the file is divided in blocks and an address is assigned for each block. The application can access and combine the blocks with the block address. The storage applications keep the metadata and use it to share data. It does not have any fileserver to authorize I/O and clients can directly access storage devices using metadata. It offers good performance. However, it is not offering promising secure data transmissions.

Object Storage: In object storage, the file and metadata are encapsulated as an object and the object is assigned with an object ID. The object can be of any type and geographically distributed. Each object can be assigned with unique metadata such as the type of application object associated, level of protection, number of replication and geographic location. It offloads storage management from applications to storage devices. This enables secure

direct data access to clients using metadata. It provides excellent scalability to support Big Data applications. Object storage support efficient erasure coding technique in addition to replication.
The variety, volume and velocity characteristics of Big Data can be fitted well in the distributed, virtualized and scalable characteristics of cloud storage systems (Kune et al., 2016). OReilly (O'Reilly, 2016) discussed advantages and drawbacks of prominent Big Data file systems in detail. HDFS, GFS, Luster, ClusterFS, Ceph, OpenStack Swift, Quantcast and PVFS are examples of other file systems that support Big Data Applications. GFS and HDFS are widely employed in cloud storage and a comparison between those file systems are presented by Vijayakumari et al. (Vijayakumari et al., 2014).

2.2. Data Failures

In a cloud storage system, many factors can lead to data failure. Data failures also lead to cloud service failures. Sharma et al. (Sharma et al., 2016) presented a detailed survey of cloud service failures. The main causes of cloud data failures are hardware, software, network, and power failures (Rajasekharan, 2014). Disks are the central element in cloud based storage (Brewer et al., 2016) and are the most common failure component (Hughes et al., 2002). Vishwanath and Nagappan analysed hardware reliability for a large cloud computing infrastructure (Vishwanath and Nagappan, 2010). As shown in Figure 1(data collected from (Vishwanath and Nagappan, 2010)) 78% of all server failures were due to hard disks, 5% to Rapid Array of Inexpensive Disk (RAID) controller, 3% due to memory, with the remaining 14% due to other factors. Hard disks are the most commonly replaced component and they are the most frequent cause of server failures (Vishwanath and Nagappan, 2010).
As depicted in Figure 2, data failures can be transient or permanent. Data unavailability due to network outage, node/machine failure, power outage, and automated repair process are transient and do not lead to permanent data loss (Rajasekharan, 2014). Data gathered from tens of Google storage cells, each of which with 1000 to 7000 nodes over one year period, reveals that less than 10% of events had node unavailability with duration under 15 minutes (Ford et al., 2010). Data unavailability due to hard disk failure or data corruption leads to permanent data loss.
Pinheiro et al. (Pinheiro et al., 2007) present a detailed analysis of failure behaviour of large scale disk drives using monitored data collected over a period of nine months. They found failure probabilities to be highly cor-

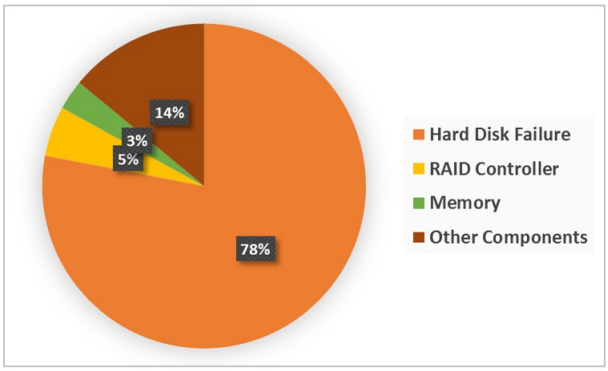

Figure 1: Causes of server failures in cloud computing systems (Vishwanath and Nagappan, 2010)

related with the drive first scan errors, reallocations, offline allocations and probation counts. Ford et al.(Ford et al., 2010) demonstrate the importance of modelling correlated failures on availability prediction. They show that failing to consider node failure results in overestimation of availability. Data availability increased 1.5% from reducing the disk failure rate by 10%. However, 10% reduction of node failure rate increases availability by 18%. Ma et al. (Ma et al., 2015) analysed disk failure from a large number of backup systems to show reallocated sectors and specific types of sector errors have large impact on disk reliability. They designed proactive protection against single and multiple disk failures.

Various component failures in cloud storage systems lead to permanent and transient data failures. Disks are the most important component to be considered in cloud storage systems. Disk failures lead to permanent data loss if they are not handled properly. Most of the other component failures in cloud storage systems cause temporary outages only. Some outages may last for hours, causing huge financial losses (Abu-Libdeh et al., 2010). The above discussions shed some light on considering the respective component failures to improve durability and availability. Next section discusses various data reliability mechanisms employed in cloud storage systems.

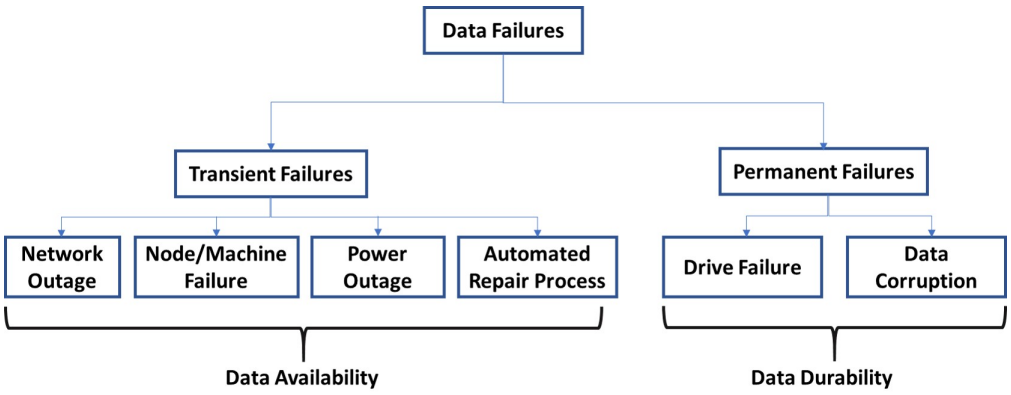

Figure 2: Data failures in cloud storage

2.3. Data Reliability

Data reliability includes maximizing durability and availability of data. Durability mitigates permanent failures and availability mitigates transient failures.

As shown in Figure 3, various mechanisms are used in cloud data centres to improve fault tolerance of the storage system. The impact of hardware failures is mitigated with RAID arrays, swappable drivers, and Error Correction Code Memory (ECC RAM). RAID arrays are a logical unit composed of several disks that stores data with striping, mirroring and parity. Swappable drivers allow administrators to swap drives that fail or predicted to fail while the system remains in operating mode. ECC RAM is used to detect and correct single bit errors by associating a parity bit with each binary code. Network failures and power outage are handled with network redundancy and dual power supply respectively.

Failures due to any issues including disasters in cloud storage are handled with erasure coding, replication and Resilient Distributed Dataset (RDD) (Zaharia et al., 2010). Replication and erasure coding are used to handle primary data failures. RDD is used to protect intermediate data generated by Big Data applications (Zaharia et al., 2010).

Erasure coding (Huang et al., 2012; Sathiamoorthy et al., 2013) and replication (Li et al., 2011) are the most popular reliability mechanisms employed on cloud storage. Figure 4 is a representation of replication and erasure coding techniques. In replication, data file/object are divided into chunks and stored several times on the storage systems. If the requested data is

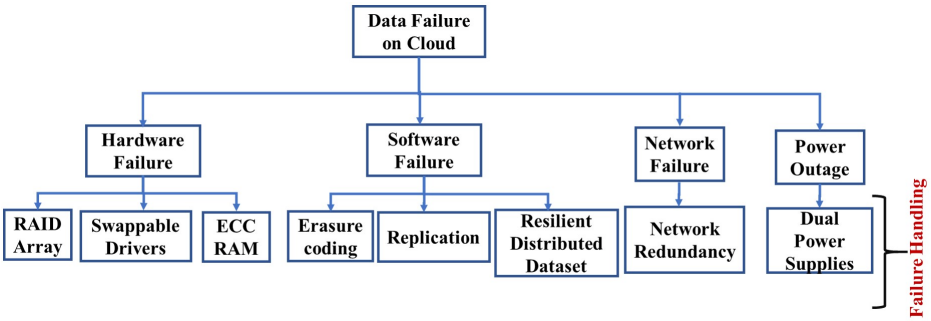

Figure 3: Failure Handling in Cloud Data Centres

not available in one disk, it is served from the next available disk (Plank, 2013). In erasure coding, data file/object is divided into chunks. Parity data is created and stored along with the original data, such that if the requested data is not available, it is reconstructed and served with the help of parity data.

