Efficient Data Reliability Management of Cloud Storage Systems for Big Data Applications

Rekha Nachiappan

A thesis presented for the degree of

Doctor of Philosophy

School of Computer, Data and Mathematical Sciences

Western Sydney University

Australia

2020

Efficient Data Reliability Management of Cloud Storage Systems for Big Data Applications

Abstract

Data Reliability Management in Cloud Storage Systems for Big Data Applications

by

Rekha Nachiappan

Doctor of Philosophy in Information and Communication Technology
Western Sydney University
Bahman Javadi, Principal Supervisor
Rodrigo N Calheiros, Kenan M. Matawie, Co-supervisors

The revolution of Big Data influences various sectors such as banking, healthcare, energy, consumer, manufacturing and education. Traditional storage systems are incapable of handling unprecedented growth of data. Cloud storage systems are distributed and scalable in nature. They offer more efficient platform to store and analyse Big Data. Cloud storage systems are composed of large number of hardware and software components that are vulnerable to failures. Hence failures in cloud storage systems are inevitable. Any failure in hardware, software, network or power supply will compromise durability and availability of data.

In order to improve data reliability, various data redundancy techniques are employed in cloud storage systems. The most prominent data redundancy techniques are replication and erasure coding. Replication maintains multiple copies of data in several locations. In case of failure, data repair is activated to maintain data reliability. Data repair in replication simply copies all missing data from next available location. Even though replication sounds simple, it incurs more storage overhead to improve the reliability of Big Data. Erasure coding is a viable alternative to replication since it improves data reliability with less storage overhead using parity data. Many popular storage systems have started adopting erasure coding to improve data reliability with huge cost savings. However, data repair in erasure coding is not as simple as replication. During data

repair, missing data has to be reconstructed data using parity data. Data reconstruction in erasure coding will not only incur more disk I/O and network bandwidth but also increase data access latency. Data repair overhead prevents erasure coding being more pervasive in cloud storage. The mission of this thesis is to address the challenges involved in employing erasure coding in cloud storage system. The contributions of this thesis are listed below:

- Replication offers exceptional read performance since it does not incur exponential resource consumption during data repair like erasure coding. Activating proactive replication of failure predicted data in erasure coding will not only reduce resource consumption during repair but also improve read performance. Based on this, a system with novel proactive recovery techniques are proposed in this thesis. The proposed system adapts to client requirements and selects an appropriate proactive recovery technique utilizing failure predictions.

- To further improve resource savings in erasure coded storage systems, we propose an optimization technique that attempts to minimize the number of data blocks to be replicated during proactive recovery, in the event of any failure prediction. We formulate the optimization problem as an integer linear program using data duplication information and system's network traffic. The objective to minimize the number of data blocks to be replicated during proactive recovery process.

- In erasure coding, any data read request to a failed data is served by performing data reconstruction on the fly. Such data reconstructions increase data access latency (degraded read latency). To address this, we propose a novel caching technique that proactively replicates failure predicted data into cache. Since data access from cache is faster, this technique eliminates degraded read latency.

Statement of Authentication

To the best of my knowledge, I declare that the work presented in this thesis is original. I hereby declare that I have not submitted this material, either in full or in part, for a degree at this or any other institution.

Rekha Nachiappan

Acknowledgements

I am delighted to acknowledge the people who have supported me to successfully complete my PhD study. Firstly, I would like to express my sincere gratitude to my advisor Associate Professor Bahman Javadi for his continued support, guidance and immense of knowledge throughout my PhD study. He motivated me to achieve high since the beginning of my PhD. He has provided the freedom of choosing research problems on my own, while also providing necessary guidance. His constant support and dedication helped me to complete my PhD with high scientific quality. I would like to extend my gratitude to my co-advisor Dr. Rodrigo N. Calheiros for his feedback and comments. His extensive feedback helped me to improve my presentation and writing skills. His technical suggestions and advice helped me to automate some of the manual steps, between each simulation iteration. I would also like to extend my gratitude to my other co-supervisor Dr. Kenan M. Matawie, for his support, feedback and motivational examples.

I would like to extend my sincere gratitude to Western Sydney University for granting WSU HDR Community Scholarship and Post Graduate Research Award. Special thanks to Western Sydney University for identifying and recognising my hard work with WSU HDR Community Scholarship. This reward motivated me to achieve high, regardless of the maternity break during my study.

Thanks to my team members Yogesh, Raed and Mohammad for sharing some useful materials. Thanks to Jesse for his guidance with my first tutoring, in Australia. Thanks to Guang and Jhon for their technical support.