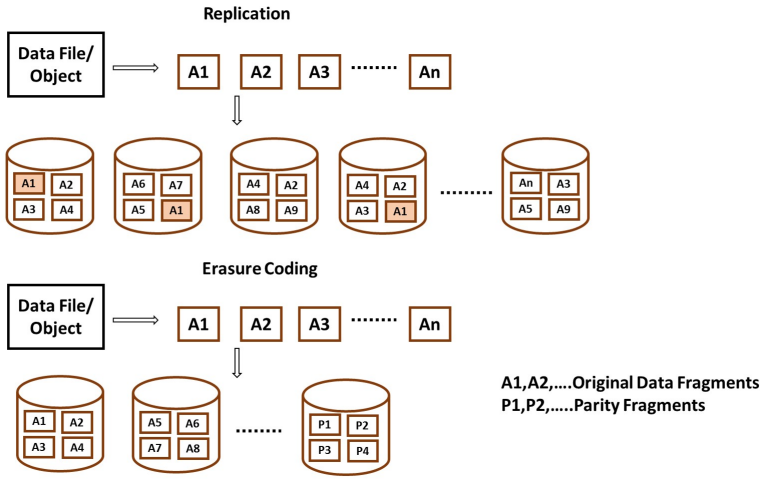

Figure 4: Replication and Erasure Coding

Even though cloud providers utilize various reliability techniques to improve fault tolerance against various component failures, replication and erasure coding stand out from all the others by its geographically distributed redundancy. Hence, replication and erasure coding support any kind of data

loss including disasters. The next two sections discuss erasure coding and replication.

3. Erasure Codes

Erasure coding is playing a predominant role in protecting data from failures in large scale storage systems (Plank, 2013). Before the emergence of cloud computing, erasure coding was used to detect and correct errors in storage and communication systems (Vins et al., 2014). In (n, k) erasure codes storage system, a file of size B will be divided into k equal chunks and $n-k$ parity chunks are added such that any k out of n chunks can restore the original file. For example, Figure 5 represents (4,2) erasure code which can tolerate any two failures. The arithmetic used to calculate parity data can be standard arithmetic or Galois Field arithmetic (Plank, 2013). In standard arithmetic, addition is carried out as binary XOR and multiplication as binary AND. Standard arithmetic is performed if the number of bits in a word is 1. When the number of bits in a word increases, parity is calculated using Galois Field arithmetic. In Galois Field, GF(2^n), arithmetic operations are bound within a finite set of numbers from 0 to 2^{n-1};addition is bitwise XOR and multiplication is more complex which depends on hardware, memory and number of bits in a word (Plank, 2013).

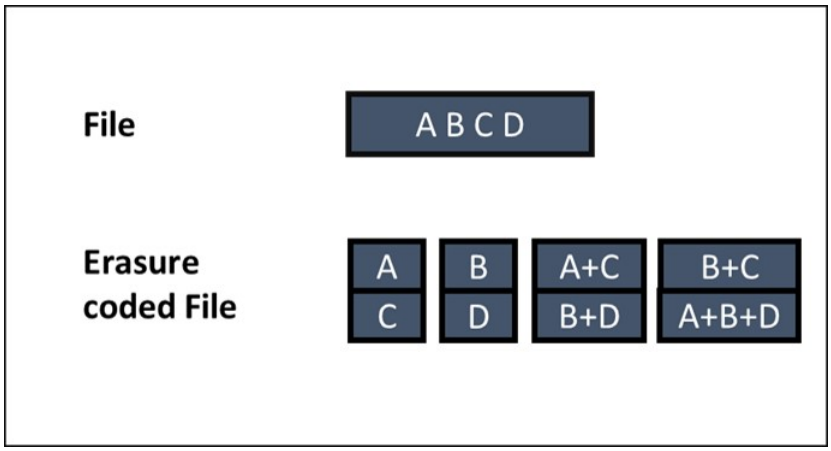

Figure 5: Erasure Coding

Erasure codes can be classified as Maximum Distance Separable (MDS) and non-MDS. The code is said to be MDS if m disks hold parity data and the

system tolerates any combination of m disk failures; non-MDS codes can tolerate only few combination of m disk failures, if m disks are dedicated to hold parity data. For example, in Figure 6.a, disks D5 and D6 are dedicated for parities, so this system can tolerate any two disk failures. This makes it MDS codes. In Figure 6.b, D5, D6 and D7 are dedicated for parities but it cannot tolerate any three disk failures. For example, if D1, D5 and D6 fail at the same time the data in D1 will not be recovered. This is known as non-MDS codes.

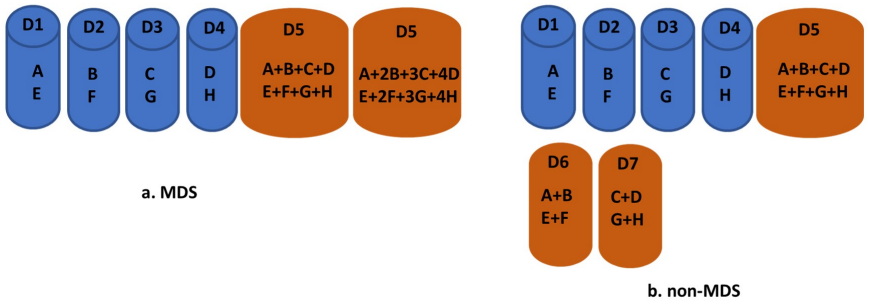

Figure 6: Different erasure coding types a. MDS codes b. non-MDS codes

Examples of simple erasure codes are RAID-6, Array codes, and Reed-Solomon codes (Plank, 2013). RAID-6 codes are MDS that creates two parity blocks for data blocks such that it can handle two disk failures (Jin et al., 2009). Array codes are implemented with standard arithmetic (i.e., XOR operations). In array codes parity is calculated as different linear combination of systematic data (original data). Row Diagonal Parity (RDP) (Xiang et al., 2010), EVENODD (Blaum et al., 1995), Blaum-Roth (Yixian, 1994) and Liberation codes (Plank, 2009) are array codes for RAID 6 that can tolerate up to any two disk failures. Star code is an array code and it can tolerate any combination of three disk failures (Huang and Xu, 2008). Cauchy Reed-Solomon, Generalized EVENODD and Generalized RDP are array codes that can be defined for any values of k and m (Plank, 2013). Recent advancements reduce CPU burden on Galois Field arithmetic for Reed-Solomon codes. Moreover, it is straightforward to define Reed-Solomon code for any values of k and m. Hence Reed Solomon has gained prominence over other erasure codes(Plank, 2013).

Reed-Solomon codes are the most popular erasure codes. They can be defined for any combination of data and parity disks. Reed-Solomon codes are MDS codes. Encoding and decoding can be done with Galois Field arithmetic. Facebook and Microsoft Azure implemented Reed-Solomon codes in their storage systems (Beaver et al., 2010; Huang et al., 2012). Any data failures in erasure coded storage systems trigger data reconstruction to serve the failed data. Since data reconstruction in erasure coding involves high disk I/O and network bandwidth it increases the cost of data reconstruction. Many contemporary research focus on reducing reconstruction costs of failed data on Reed-Solomon coded storage systems.
This paper highlights the recent works on two important categories. One is on reducing network bandwidth for reconstruction and these codes are called regeneration codes. The other is on reducing disk I/O needed for reconstruction of lost data and it is known as Locally Repairable codes (LRC). In the following sections, we discuss non-MDS/LRC codes and regeneration codes respectively.

3.1. Non-MDS Codes/Locally Repairable Codes

Non-MDS codes have local parities for original data blocks along with global parities in such a way that the reconstruction needs minimum disk I/O.

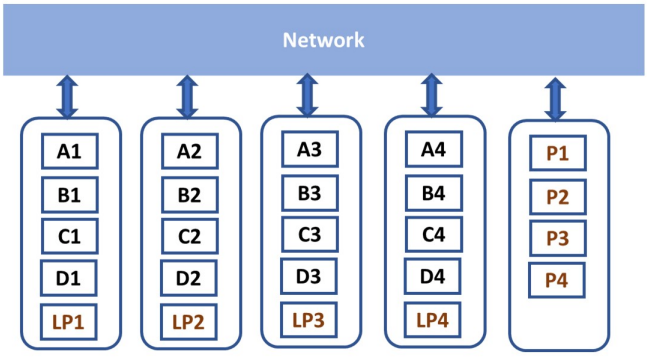

Figure 7: Locally Repairable Codes

Figure 7 represents locally repairable codes. Local parity helps blocks with single failures to be reconstructed with less number of data blocks than global

parity. Global parity can be utilized for reconstructing blocks with two or more simultaneous failures. Adding local parity makes the codes non-MDS and increases storage overhead.

Huang et al. (Huang et al., 2013) designed two new non-MDS erasure codes (Basic Pyramid Codes and Generalized Pyramid Codes). They designed Basic Pyramid Code from MDS codes. For example, Pyramid Code can be constructed from (11,8) MDS code as follows. Eight data blocks of (11,8) MDS codes should be separated into two equal size groups. Two out of three parity blocks can be kept unchanged and it is called global parities. Two new redundant blocks can be constructed from two equally separated data groups respectively and it is called local parities. This technique can significantly improve the read performance as local parities reduce the disk I/O involved in the reconstruction of lost data. Compared to (9,6) MDS code, (10,6) Basic Pyramid code reduces reconstruction read cost by 50%, with 11% additional storage overhead and 5.6x10-7 unrecoverable probability. Hence, it improves the performance of reconstruction with high fault tolerance and with additional storage overhead.