Thanks to Dr. Weisheng Si and Mr. Paul Davies for consistently offering me with some teaching positions. I have learned many things from both of them and they inspired me to pursue my career as an academia. I would like to thank my school teacher Stephan for teaching me English grammar and my under graduate lecturer Subramanian for always appreciating my reasoning skills and encouraging me to constantly ask questions during his class.

I would like to thank my parents and sisters and their family for their love and support.

My mother Tamilarasi is a person who always encourages and supports me to take part in several activities since my childhood. My father Nachiappan always admires me of my determination and clarity in my work. He have supported and encouraged me to pursue post graduate. My elder sister Meenambigai taught me many life skills. My younger sister Durga is the first person who I call when I feel like talking to someone. I would also like to thank my in laws Subramanian, Ramalakshmi, Sankar, Suresh Kumar, Ammu and Aishwarya, and my little darlings Priyangka, Karthik and Deepak.

Thanks to my little princess Kaavya. She has closely travelled with me during my PhD study as being both internal and external bond. Thanks to my little prince Naren for being my best companion. He has spent most of his holidays at WSU campus, politely with me. Finally and most importantly, thanks to my husband Umamaheswaran for his love, support and understanding. His moral support enabled me to successfully fulfil my dream. This work would have not been possible, without his constant love and support. Thank you for being an amazing better half!!!

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Digital data is rapidly growing in today's Big Data era. Internet of Things (IoT), online transactions and social media are playing an important role in generating massive amount of digital data. International Data Corporation (IDC) has predicted that the data generated in whole world will reach 175 zettabytes by 2025 [1]. The knowledge derived from Big Data has shown significant impact on business and society. To derive valuable insight from Big Data on time, it is imperative to store and processes them in an efficient platform. The on-demand scalability nature of cloud computing plays a vital role in efficiently handling, rapid and unprecedented growth of digital data. IDC also predicts that 40% of world data will reside in public cloud environment by 2025 [1].

Cloud storage systems are composed of large number of hardware and software components. Failures are the norm rather than exception in cloud storage systems. Any failures such as hardware failures, power outage, software glitches, maintenance shut down or network failures in cloud storage system will raise temporary data unavailability events and sometimes it leads to permanent data loss. Figure 1.1 represents node failure behaviour in Facebook's 3000 machine production cluster. It shows at least 20 machine failures are encountered in each day [2]. In spite of these failures, to provide reliable service to the customers, various fault tolerant mechanisms are employed.

To meet large scale storage needs of clients, cloud defines virtual storages using Network Attached Storage (NAS) and Storage Area Network (SAN). The networked storage

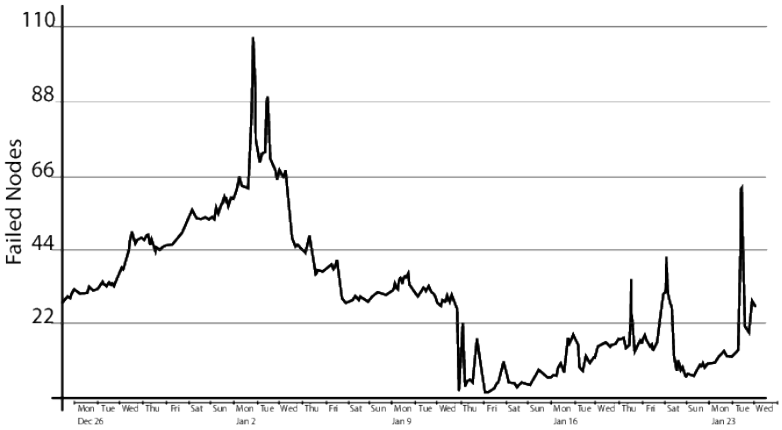


Figure 1.1: Node failures in Facebook [2].

NAS and SAN are easily scalable in terms of both performance and capacity and hence they are highly influential in cloud storage systems. They use distributed file system to organize data into storage and to provide controlled data access to clients. Distributed File System (DFS) spreads data in a storage cluster which is composed of thousands of nodes. DFS is also designed to ensure durability, availability and I/O performance of the storage according to client's Service Level Agreement (SLA). DFS applies data redundancy to improve the fault tolerance of cloud storage system and it spreads redundant data into nodes from different failure zones. Any aforementioned failures in cloud storage system may lead to unavailability events from time to time. Whenever an unavailability event occurs, it activates data recovery to maintain durability and availability of data.