Generalized pyramid code is not an extension of Basic Pyramid code but it is defined with maximum recoverable (MR) property. Parity blocks for generalized pyramid code are calculated using a generator matrix. For erasure codes with MR property, the matching condition becomes sufficient i.e., all failure cases satisfying the matching condition are recoverable. Basic Pyramid code in comparison with generalized pyramid code provides 45% less unrecoverable property (Huang et al., 2013).

Following this work, Huang et al presented a new set of non-MDS erasure codes (Local Reconstruction Code) for Microsoft Azure Storage (Huang et al., 2012). This code is defined with (k, n, r) parameters. It divides k data fragments into n groups and generates n local parities for each group along with r global parity. It can tolerate up to $r+1$ failures and reduces the bandwidth and I/O traffic to reconstruct offline data fragments while has 1.33x more storage overhead compared to Reed Solomon codes. The average latency of decoding 4KB fragments is 13.2us for Reed-Solomon and 7.12us for LRC. Decoding is faster in LRC since the number of fragments needed for reconstruction is reduced to half.

Sathiamoorthy et al. (Sathiamoorthy et al., 2013) proposed a novel non-MDS erasure code (XORing the Elephants). They defined LRC (10,6,5) code on top of Facebook's RS (10,4) storage system by incorporating local parity. They further defined local parity for each 5 data blocks such that any single

lost data block can be reconstructed by only communicating with the remaining blocks in that group. It reduces approximately 2x on disk I/O and network traffic upon reconstruction, with 14% of storage overhead compared to Reed Solomon code.
Plank et al. (Plank et al., 2013) proposed Sector-Disk (SD) codes, which can tolerate a combination of disk and sector failures. It is a non-MDS code and can tolerate failure of any two disks and any two words in the stripe. It has minimum storage overhead compared to other non-MDS codes. They also noted that it needs less computation and disk I/O.
While all the above non-MDS codes improve the performance with better reliability, all impose additional storage overhead. Local parity is effective only for single block failures in the disk.

3.2. Regeneration Codes

Regeneration codes are defined for efficient repair of a failed nodes in terms of minimizing the amount of data downloaded for repair. Traditionally, a failed node data can be reconstructed by communicating and downloading the entire data with any k available nodes. Dimakis et al., (Dimakis et al., 2010) proved that the fraction of data from any d surviving nodes ($k \leq d \leq n-1$) are enough to reconstruct the failed node with network coding. (n, k) erasure coded storage system assumes that B is the size of the file and each fragment comprised of α symbols over a finite field. According to the definition of regeneration codes, any $\beta<\alpha$ symbols from any d surviving nodes are enough to repair the failed node. Hence the total amount of data $d\beta$ downloaded for repair purpose is smaller than the size of file B as shown in Figure 8 (Rashmi et al., 2011). Assume that each data block in the figure is 1GB. Upon failure, the reconstruction needs only 3 GB instead of 4 GB.

3.2.1. Minimum Storage and Minimum Bandwidth Regeneration Codes

Regenerating codes can be Minimum Storage Regenerating (MSR) or Minimum Bandwidth Regeneration (MBR). Minimizing α is known as Minimum Storage Regeneration. Minimizing β is known as Minimum Bandwidth Regeneration. In MSR, α and β can be decided by first minimizing α and then minimizing β. In MBR, α and β can be decided by first minimizing β and then minimizing α.
The repair process can be partial, functional or exact. In exact regeneration code, the replacement node stores exactly the same data as the failed

Table 1: Related work on reducing latency of erasure coded storage systems

Author	**Type of storage systems**	**Performance on Data Failure**	**Reliability**	**Energy Efficiency**	**Storage Overhead**
(Huang et al., 2013)	Cloud	Reduces number of blocks needed to reconstruct failed data	Tolerates any k-1 failures,and 86 % of k failures	NA	11% additional storage overhead
(Huang et al., 2012)	Windows Azure Storage	Reduces disk I/O and network traffic	Tolerates any k-1 failures, and 86% of k failures	NA	Approximately 1.6x of storage overhead
(Sathiamoorthy et al., 2013)	HDFS	Reduces approximately 2x on network traffic and disk I/O	Mean time to Data loss is high compared to Reed-Solomon code	NA	14% additional storage overhead
(Dimakis et al., 2010)	Distributed Storage	Improved performance in terms of network traffic	Improved Reliability	NA	No
(Pei et al., 2015)	Distributed Storage	Improved performance in terms of network traffic	Improved Reliability	NA	No
(Khan et al., 2012)	Cloud	Reduces the number of symbols for recovery and improve performance by 20%	Tolerates arbitrary k failures	NA	No
(Rashmi et al., 2014)	HDFSRAID in Facebook data warehouse	Reduces both network traffic and disk I/O around 25% to 45% compared to Reed-Solomon code	Tolerates arbitrary k failures	NA	No

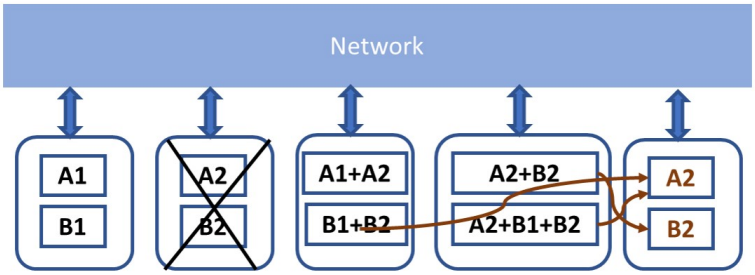

Figure 8: Regeneration Codes

node. Functional regeneration codes reconstruct a new node, which may contain different data from the corresponding failed node, although it should form an MDS code. In partial regeneration, original data nodes are repaired exactly and parity nodes are repaired functionally (Suh and Ramchandran, 2011).

Suh and Ramchandran (Suh and Ramchandran, 2011) proposed an exact MSR code where $d \geq 2k-1$ over finite field of size at least $2(n-k)$ with interference alignment property. Rashmi et al. (Rashmi et al., 2011) proposed optimal construction of an exact MBR code for all values of (n, k, m) and MSR codes for all $(n, k, d \geq 2k-2)$ using the new product-matrix framework with finite field of size at least $n(m-k+1)$. Various choices of parameters (n, k, m) for exact MSR codes have been defined in (Cadambe et al., 2010; Papailiopoulos and Dimakis, 2011; Suh and Ramchandran, 2010). Hybrid MSR codes with various choices of parameters have been defined in various works (Cadambe et al., 2011; Tamo et al., 2011; Wu, 2011), which support the exact repair of systematic parts and functional repair for parity part.

MSR and MBR codes focus on storage and bandwidth minimization but may increase disk I/O. Various choices of parameters for exact repair remain an open problem.

3.2.2. Repair-by-Transfer Regenerating Codes

In the regeneration of codes, the replacement of the failed node needs to be connected to the remaining nodes and will receive $\beta<\alpha$ data blocks which are the function of α symbols stored on it. The nodes helping in the

repair process read several data blocks and pass the function of the α data blocks stored in it. This process may lead to disk I/O overhead. In order to minimize I/O overhead and to avoid arithmetic operations performed by the providers, repair-by-transfer regenerating codes have been proposed.
Rashmi et al. (Rashmi et al., 2009) proposed an intuitive repair by transfer exact MBR codes for any $(n, k, d = n - 1)$. Functional repair is carried out by transfer MSR codes for different values of (n, k, d) defined in (Hu et al., 2013; Shum and Hu, 2012). Exact repair is carried out by transfer MBR codes $(n, k = n - 2, d = n - 1)$ over finite field of size 2 defined in (Shah et al., 2012). Lin and Chung (Lin and Chung, 2014) define a novel repair by transfer exact MBR codes at $m = n-1$ MBR points which demands a smaller finite field. Chen and Wang reveal the non-existence of a minimum storage regenerating (MSR) code with the repair-by-transfer property for $k \geq 3, \beta < d - k + 1$ (Chen and Wang, 2015).
Repair-by-transfer regenerating codes minimize disk I/O and also have all the benefits of MSR and MBR codes. However, there are only some specific choices of parameters.

3.2.3. Cooperative Recovery Regeneration Codes

Hu et al. (Hu et al., 2010) first proposed a mutually cooperative recovery (MCR) mechanism for multiple node failures. In this mechanism, nodes to be repaired can exchange data among themselves to provide better trade-off between storage and bandwidth. Cooperative regenerating code bound on bandwidth consumption of the new node is defined in (Kermarrec et al., 2011; Shum, 2011). Shum and Hu (Shum and Hu, 2011) propose an explicit construction of exact MBCR for $(n, k, d = k, t = n-k)$ where t is the number of new nodes communicated for the reconstruction. Wang and Zhang (Wang and Zhang, 2013) show that for all possible values of (n, k, d, t), there exists exact MBCR code on field size of at least n. Le Scouarnec (Le Scouarnec, 2012) explain the construction of exact MSCR code for some choices of parameter when $d \geq k = 2$. Pei et al. (Pei et al., 2015) propose cooperative regeneration repair based on the tree structure CTREE for multiple failures to optimize repair network traffic and time. They propose CExchange to reduce the network traffic cost. ED-TREE and PTransmission were proposed to reduce repair time and improve data transmission efficiency. All the above codes are limited to only some specific choices for the parameters.
The following works concentrated on optimizing the disk I/O needed for reconstruction and reducing I/O cost of recovery without any storage overhead

unlike non- MDS. This algorithm supports any XOR based erasure codes (i.e. array codes). Xiang et al. (Xiang et al., 2010) propose Row Diagonal Optimal Recovery (RDOR) for single disk failure in RDP codes and it significantly reduces I/O costs for recovery. He proposes I/O optimal recovery for single disk failure. Khan et al. (Khan et al., 2012) propose an algorithm to minimize the disk I/O needed for reconstruction based on symbols(partitions of block in each disk). This algorithm supports any number of parity blocks $\leq$ 3.
The following code is aiming at reducing bandwidth and disk I/O without any storage overhead. Rashmi et al. (Rashmi et al., 2014) propose Hitchhiker code, which is built on top of RS Code using a "Piggybacking" framework with Hop-and-couple (disk layout). It supports any choice of systematic and parity data fragments. While Hitchhiker reduces the time required for reading data during reconstruction by 32% and reduces the computation time during reconstruction by 36% with 35% reduction in network traffic and disk I/O, it increases the encoding time.

4. Replication

Replication is the most common reliability mechanism used in cloud data centres to improve availability and durability with low latency and minimum bandwidth consumption (Bonvin et al., 2009). Upon failure, in order to maintain the durability, the failed replica needs to be restored in the active disk. This restoration can be performed either reactively or proactively. In reactive replication, the replica will be created after the failure. In proactive replication, the replica will be created even before the occurrence of failure. Common approaches used in replication are static and dynamic replication.

4.1. Static Replication

In static replication, the number of replicas and their locations are fixed (Bonvin et al., 2009). Replicas are created and managed manually regardless of the changes in user behaviour. Random replication is the most common replication technique used in HDFS, RAMCloud, GFS and Windows Azure. In this technique, data are replicated on randomly selected nodes on different racks. Random replication can tolerate concurrent failure as the chunks are placed on different racks. However, it is ineffective when all the replicas are lost. Also, fixing lost data involves high cost associated with locating and recovering the lost data. Cidon et al (Cidon et al., 2013) propose Copyset

replication. It splits the nodes into copysets with respect to number of replications, which corresponds to random permutation. Replicas are placed in one of the copysets. Data loss only occurs if all the nodes of some copyset fail concurrently. It increases the data durability without significant storage overhead and with the same performance as random replication.
Liu and Shen (Liu and Shen, 2016) proposed Multi-Failure Resilient Replication (MRR) to improve availability in cloud storage. Authors define different number of replication for each object based on the popularity of the object. Nodes are separated into different groups such that groups can handle different number of replications and each set consists of the nodes from different data centres. It reduces the probability of data loss with low consistency maintenance cost. Long et al (Long et al., 2014) proposed the Multiobjective Optimized Management (MOM) algorithm for cloud storage. MOM decides the number of replicas and location of replicas based on a mathematical model with five objectives, namely unavailability, service time, load variance, energy consumption and latency. The parameters size, access rate of the file, failure probability, transfer rate and capacity data node have been considered in the definition of the model. Authors show that this algorithm increases file availability, load balancing and decrease service time, latency and energy consumption.

4.2. Dynamic Replication

In dynamic replication, replicas are created and removed dynamically. Replica creation, location, management and deletion are handled automatically according to the user behaviour in order to improve durability, availability, cost, storage efficiency, bandwidth, latency, energy and execution time. Bonvin et al. (Bonvin et al., 2009) proposed a dynamic cost efficient replication in clouds with consideration of geographical diversity while maintaining high availability and low latency. He proposed Skute, a key-value store which determines cost efficient position of replicas. Sun et al. (Sun et al., 2012) defined a mathematical model of relationship between system availability and number of replicas. They proposed dynamic replication strategy that determines which data to replicate, time of replication, number of replicas, and location of the new replicas to improve read performance and availability. Qu and Xiong (Qu and Xiong, 2012) proposed resilient, fault-tolerant and high efficient algorithm to achieve high availability in cloud storage systems. It dynamically balances workload among the nodes by considering the traffic load, node storage capacity and bandwidth for replica. Hence it increases

Table 2: Related work on improving reliability, cost and efficiency of replicated storage system

Author	**Type of storage systems**	**Replication type**	**Objective**
(Cidon et al., 2013)	Cloud	Static	To reduce probability of data loss without any additional storage overhead and performance lag
(Liu and Shen, 2016)	Cloud	Static	To improve availability with low storage and maintenance cost
(Long et al., 2014)	Cloud	Static	To improve availability with high performance and energy efficiency.
(Bonvin et al., 2009)	Cloud	Dynamic	To improve availability guarantee at minimum cost
(Sun et al., 2012)	Cloud	Dynamic	To improve performance and availability with high storage efficiency
(Qu and Xiong, 2012)	Cloud	Dynamic	To improve availability with high storage efficiency
(Hussein and Mousa, 2012)	Cloud	Dynamic	To improve reliability with minimum cost.
(Boru et al., 2015)	Cloud	Dynamic	To minimize network and energy efficiency.
(Li et al., 2011)	Cloud	Dynamic	To maintain reliability with low storage overhead.

data availability with high storage efficiency.
Hussein and Mousa (Hussein and Mousa, 2012) also proposed dynamic replication strategy. Based on the history of data requests and time series technique, it predicts future data access frequency. If the predicted frequency exceeds the threshold, then data chunks are selected for replication. After that, the number of replicas and location of the replicas are decided. Experimental results show that this strategy keeps response time stable regardless of the high number of tasks and improves reliability. Boru et al. (Boru et al., 2015)propose a data replication technique to optimize energy consumption, network bandwidth and communication delay in cloud data centres. They defined models for energy consumption and bandwidth demand and propose an energy efficient replication strategy based on this model that reduces communication delays. Li et al. (Li et al., 2011) proposed cost effective replication of Big Data applications on cloud storage, defined as a generic data reliability model in cloud based on replication. They used an algorithm for determining the minimum number of replicas with assurance of data reliability. In order to assure data reliability with minimum replication, a generic data reliability model has been utilized to predict data reliability. Data reliability has been maintained across the period using a proactive replication algorithm that detects replica loss and triggers the data recovery process if needed.

5. Comparison Between Replication and Erasure Coding

Replication and erasure coding are important reliability mechanisms used in cloud data centres to protect data against failure. It is important to understand the advantages and pitfalls of those techniques to implement an optimal technique in cloud storage systems to improve reliability with significant savings. The analysis of those technique with respect to various parameters are detailed below.
Figure 9 shows how a read request to unavailable data is handled in a replication and an erasure coded storage system. It also shows how the data is reconstructed in case of transient and permanent data failure. A request to unavailable data in a replicated storage system is served by simply redirecting the request to the next available replica. On the other hand, in an erasure coded storage system, temporarily unavailable data is served by reconstructing data from the next k available disk on the fly. Reconstruction in erasure coded storage involves more disk I/O than in replicated storage. For example, in Figure 9.c reconstruction of block A involves two blocks of

data read from two different disks. This increases the latency of the read request in an erasure coded storage system in comparison to replication.
Disk reconstruction upon permanent failure in an erasure coded system involves more disk I/O than replication. For example, in Figure 9.b the reconstruction of a failed disk involves only three disk access to reconstruct three data fragments. However, in Figure 9.d reconstruction of the failed disk involves four disk access to reconstruct two fragments. This increases the cost of reconstruction in an erasure coding system.
Figure 10 depicts storage overhead and Figure 11 depicts the reliability in terms of Mean Time to Failure (MTTF) in years with correlated failure for both redundancy policies. Data from (Ford et al., 2010) are used to depict Figure 10 and Figure11. These figures show that erasure codes provide better reliability with low storage overheads compared to replication. In large scale storage systems, replacing replication with erasure coding leads to significant cost savings.
Erasure coding is more storage efficient than replication, however there is a performance trade off (Cook et al., 2014). Encoding data in an erasure coded storage system is time consuming, while a request to the failed object can be redirected to the next available replica in a replicated system with no latency (Cook et al., 2014). In an erasure coded system, the failed object should be reconstructed from the next available objects, which increases the latency for the read request (Cook et al., 2014). Moreover, costs associated with reconstruction is high in terms of bandwidth and disk I/O (Li and Li, 2013).
Several works compare replication and erasure coding (Weatherspoon et al., 2002; Lin et al., 2004; Rodrigues and Liskov, 2005; Cook et al., 2014; Ford et al., 2010). These comparisons assume independence between parameters. For example,Erasure coding provides significant storage efficiency compared to replication or it increases durability and availability. However, Erasure coding can not significantly increase both storage efficiency and reliability. We summarized the comparison in Table 3. The keywords high and low are used to represent the superiority of one technique over other.