Data redundancy mechanisms employed in cloud storage systems are replication and erasure coding. Replication maintains multiple copies of data on distinct nodes from different failure domains. Replication is simple and straightforward fault tolerant method. However, it is not an efficient solution for Big Data due to the volume of data. Erasure coding is a storage efficient alternative reliability method. A file system with (n, k) erasure codes divides a file or object into k equal chunks and calculates n-k parity chunks. The set of n+k chunks compose a stripe and each chunk is stored on unique n+k locations from different failure domains such that any unavailable chunk can be reconstructed using any other k available chunks. Hence (n, k) erasure code can tolerate any n-k failures. Figure 1.2 depicts distribution of data in (5, 3) erasure coded storage. Obj1 is decomposed into

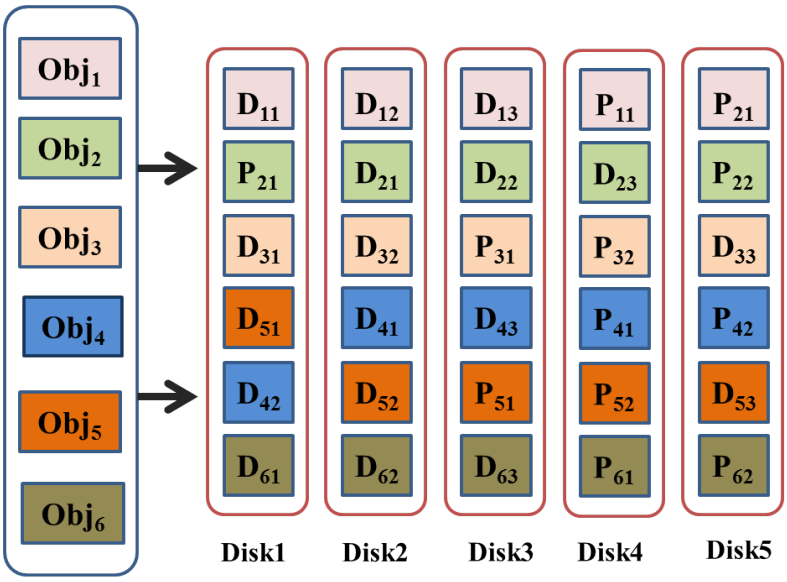


Figure 1.2: Data distribution in (5, 3) erasure coded storage.

chunks D11, D12 and D13 and it is stored along with the parity chunks P11 and P12 such that the chunks D11, D12, D13, P11 and P12 together constitute a stripe and hence the system can tolerate any 2 failures. Failure domain can be chosen as machine, disk or rack. In Figure 1.2, the failure domain is defined as disks. Many popular cloud storage systems like Facebook and Microsoft have employed erasure coding to increase storage efficiency [3].

When there is a failure in cloud storage systems, the objects that are resided in the failed zone will enter into degraded mode. To avoid any unnecessary repair, a delay is applied to recover any degraded objects[3]. Degraded objects will remain in degraded mode from the time of failure till complete recovery. Any data read request to degraded object in replication is handled by redirecting requests to the next available replica. On the other hand, in erasure coding, degraded object is reconstructed on the fly. In replication, object is recovered by copying it from next available replica, whereas in erasure coding, object is recovered using data reconstruction of any other k available chunks.

Popular Hadoop Distributed File System (HDFS) uses three replicas. Hence it can tolerate any two simultaneous failures with storage overhead of 3x. The most popular Reed-Solomon(14, 10) can manage any 4 simultaneous failures with 1.4x storage overhead [4]. Even though storage efficiency of erasure coding sounds appealing, data recovery/

repair in erasure coding involves enormous resource consumption. For example, data recovery in Reed-Solomon(14, 10) code increases disk I/O and network bandwidth by 10x compared to replication [5]. Increased resource consumption due to data recovery also impacts read performance. Data recovery in replication has limited impact on both resource consumption and on read performance. Data recovery issues of erasure coding prevent it being more pervasive in cloud storage systems. For example, in a 3000 nodes production cluster of Facebook, erasure code can replace replication for only 8% of data. In case 50% of data are replaced with erasure code, the repair network traffic will saturate cluster network links [2].

1.1 Research Challenges and Hypothesis

There are many open challenges on improving reliability of Big Data applications. Some of the important challenges are as follows:

1.1.1 Storage Efficiency

Data reliability and storage overhead in replication are directly proportional to each other. Improving storage overhead without sacrificing reliability is the greatest challenge in replication. Even though erasure coding offers tremendous storage savings with fair reliability, repair resource consumption compensates it. Enabling automation of dynamic redundancy, such as changing number of replicas of erasure coded data chunks by incorporating failures and data access spikes could also improve storage efficiency further.