6. State of the Art in Cloud Storage Reliability for Big Data Applications

As failures are frequent in cloud storage system, data redundancy is employed in cloud storage systems to handle failures. Replication is simple

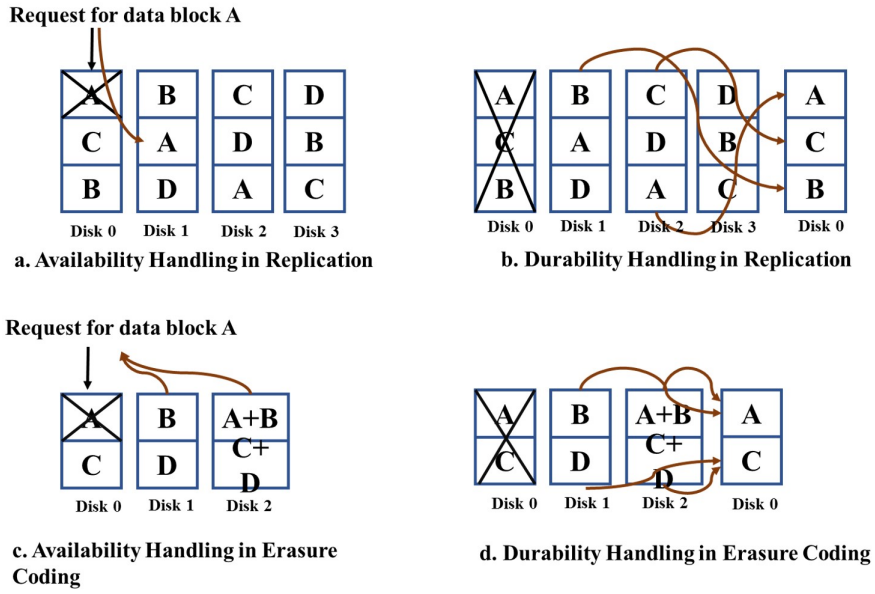

Figure 9: Durability and availability handling in replication and erasure coding techniques

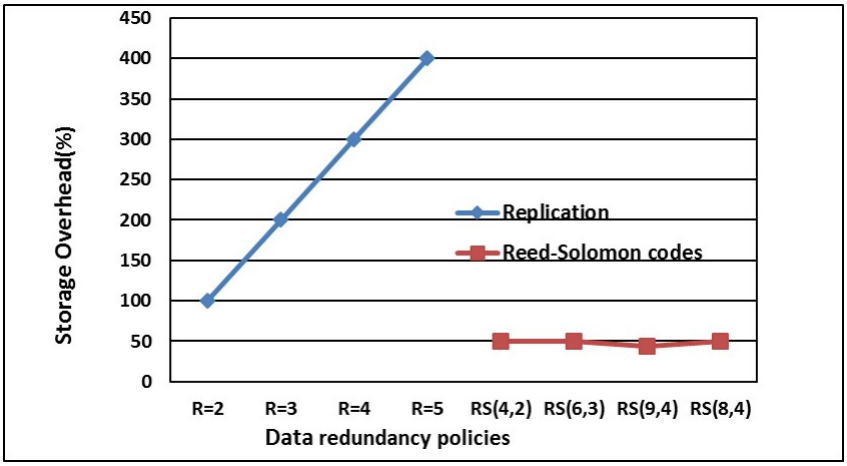

Figure 10: Storage overhead (percentage) of various redundancy policies (Ford et al., 2010)

solution to improve data reliability. But replicating terabytes and petabytes of data increase the storage overhead drastically. Nowadays erasure coding is gaining traction because it offers huge savings in terms of storage with

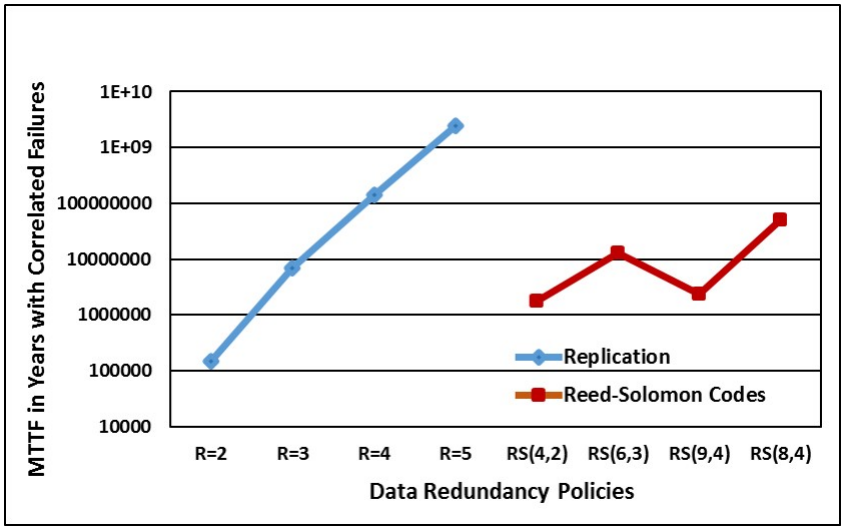

Figure 11: MTTF in years with correlated failures for various redundancy policies (Ford et al., 2010)

Table 3: Comparison between replication and erasure coding

Parameters	**Replication**	**Erasure coding**
Storage Overhead	High	Low
Availability	Low	High
Durability	Low	High
Latency on Failure	Low	High
Cost of Reconstruction	Low	High
Encoding & Decoding Complexity	Low	High

extensive reliability and durability assurance. However, reconstruction cost involved in recovering the lost data balances the storage savings. Reed-Solomon code requires approximately ten times more repair overhead per bit compared to replication. The challenges involved in employing the redundancy techniques for Big Data applications in cloud storage systems are discussed in the rest of this section.

Several studies (Table 1) have focused on reducing network traffic and reducing the disk I/O associated with reconstruction of failed data in erasure coded storage systems. Few works dedicate extra storage overhead to improving performance of erasure coded storage systems but none could make performance of erasure codes like the performance achieved with replication

Table 4: Related work on improving reliability, cost and efficiency of replicated storage system

Author	**Type of storage systems**	**Objective**	**Method**
(Araujo et al., 2011)	Distributed storage	Reduce latency (In terms of network bandwidth and disk I/O)	One full-replica of data is kept in one peer and erasure coded fragments are spread 22 in the network.
(Ma et al., 2013)	Cloud	Reduce latency (In terms of network bandwidth and disk I/O)	Cache the whole file upon write requests for serving the subsequent read and write requests.
(Li et al., 2016)	Cloud	Reduce latency (In terms of network bandwidth and disk I/O)	Adjust replication factor of data based on drive failure prediction.

for Big Data applications.
Some studies (Table 2) have focused on minimizing the number of redundancies in replicated storage systems to improve storage efficiencies. None could significantly reduce the storage overhead in comparison to erasure coding without sacrificing reliability. Achieving reliability, storage efficiency and performance together with either replicated or erasure coded storage systems has not yet been achieved.

Hybrid reliability mechanisms could be the choice of future data centres. Hybrid reliability mechanism combines replication and erasure coding. There are very limited works in hybrid reliability mechanisms, which are listed in Table 4. Araujo et al. (Araujo et al., 2011) proposed double coding based on hybrid coding. The idea here is to keep one full-replica of data in one peer and erasure coded fragments spread in the network. In double coding, the copy of original data fragments and parity fragments are arranged in different peers in the network. Even though it saves bandwidth upon reconstruction it affects storage efficiency. Ma et al. (Ma et al., 2013) proposed a novel scheme named CAROM, an ensemble of replication and erasure coding. Their approach caches the whole file upon write requests for serving the subsequent read and write requests. It also caches the requested block upon read request in order to serve subsequent reads. It saves storage cost by up to 60% and erasure coded bandwidth cost by up to 43% while keeping

the latency, as in replication. When the requested data is not in memory, however, it needs to reconstruct the data upon block unavailability. Li et al., (Li et al., 2016) presents proactive erasure coding (ProCode), which automatically adjusts replication factor of data based on drive failure prediction. It reduces degraded read latency by 63% and reconstruction time by 78%. This ProCode has no effect in the storage system consisting of flash drive and swappable drivers can handle the drive failures more efficiently.

6.1. Research Challenges

There are many open challenges in the field of Data Reliability of Big Data applications in cloud storage systems. In this section, we examine the challenges of improving data reliability of Big Data applications in cloud storage systems.

6.1.1. Storage Efficiency

Data reliability of replication is directly proportional to storage overhead. Hence improving storage overhead without sacrificing reliability is the greatest challenge in replication. Even though erasure coding offers tremendous storage savings with fair reliability, it increases network traffic and latency of Big Data applications in the presence of failure. This survey addressed various researches in replication that tried to conserve storage with dynamic replication strategy but it pawns reliability. It also showed that various researches in erasure codes focused on reducing network traffic and latency. Enabling automation of dynamic redundancy, such as changing number of replicas in erasure coded data to support failures and usage spikes could also improve storage efficiency further.

6.1.2. Bandwidth Efficiency

Network bandwidth is always a scarce resource in the distributed storage. Bandwidth usage is directly proportional to the amount of data transferred in the distributed storage. In both replicated and erasure coded storage system, fixing the failed data consumes considerable amount of network bandwidth. Traditional erasure codes involve more bandwidth consumption than replication. Node failures are events that trigger data reconstruction in cloud data centres and increase network traffic in erasure coded storage system (Sathiamoorthy et al., 2013). Various works towards reducing the network traffic in erasure coded storage system has been highlighted. However, those could not reduce network traffic as good as replicated storage system. Even

after recent improvements of erasure coding, network traffic involved in reconstruction is one of the most important challenges. Improving bandwidth efficiency without sacrificing performance and energy efficiency is one of the most important challenges. Failure prediction techniques should be improved to spot data failures in cloud storage systems. Dynamic replication of failure predicted data even before the failure occurs could significantly reduce the number of reconstructions in erasure coded storage and thereby reduces the network traffic.

6.1.3. Energy Efficiency

Storage systems are one of the most important energy consuming components in cloud computing (Sharma et al., 2016). Energy efficiency methods used in data centres save operational costs and help to conserve the environment (Butt et al., 2014). The energy efficiency of storage systems is highly dependent on read/write latency (Kumar et al., 2014). Pinheiro et al. (Pinheiro et al., 2006) introduced a technique called diverted access technique that separates original and redundant data on different disks in storage systems. This technique keeps disks containing redundant data in an idle state until there is a high disk failure. This technique has been proven to save 20-61% of disk energy. Harnik et al. (Harnik et al., 2009) proposed a method for full coverage in low power mode using auxiliary nodes (pool of extra nodes with additional copies of data) of any placement function. The power saving potential for an erasure coded storage system is limited in low power mode however it improves when the ratio between n and k grows. Butt et al. (Butt et al., 2014) presented an Energy Reliability Product (ERP) metric to compare different designs with respect to energy efficiency and reliability of data centre storage systems. Greenan et al. (Greenan et al., 2008) proposed power aware coding and present a generic technique for reading, writing and activating devices in a power aware erasure coded storage system. They also showed that activating the inactive disk increases power consumption. Li et al. (Li et al., 2011) proposed a link rate controlled data transfer (LRCDT) strategy for energy efficient data transfer in replication based cloud storage systems.

Energy savings in terms of storage in erasure codes is dedicated to reconstruction of data and it is vice versa for replication. Hence improving the reconstruction of data without sacrificing storage efficiency is a challenge. Reducing number of reconstructions in erasure code could reduce network traffic and improve energy efficiency and reducing number of reconstructions

can be achieved by seasonal replication of selected data blocks.

6.1.4. Data Access Latency

In erasure coded storage system when there is a failure, decoding must be performed to reconstruct the original data. Hence data access latency is one of the most important challenges in erasure coded storage system. In replicated storage systems, the access latency could be significantly reduced by choosing optimal location of replication. Energy and performance of the Big Data applications can be considerably improved with reduced access latency. Reducing the latency with less storage overhead is a challenge for the researchers. By activating proactive, dynamic replication based on the access history and failure logs in erasure codes, could help to reduce access latency with less storage overhead.

6.2. Conceptual Architecture

We propose the following hybrid technique for Big Data applications in cloud storage systems to reduce the network traffic and improve the performance with less storage overhead. This hybrid technique applies proactive dynamic data replication of erasure coded data based on node failure prediction. This hybrid technique will significantly reduce network traffic and will improve the performance of Big Data applications with less storage overhead. The conceptual architecture of the hybrid technique to improve reliability, performance with less storage overhead is depicted in Figure 12.
Cloud System Management is the centre point of the proposed architecture. All the management regarding application management and data reliability decisions will be made here. It is divided into two components: Big Data Application Management and Data Reliability Management.
Big Data Application Management is responsible for provisioning resources for applications, monitoring the health of the application, and shutting down and cleaning up of resources and billing.
Data Reliability Management is responsible for improving data reliability, performance and reducing network traffic of Big Data applications. It is divided into two components as follows:

- *Erasure Coding Management:* This component has both encoder and decoder. The encoder will partition the data into original and parity fragments based on the adapted erasure code. The decoder will reconstruct the missing data.

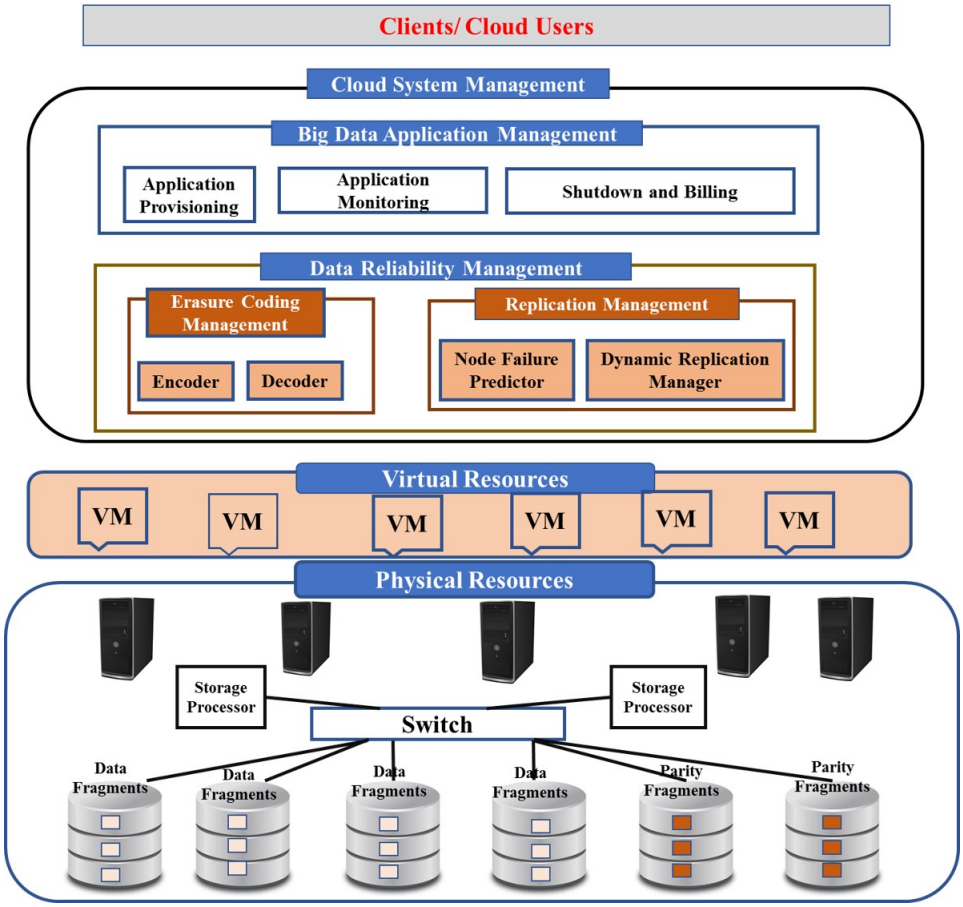

Figure 12: Reliability management of Big Data applications on cloud computing: conceptual architecture

- *Replication Management:* This is the most important component of this conceptual architecture. This is the component we propose to proactively handle failures of erasure coded data based on node access history and failure history. Each provisioned node in the cloud storage system runs a daemon for node failure prediction based on node failure history. Node Failure Predictor component periodically collects node′s failure status(Agrawal et al., 2015). When the status shows that the node predicted to fail, the data access history of that node will be examined. Based on the node access history the access pattern of the data blocks could be derived and the data blocks which are likely to be accessed soon can be predicted (Dai et al., 2014). These data blocks in failure predicted node should be proactively replicated into the best, next available node by Dynamic Replication Manager. This will decrease the need for reconstruction. In turn, it will reduce the network traffic, improve the reliability and performance of Big Data applications. It will also increase storage efficiency as it replicates the data only if failure is predicted and it is likely to be accessed soon. The unpredicted failure data blocks will be served with typical reconstruction of decoder.

6.3. Future Research Directions

In the era of Big Data, improving fault tolerance against data failure brings various challenges, such as how to reduce storage overhead; how to reduce network bandwidth and/or network traffic; how to minimize latency; and finally how to minimize energy consumption. To address these issues, the following research directions are listed for future.

- *New techniques to reduce storage overhead in replication, without sacrificing durability and availability of data:* Techniques that apply proactive failure handling and dynamic replication should be developed to reduce storage overhead of replication without sacrificing the reliability of Big Data applications.

- *Big Data-optimized erasure coding:* Erasure Coding helps to reduce storage overhead while maintaining durability and availability. However, this comes at the cost of increased network traffic, high disk I/O and large latency. Therefore, a novel erasure coding is needed to reduce the network traffic, disk I/O and latency while reducing storage overhead for Big Data.

- *Hybrid techniques to improve data reliability:* Since both replication and erasure coding have advantages and limitations, improving reliability with hybrid techniques, that can leverage the best aspects of each technique is a promising research topic. In this direction, a conceptual architecture of a hybrid technique is proposed in the following section.

- *Dynamic replication of fragments based on prediction:* Although erasure coding reduces energy consumption of storage systems, this is offset by the extra energy needed to reconstruct data when failures occur. Therefore, if an algorithm can predict items that are more likely to become unavailable due to failures, energy could be saved if the predicted items are proactively replicated, because in this case reconstruction can be avoided.

In addition, further consideration is needed for:

- Methods to optimize configuration parameters for both replication and erasure coding according to users reliability requirements.

- Geographical diversity of cloud storage while defining data reliability for Big Data applications.