1.1.2 Bandwidth Efficiency

Network bandwidth is always a scarce resource in a distributed storage. Bandwidth usage is directly proportional to the amount of data transferred. In both replication and erasure coding, data repair consumes considerable amount of network bandwidth. However, it is exponential in erasure coding. Node failures occur more often in cloud storage and it is being a major reason of increased network traffic in erasure coded storage system [2]. Literature shows various methods to reduce network traffic in erasure coded storage systems. However, none could reduce network traffic as good as replication. Recent works

on reducing network traffics also show limited savings. Activating proactive recovery in erasure coding is a key to reduce recovery network bandwidth. Failure prediction should be utilized to handle proactive recovery.

1.1.3 Energy Efficiency

Energy savings due to minimal storage overhead in erasure code can be compromised by the extensive resource usage during data repair. Activating efficient data repair in erasure code can reserve the energy saved due to storage. Activating proactive replication in erasure code can reduce recovery energy. Since proactive replication also demands additional storage, it must be wisely defined. Additional replicas must be placed only based on the need.

1.1.4 Big Data

Cloud storage is the cost-effective platform to support the variety, volume and velocity parameters of Big Data. However, cloud storage also confronts several challenges on improving the reliability of Big Data. Cloud storage has to employ data redundancy techniques to ensure the reliability of data. The parameter *variety* will not get affected while employing data redundancy techniques, but it confronts several challenges with respect to the parameters, *volume* and *velocity* of Big Data. The most important data redundancy techniques employed in cloud storage are replication and erasure coding. Replication is not a cost efficient solution to improve the reliability of Big Data while considering its volume. Erasure coding is a cost effective solution to enhance the reliability of Big Data. However, any failures in the erasure coded storage system activate data repair, which increases disk I/O, network bandwidth and data access latency. Data repair in erasure coding affects the velocity of data read. The cost effective novel hybrid redundancy techniques should be proposed to improve the reliability of Big Data in cloud storage.

1.1.5 Data Access Latency

In replication, an object is placed in multiple locations. When the object is degraded, the replicas can be used to serve any read request. Replicas can also be utilized to effectively handle a sudden spike in I/O queue. Data access latency can be significantly reduced

using replication in both of these cases. However, it is not an efficient solution when we consider the volume attribute of Big Data. The computational and I/O overhead involved in reconstructing degraded objects of erasure code increase latency. Increased latency of storage efficient erasure code is a pitfall to the velocity attribute of Big Data. Reducing access latency with less storage overhead especially for Big Data is a challenge to the researchers. Activating cognitive, dynamic, proactive replication in erasure codes, using data access history and failure logs reduce degraded read latency with less storage overhead. Cache can also be utilized wisely to reduce degraded read latency.

1.1.6 Research Questions

By considering the above research challenges the research questions are formulated as follows,

- How to define a storage efficient hybrid reliability technique for Big Data using replication and erasure coding?
- How to define a bandwidth and energy efficient hybrid reliability technique?
- How to enable cloud to support velocity attribute of Big Data with less storage overhead?

Replication cannot be a promising solution for improving reliability of Big Data as it naturally increases storage overhead while improving reliability. However, data repair in replication is simple. It does not activate sudden peak in network usage as it activates less number of disk I/O compared to erasure coding. It does not affect the access latency of degraded objects. Erasure coding is a viable storage efficient alternative to improve reliability of Big Data. However, data repair process activates sudden peak in network usage and disk I/O. Any read request to degraded data is handled by performing decoding on the fly. This increases data access latency. Several researches have been conducted to improve data reliability and storage efficiency in parallel using replication. However, they all suggest compromising one for other. Many researches concentrated on defining repair efficient novel erasure codes but none of them could reduce recovery resource usage as good as replication.

1.2 Thesis motivations, goals and contributions

To bring together the benefits of both replication and erasure coding, an optimal hybrid reliability technique using replication and erasure coding has to be defined. The motivations of the thesis are derived from the following considerations:

- Improving reliability in replication involves incredible storage overhead. Defining a repair efficient erasure code suggests additional storage space for minimal resource savings. To bring together benefits of both replication and erasure coding, in this thesis, we will consider defining a proactive replication technique in erasure coded storage. This technique has to leverage storage efficiency and reliability benefits of erasure coding while also reducing recovery resource consumption.

- Availability and access latency requirements may vary with respect to hot and cold status of data. Lazy recovery delays repair until certain number of blocks in a stripe are degraded. This reduces repair bandwidth significantly. However, lazy recovery impacts availability and read performance of data. Proactive recovery could increase reliability and read performance but it consumes additional resources. To apply appropriate recovery methods client SLA can be utilized. In this thesis we will define a system that makes an appropriate choice between lazy (delaying repair until certain blocks in a stripe are degraded) and proactive recovery utilizing client SLA to maximize resource savings in erasure codes.