7. Conclusions

Cloud computing is playing a predominant role to serve Big Data applications as it provides cost-effective on demand services. Cloud computing enables storage and computing resources to be scaled up and down rapidly in accordance with the demand. As failures are becoming the norm in cloud storage systems various fault tolerant mechanisms has been employed in cloud storage systems to improve data reliability. As replication involves huge storage overhead, erasure coding gains traction in cloud storage systems for Big Data applications. However, in case of failure, reconstructing lost data involves lots of resources which affects performance of the applications. This prevents cloud storage systems to move towards erasure coding. In this paper, we discussed the state-of-art of both techniques. In erasure coded storage systems, various techniques are proposed to reduce the resources involved in reconstruction. In replicated storage system improving data reliability with minimum replications are proposed. While existing hybrid techniques are addressed in this paper, a novel hybrid technique based on dynamic replication

in erasure coded storage systems is proposed. The proposed technique and the conceptual architecture can effectively handle the reconstruction issues of erasure code proactively with less storage overhead and improved reliability and energy consumption.

References

Abu-Libdeh, H., Princehouse, L., Weatherspoon, H., 2010. Racs: a case for cloud storage diversity. In: Proceedings of the 1st ACM symposium on Cloud computing. ACM, pp. 229–240.

Agrawal, B., Wiktorski, T., Rong, C., 2015. Analyzing and predicting failure in hadoop clusters using distributed hidden markov model. In: International Conference on Cloud Computing and Big Data in Asia. Springer, pp. 232–246.

Araujo, J., Giroire, F., Monteiro, J., 2011. Hybrid approaches for distributed storage systems. Data Management in Grid and Peer-to-Peer Systems, 1–12.

AWS, 2016. Summary of the amazon dynamodb service disruption and related impacts in the useastregion.
URL `https:aws.amazon.commessage5467D2`

Baesens, B., 2014. Analytics in a big data world: The essential guide to data science and its applications. John Wiley & Sons.

Beaver, D., Kumar, S., Li, H. C., Sobel, J., Vajgel, P., et al., 2010. Finding a needle in haystack: Facebook's photo storage. In: OSDI. Vol. 10. pp. 1–8.

Blaum, M., Brady, J., Bruck, J., Menon, J., 1995. Evenodd: An efficient scheme for tolerating double disk failures in raid architectures. IEEE Transactions on computers 44 (2), 192–202.

Bonvin, N., Papaioannou, T. G., Aberer, K., 2009. Dynamic cost-efficient replication in data clouds. In: Proceedings of the 1st Workshop on Automated Control for Datacenters and Clouds. ACM, pp. 49–56.

Boru, D., Kliazovich, D., Granelli, F., Bouvry, P., Zomaya, A. Y., 2015. Energy-efficient data replication in cloud computing datacenters. Cluster computing 18 (1), 385–402.

Brewer, E., Ying, L., Greenfield, L., Cypher, R., Tso, T., 2016. Disks for data centers. White paper for FAST 1 (1), p4.

Butt, A. R., Bhattacharjee, P., Wang, G., Gniady, C., 2014. Exploring trade-offs between energy savings and reliability in storage systems. The Green Computing Book: Tackling Energy Efficiency at Large Scale, 149.

Cadambe, V. R., Huang, C., Jafar, S. A., Li, J., 2011. Optimal repair of mds codes in distributed storage via subspace interference alignment. arXiv preprint arXiv:1106.1250.

Cadambe, V. R., Jafar, S. A., Maleki, H., 2010. Distributed data storage with minimum storage regenerating codes-exact and functional repair are asymptotically equally efficient. arXiv preprint arXiv:1004.4299.

Chen, C. P., Zhang, C.-Y., 2014. Data-intensive applications, challenges, techniques and technologies: A survey on big data. Information Sciences 275, 314–347.

Chen, Y., Wang, Y., 2015. On the non-existence of minimum storage regenerating codes with repair-by-transfer property. IEEE Communications Letters 19 (12), 2070–2073.

Cidon, A., Rumble, S. M., Stutsman, R., Katti, S., Ousterhout, J. K., Rosenblum, M., 2013. Copysets: Reducing the frequency of data loss in cloud storage. In: Usenix Annual Technical Conference. pp. 37–48.

Cook, J. D., Primme, R., de Kwant, A., 2014. Compare cost and performance of replication and erasure coding. hitachi Review 63, 304.

Dai, D., Chen, Y., Kimpe, D., Ross, R., 2014. Provenance-based object storage prediction scheme for scientific big data applications. In: Big Data (Big Data), 2014 IEEE International Conference on. IEEE, pp. 271–280.

Dimakis, A. G., Godfrey, P. B., Wu, Y., Wainwright, M. J., Ramchandran, K., 2010. Network coding for distributed storage systems. IEEE Transactions on Information Theory 56 (9), 4539–4551.

Ford, D., Labelle, F., Popovici, F. I., Stokely, M., Truong, V.-A., Barroso, L., Grimes, C., Quinlan, S., 2010. Availability in globally distributed storage systems. In: Osdi. Vol. 10. pp. 1–7.

Gantz, J., Reinsel, D., 2012. The digital universe in 2020: Big data, bigger digital shadows, and biggest growth in the far east. IDC iView: IDC Analyze the future 2007 (2012), 1–16.

Greenan, K. M., Long, D. D., Miller, E. L., Schwarz, T. J., Wylie, J. J., 2008. A spin-up saved is energy earned: Achieving power-efficient, erasure-coded storage. In: HotDep.

Groenfeldt, T., 2012. Big databig money says it is a paradigm buster.

Gunawi, H. S., Do, T., Joshi, P., Hellerstein, J. M., Arpaci-Dusseau, A. C., Arpaci-Dusseau, R. H., Sen, K., 2010. Towards automatically checking thousands of failures with micro-specifications. In: HotDep.

Harnik, D., Naor, D., Segall, I., 2009. Low power mode in cloud storage systems. In: Parallel & Distributed Processing, 2009. IPDPS 2009. IEEE International Symposium on. IEEE, pp. 1–8.

Hu, Y., Lee, P. P., Shum, K. W., 2013. Analysis and construction of functional regenerating codes with uncoded repair for distributed storage systems. In: INFOCOM, 2013 Proceedings IEEE. IEEE, pp. 2355–2363.

Hu, Y., Xu, Y., Wang, X., Zhan, C., Li, P., 2010. Cooperative recovery of distributed storage systems from multiple losses with network coding. IEEE Journal on Selected Areas in Communications 28 (2).

Huang, C., Chen, M., Li, J., 2013. Pyramid codes: Flexible schemes to trade space for access efficiency in reliable data storage systems. ACM Transactions on Storage (TOS) 9 (1), 3.

Huang, C., Simitci, H., Xu, Y., Ogus, A., Calder, B., Gopalan, P., Li, J., Yekhanin, S., et al., 2012. Erasure coding in windows azure storage. In: Usenix annual technical conference. Boston, MA, pp. 15–26.

Huang, C., Xu, L., 2008. Star: An efficient coding scheme for correcting triple storage node failures. IEEE Transactions on Computers 57 (7), 889–901.

Hughes, G. F., Murray, J. F., Kreutz-Delgado, K., Elkan, C., 2002. Improved disk-drive failure warnings. IEEE Transactions on Reliability 51 (3), 350–357.

Hussein, M.-K., Mousa, M.-H., 2012. A light-weight data replication for cloud data centers environment. International Journal of Engineering and Innovative Technology 1 (6), 169–175.

Jin, C., Jiang, H., Feng, D., Tian, L., 2009. P-code: A new raid-6 code with optimal properties. In: Proceedings of the 23rd international conference on Supercomputing. ACM, pp. 360–369.

Kermarrec, A.-M., Le Scouarnec, N., Straub, G., 2011. Repairing multiple failures with coordinated and adaptive regenerating codes. In: Network Coding (NetCod), 2011 International Symposium on. IEEE, pp. 1–6.

Khan, O., Burns, R. C., Plank, J. S., Pierce, W., Huang, C., 2012. Rethinking erasure codes for cloud file systems: minimizing i/o for recovery and degraded reads. In: FAST. p. 20.

Kumar, A., Tandon, R., Clancy, T. C., 2014. On the latency and energy efficiency of erasure-coded cloud storage systems. arXiv preprint arXiv:1405.2833.

Kune, R., Konugurthi, P. K., Agarwal, A., Chillarige, R. R., Buyya, R., 2016. The anatomy of big data computing. Software: Practice and Experience 46 (1), 79–105.

Le Scouarnec, N., 2012. Exact scalar minimum storage coordinated regenerating codes. In: Information Theory Proceedings (ISIT), 2012 IEEE International Symposium on. IEEE, pp. 1197–1201.

Li, J., Li, B., 2013. Erasure coding for cloud storage systems: a survey. Tsinghua Science and Technology 18 (3), 259–272.

Li, P., Li, J., Stones, R. J., Wang, G., Li, Z., Liu, X., 2016. Procode: A proactive erasure coding scheme for cloud storage systems. In: Reliable Distributed Systems (SRDS), 2016 IEEE 35th Symposium on. IEEE, pp. 219–228.

Li, W., Yang, Y., Yuan, D., 2011. A novel cost-effective dynamic data replication strategy for reliability in cloud data centres. In: Dependable, Autonomic and Secure Computing (DASC), 2011 IEEE Ninth International Conference on. IEEE, pp. 496–502.

Lin, S.-J., Chung, W.-H., 2014. Novel repair-by-transfer codes and systematic exact-mbr codes with lower complexities and smaller field sizes. IEEE Transactions on Parallel and Distributed Systems 25 (12), 3232–3241.