- Defining a cache tier on erasure coded storage can reduce data access latency. However, this will have very minimal effect on reducing degraded read latency especially in the following scenarios. Cache is ineffective when an application constantly changes access pattern or it does not follow any access pattern. A freshly introduced cache tier will have minimal impact on reducing access latency. This research will also focus on defining a cache based solution to address the issue of a peak in access latency due to a failure in erasure coded storage.

The goals of this thesis are as follows:

- Defining efficient proactive recovery techniques for erasure codes to mitigate its extensive resource consumptions during recovery.
- Defining a system that adapts to client SLA and enforces different proactive recovery techniques in erasure codes using status (hot, cold) of data blocks.
- Defining a cache based solution to reduce degraded latency.

Regarding the above goals this thesis makes following contributions:

1. Literature Survey: This thesis provides a comprehensive survey on popular data reliability techniques; replication and erasure coding. Each of those techniques has their own trade off with various parameters such as durability, availability, storage overhead, performance energy consumption and network bandwidth/traffic. This survey highlights the challenges involved in employing each method. It also highlights the research gaps on improving fault tolerance in cloud storage systems.

2. Novel proactive data recovery techniques: To address the recovery resource consumptions and performance issues of erasure codes, several prediction based proactive recovery methods are proposed.

   This thesis presents several novel proactive recovery techniques in erasure codes. The proposed novel proactive recovery techniques are proposed as a combination of proactive replication, typical reconstruction of erasure codes and lazy recovery. Proactive replication creates a replica ahead of an occurrence of failure, whereas lazy recovery applies a delay in reconstruction until a certain number of blocks in a stripe are failed. Proactive recovery methods in erasure coding are defined using failure predictions. We propose a system with novel proactive recovery techniques ProDisk, ProMachine, ProHot and ProHot_LazyCold. The proposed system must use data access pattern to identify hot data. The proposed system must also confirm and utilize client's latency, durability and availability requirements to select one of the most appropriate proactive techniques. In case of any failure predictions or failures, the system adapts to client SLA and selects one of the most suitable recovery techniques to reduce recovery network bandwidth/traffic.

Optimizing the recovery techniques ProDisk, ProMachine, ProHot and ProHot_LazyCold can further improve storage, network bandwidth and energy savings. By examining storage system's current network usage and data redundancy information, proactive replication of certain data blocks can be avoided. In order to maximize the resource savings, we present an optimization problem. It is formulated using Integer Linear Programming (ILP). The objective of this problem is to minimize proactive replicas. A novel optimization based proactive recovery technique is also introduced. An energy estimator is introduced to analyse the energy consumption of various storage systems.

3. Novel caching technique: A novel proactive caching solution is proposed to reduce degraded read latency in erasure codes.

   This thesis presents a pre-fetching method for cache to reduce degraded read latency. It uses disk failure prediction information to perform pre-fetching and moving data into cache. The proposed proactive recovery techniques can improve read performance using proactive replicas. However, they require changing metadata. Novel caching technique reduces degraded read latency with no changes in underlying storage and metadata.

Novel proactive recovery techniques significantly reduce recovery network bandwidth in erasure coding, which prevented erasure being more pervasive in cloud storage. The novel proactive techniques also reduce number of degraded slices, which could lead to the reduction in degraded read latency. Hence the proposed novel proactive techniques can define cost effective solution for large volume of data. They can also support data read in high frequency. The novel caching technique proposed in this thesis addresses the important challenges on improving the reliability of Big Data. It reduces the degraded read latency of erasure codes with minimum storage overhead. Hence it improves reliability of Big Data with less storage overhead while also supporting data read in high velocity.

1.3 Thesis organization

The structure of this thesis is shown in Figure 1.3 and it is organised as follows:

Chapter 2 presents a state-of-the-art survey on improving fault tolerance in cloud storage systems. This describes the importance of improving fault tolerance and highlights the challenges of various fault tolerance techniques employed in cloud storage systems. This chapter also advises necessary future research directions. This chapter is mainly derived from: - Nachiappan, R., Javadi, B., Calheiros, R. N., & Matawie, K. M. (2017).

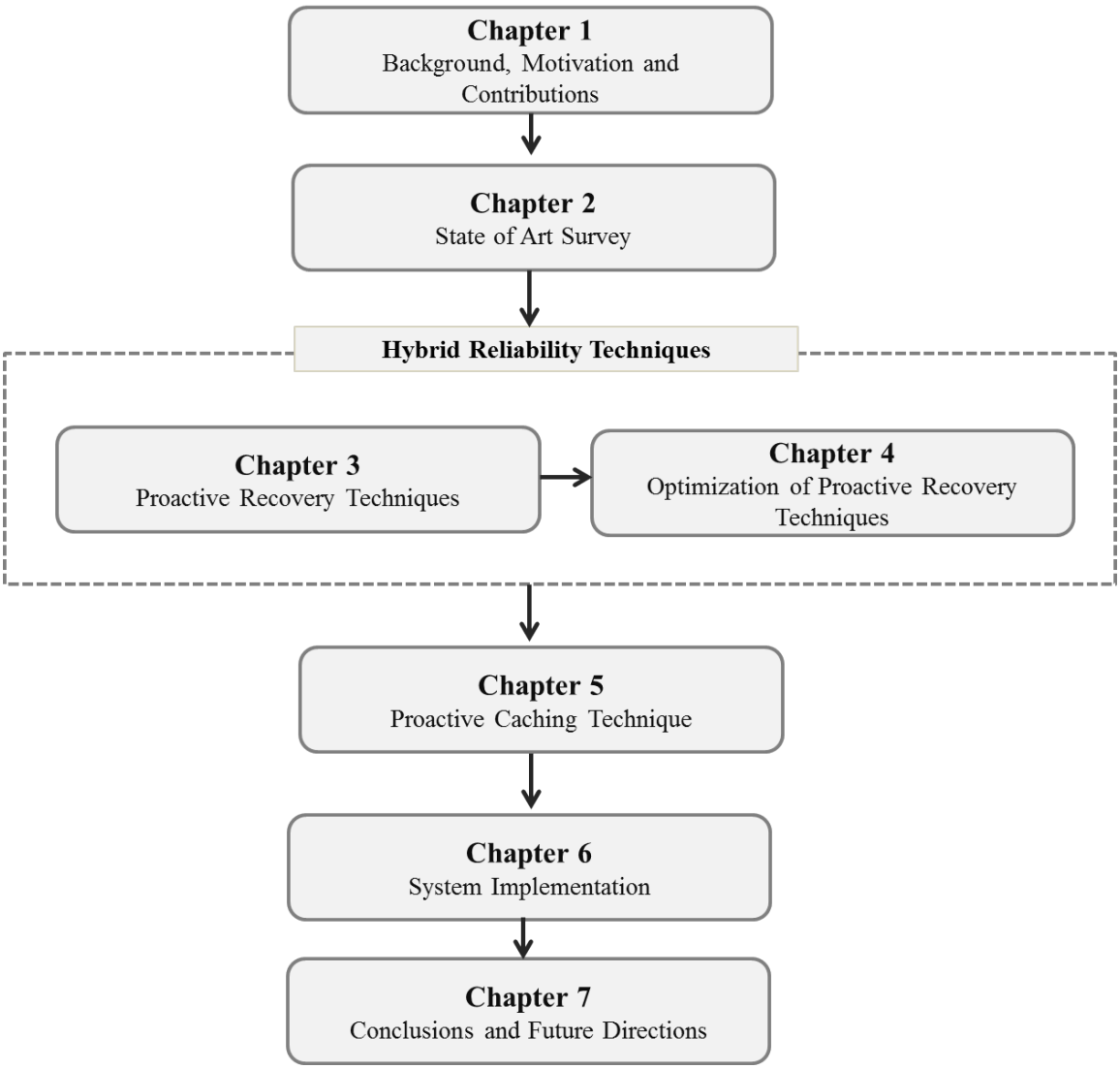


Figure 1.3: Thesis structure.

Cloud storage reliability for big data applications: A state of the art survey. Journal of Network and Computer Applications, 97, 35-47.

Chapter 3 presents several novel proactive recovery techniques to reduce resource usage during failure. This also presents a system to employ proposed novel proactive

recovery techniques. The system is adapt to client SLA and select one of the suitable proactive recovery techniques in the event of any failure prediction. Results show significant bandwidth savings in erasure coding. This chapter is derived from:

- Nachiappan, R., Javadi, B., Calheiros, R. N., & Matawie, K. M. (2018, August). Adaptive Bandwidth-Efficient Recovery Techniques in Erasure-Coded Cloud Storage. In European Conference on Parallel Processing (pp. 325-338). Springer, Cham.