Lin, W., Chiu, D. M., Lee, Y., 2004. Erasure code replication revisited. In: Peer-to-Peer Computing, 2004. Proceedings. Proceedings. Fourth International Conference on. IEEE, pp. 90–97.

Liu, J., Shen, H., 2016. A low-cost multi-failure resilient replication scheme for high data availability in cloud storage. In: High Performance Computing (HiPC), 2016 IEEE 23rd International Conference on. IEEE, pp. 242–251.

Long, S.-Q., Zhao, Y.-L., Chen, W., 2014. Morm: A multi-objective optimized replication management strategy for cloud storage cluster. Journal of Systems Architecture 60 (2), 234–244.

Ma, A., Traylor, R., Douglis, F., Chamness, M., Lu, G., Sawyer, D., Chandra, S., Hsu, W., 2015. Raidshield: characterizing, monitoring, and proactively protecting against disk failures. ACM Transactions on Storage (TOS) 11 (4), 17.

Ma, Y., Nandagopal, T., Puttaswamy, K. P., Banerjee, S., 2013. An ensemble of replication and erasure codes for cloud file systems. In: INFOCOM, 2013 Proceedings IEEE. IEEE, pp. 1276–1284.

Mesnier, M., Ganger, G. R., Riedel, E., 2003. Object-based storage. IEEE Communications Magazine 41 (8), 84–90.

Muralidhar, S., Lloyd, W., Roy, S., Hill, C., Lin, E., Liu, W., Pan, S., Shankar, S., Sivakumar, V., Tang, L., et al., 2014. f4: Facebooks warm blob storage system. In: Proceedings of the 11th USENIX conference on Operating Systems Design and Implementation. USENIX Association, pp. 383–398.

O'Reilly, J., 2016. Network Storage: Tools and Technologies for Storing Your Companys Data. Morgan Kaufmann.

Papailiopoulos, D. S., Dimakis, A. G., 2011. Distributed storage codes through hadamard designs. In: Information Theory Proceedings (ISIT), 2011 IEEE International Symposium on. IEEE, pp. 1230–1234.

Pei, X., Wang, Y., Ma, X., Fu, Y., Xu, F., 2015. Cooperative repair based on tree structure for multiple failures in distributed storage systems with regenerating codes. In: Proceedings of the 12th ACM International Conference on Computing Frontiers. ACM, p. 14.

Pinheiro, E., Bianchini, R., Dubnicki, C., 2006. Exploiting redundancy to conserve energy in storage systems. In: ACM SIGMETRICS Performance Evaluation Review. Vol. 34. ACM, pp. 15–26.

Pinheiro, E., Weber, W.-D., Barroso, L. A., 2007. Failure trends in a large disk drive population. In: FAST. Vol. 7. pp. 17–23.

Plank, J. S., 2009. The raid-6 liber8tion code. The International Journal of High Performance Computing Applications 23 (3), 242–251.

Plank, J. S., 2013. Erasure codes for storage systems: A brief primer. The Usenix Magazine 38 (6), 44–50.

Plank, J. S., Blaum, M., Hafner, J. L., 2013. Sd codes: erasure codes designed for how storage systems really fail. In: FAST. pp. 95–104.

Qu, Y., Xiong, N., 2012. Rfh: A resilient, fault-tolerant and high-efficient replication algorithm for distributed cloud storage. In: Parallel Processing (ICPP), 2012 41st International Conference on. IEEE, pp. 520–529.

Rajasekharan, 2014. Data reliability in highly fault-tolerant cloud systems. URL https://pdfs.semanticscholar.org/abe7/7e70864a0d914365ed755cac5ce1abc3b8b0.pdf/

Rashmi, K., Shah, N. B., Gu, D., Kuang, H., Borthakur, D., Ramchandran, K., 2013. A solution to the network challenges of data recovery in erasure-coded distributed storage systems: A study on the facebook warehouse cluster. In: HotStorage.

Rashmi, K., Shah, N. B., Gu, D., Kuang, H., Borthakur, D., Ramchandran, K., 2014. A hitchhiker's guide to fast and efficient data reconstruction in erasure-coded data centers. In: ACM SIGCOMM Computer Communication Review. Vol. 44. ACM, pp. 331–342.

Rashmi, K., Shah, N. B., Kumar, P. V., Ramchandran, K., 2009. Explicit construction of optimal exact regenerating codes for distributed storage.

In: Communication, Control, and Computing, 2009. Allerton 2009. 47th Annual Allerton Conference on. IEEE, pp. 1243–1249.

Rashmi, K. V., Shah, N. B., Kumar, P. V., 2011. Optimal exact-regenerating codes for distributed storage at the msr and mbr points via a product-matrix construction. IEEE Transactions on Information Theory 57 (8), 5227–5239.

Rodrigues, R., Liskov, B., 2005. High availability in dhts: Erasure coding vs. replication. Peer to Peer Systems IV, 226–239.

Sathiamoorthy, M., Asteris, M., Papailiopoulos, D., Dimakis, A. G., Vadali, R., Chen, S., Borthakur, D., 2013. Xoring elephants: Novel erasure codes for big data. In: Proceedings of the VLDB Endowment. Vol. 6. VLDB Endowment, pp. 325–336.

Shah, N. B., Rashmi, K. V., Kumar, P. V., Ramchandran, K., 2012. Distributed storage codes with repair-by-transfer and nonachievability of interior points on the storage-bandwidth tradeoff. IEEE Transactions on Information Theory 58 (3), 1837–1852.

Sharma, Y., Javadi, B., Si, W., Sun, D., 2016. Reliability and energy efficiency in cloud computing systems: Survey and taxonomy. Journal of Network and Computer Applications 74, 66–85.

Shum, K. W., 2011. Cooperative regenerating codes for distributed storage systems. In: Communications (ICC), 2011 IEEE International Conference on. IEEE, pp. 1–5.

Shum, K. W., Hu, Y., 2011. Exact minimum-repair-bandwidth cooperative regenerating codes for distributed storage systems. In: Information Theory Proceedings (ISIT), 2011 IEEE International Symposium on. IEEE, pp. 1442–1446.

Shum, K. W., Hu, Y., 2012. Functional-repair-by-transfer regenerating codes. In: Information Theory Proceedings (ISIT), 2012 IEEE International Symposium on. IEEE, pp. 1192–1196.

Suh, C., Ramchandran, K., 2010. On the existence of optimal exact-repair mds codes for distributed storage. arXiv preprint arXiv:1004.4663.

Suh, C., Ramchandran, K., 2011. Exact-repair mds code construction using interference alignment. IEEE Transactions on Information Theory 57 (3), 1425–1442.

Sun, D.-W., Chang, G.-R., Gao, S., Jin, L.-Z., Wang, X.-W., 2012. Modeling a dynamic data replication strategy to increase system availability in cloud computing environments. Journal of computer science and technology 27 (2), 256–272.

Tamo, I., Wang, Z., Bruck, J., 2011. Mds array codes with optimal rebuilding. In: Information Theory Proceedings (ISIT), 2011 IEEE International Symposium on. IEEE, pp. 1240–1244.

Vijayakumari, R., Kirankumar, R., Rao, K. G., 2014. Comparative analysis of google file system and hadoop distributed file system.

Vins, V. A., Umamageswari, S., Saranya, P., 2014. A survey on regenerating codes.

Vishwanath, K. V., Nagappan, N., 2010. Characterizing cloud computing hardware reliability. In: Proceedings of the 1st ACM symposium on Cloud computing. ACM, pp. 193–204.

Wang, A., Zhang, Z., 2013. Exact cooperative regenerating codes with minimum-repair-bandwidth for distributed storage. In: INFOCOM, 2013 Proceedings IEEE. IEEE, pp. 400–404.

Wang, P., Dean, D. J., Gu, X., 2015. Understanding real world data corruptions in cloud systems. In: Cloud Engineering (IC2E), 2015 IEEE International Conference on. IEEE, pp. 116–125.

Weatherspoon, H., Kubiatowicz, J., et al., 2002. Erasure coding vs. replication: A quantitative comparison. In: IPTPS. Vol. 1. Springer, pp. 328–338.

Wu, Y., 2011. A construction of systematic mds codes with minimum repair bandwidth. IEEE Transactions on Information Theory 57 (6), 3738–3741.

Xiang, L., Xu, Y., Lui, J., Chang, Q., 2010. Optimal recovery of single disk failure in rdp code storage systems. ACM SIGMETRICS Performance Evaluation Review 38 (1), 119–130.

Yixian, Y., 1994. Peiod distribution for blaum-roth code. JOURNAL OF BEIJING UNIVERSITY OF POSTS AND TELECOMMUNICATIONS 3.

Zaharia, M., Chowdhury, M., Franklin, M. J., Shenker, S., Stoica, I., 2010. Spark: Cluster computing with working sets. HotCloud 10 (10-10), 95.

Zeng, W., Zhao, Y., Ou, K., Song, W., 2009. Research on cloud storage architecture and key technologies. In: Proceedings of the 2nd International Conference on Interaction Sciences: Information Technology, Culture and Human. ACM, pp. 1044–1048.