Chapter 4 presents an optimization problem which attempts to maximize the resource savings due to proactive recovery. To improve the resource savings in erasure coded storage, an ILP based optimization problem is defined to minimize proactive replication of data blocks. A novel optimization based proactive recovery technique is also proposed. This chapter is derived from:

- Nachiappan, R., Javadi, B., Calheiros, R. N., & Matawie, K. M. Enhancing Efficiency of Proactive Recovery in Erasure-Coded Cloud Storage Systems. (Submitted to IEEE Transactions on Parallel and Distributed Systems.)

Chapter 5 presents a cache based solution to reduce degraded read latency in erasure coded storage. In this chapter, we propose a system that proactively configures cache tier with the objects that are predicted to be degraded. This chapter is part of the below publication.

-Nachiappan, R., Javadi, B., Neves Calheiros, R., & Matawie, K. M. (2019). ProactiveCache: on reducing degraded read latency of erasure coded cloud storage. In Proceedings of the 11th IEEE International Conference on Cloud Computing Technology and Science (CloudCom 2019), the 19th IEEE International Conference on Computer and Information Technology (CIT 2019), the 2019 International Workshop on Resource brokering with blockchain (RBchain 2019), and the 2019 Asia-Pacific Services Computing Conference (APSCC 2019), 11-13 December 2019, Sydney, Australia (pp. 223-230).

Chapter 6 presents the design and architecture of a cloud storage framework which can be used to evaluate reliability and energy efficiency of storage systems. This chapter elaborates several modules of the framework which are used to estimate reliability, recovery bandwidth and energy consumption of various systems.

Chapter 7 concludes and provides the directions for future work.

Chapter 2

Literature Review

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 number of 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 chapter explores the challenges involved in employing both techniques in cloud storage systems for Big Data applications with respect to the aforementioned parameters.

2.1 Introduction

In the era of Big Data, data volume is growing faster than the storage capacity [6]. Each week, Facebook requires extra 60TB of storage just for new photos [7]. YouTube users upload over 400 hours of video every minute and it requires 1 Petabyte of new storage every day [8]. According to the International Data Corporation (IDC)'s sixth annual study, until 2020 the digital data will double every two years [6]. Cloud computing offers

This chapter is derived from:Nachiappan, R., Javadi, B., Calheiros, R. N., & Matawie, K. M. (2017). Cloud storage reliability for big data applications: A state of the art survey. Journal of Network and Computer Applications, 97, 35-47.

a cost-effective way to support Big Data applications that can derive important business value [9]. By 2020, approximately 40% of the data in the digital universe will be stored or processed in cloud [6]. Cloud storage provides reasonable scalability for storing Big Data and it helps to handle the steady growth of variety, volume and velocity properties of Big Data [10].

As cloud storage is built up on commodity servers and disk drives [11], it is subject to failures. Those failures can compromise the performance of applications relying on it. For example, in 2009, Facebook temporarily lost over 10% of its stored photographs because of a hard drive failure [12]. Amazon Simple Storage Service (S3) encountered a data corruption problem caused by a load balancer bug [13]. Amazon Web Services (AWS) suffered major disruptions due to a DynamoDB failure [14]. At Facebook, in a production cluster of 3000 nodes, it is typical to have 20 or more node failures [2]. As failures are the norm in cloud storage systems, improving performance of Big Data application during data recovery is one of the most important challenges.

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 [15]. 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 overhead for erasure coding is much smaller than replication. Hence it reduces the hardware needs for data storage and provides significant cost and energy savings in data centres [16]. 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 [17]. Microsoft Azure reduced storage overhead from 3x to 1.33x using erasure coding which provided over 50% cost savings [16].

A study on the Facebook warehouse cluster [18] 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 [2].

Another issue with the use of error correction techniques is increases latency due to network traffic. Storage systems consume up to 40% of a data centre's total energy [19] and energy efficiency of storage systems is influenced by read/write latency [20]. 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 chapter addresses ongoing researches on improving data reliability of Big Data Applications in cloud computing using replication and erasure coding. As both techniques have their own advantages and disadvantages, this chapter discuss how researchers are striving to bring the benefits of one technique to another.

2.2 Background

This section briefly discusses 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.

2.2.1 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 using storage virtualization [21]. Storage virtualization abstracts physical storage from applications and maps 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[22].

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 file server to authorize I/O and clients can directly access data using metadata. It offers good performance. However, it does not offer 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 it is 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. Nowadays object storage is becoming a popular choice of cloud clients. They provide simple put/get interface to store and retrieve data. Netflix uses Amazon S3 storage. Object storage supports 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 [23]. O'Reilly [24] has 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. [25].

2.2.2 Data Failures

In a cloud storage system, many factors can lead to data failure. Data failures will lead to cloud service failures. Sharma et al. [26] presented a detailed survey of cloud service failures. The main causes of cloud data failures are hardware, software, network, and power failures [27]. Disks are the central element in cloud based storage [28] and are the most common failure component [29]. Vishwanath and Nagappan analysed hardware reliability for a large cloud computing infrastructure [30]. As shown in Figure 2.1(data collected from [30]), 78% of failures were due to hard disks, 5% due 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 [30].

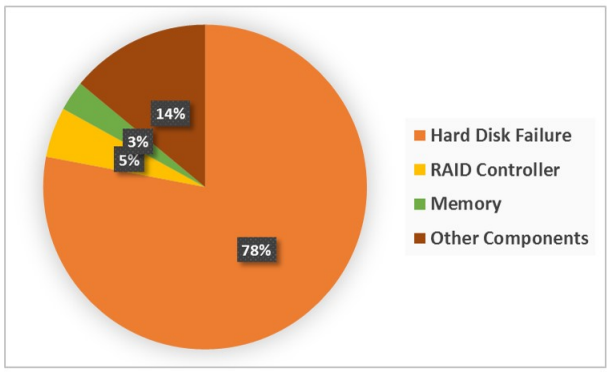


Figure 2.1: Causes of server failures in cloud computing systems [30].

As depicted in Figure 2.2, 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 [27]. 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 [11]. Data unavailability due to hard disk failure or data corruption leads to

permanent data loss.

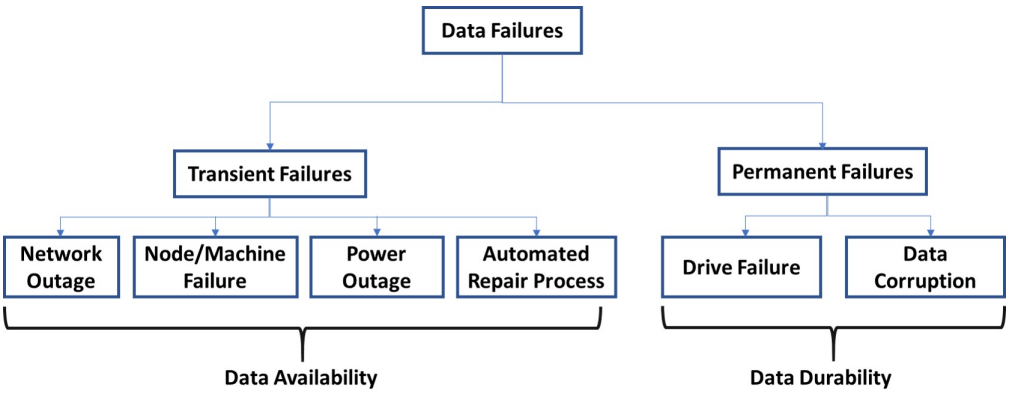


Figure 2.2: Data failures in cloud storage.

Pinheiro et al. [31] 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 correlated with the drive first scan errors, reallocations, offline allocations and probation counts. Ford et al. [11] 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. [32] 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 [33]. 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.

2.2.3 Data Reliability

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

As shown in Figure 2.3, 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) [34]. Replication and erasure coding are used to handle primary data failures. RDD is used to protect intermediate data generated by Big Data applications [34].

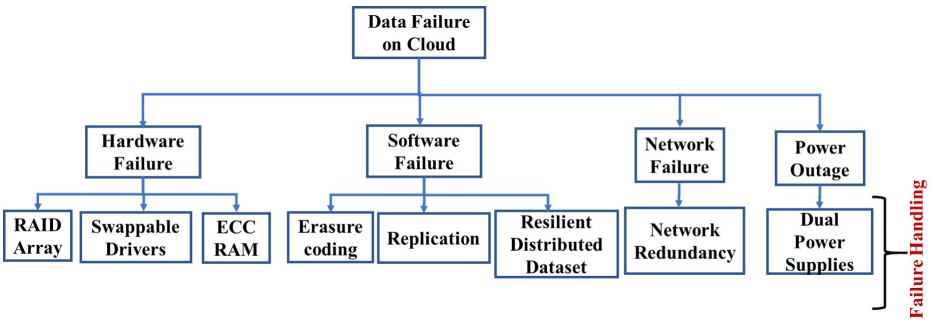


Figure 2.3: Failure Handling in Cloud Data Centres.

Erasure coding [16, 2] and replication [35] are the most popular reliability mechanisms employed on cloud storage. Figure 2.4 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 not available in one disk, it is served from the next available disk [15]. 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.

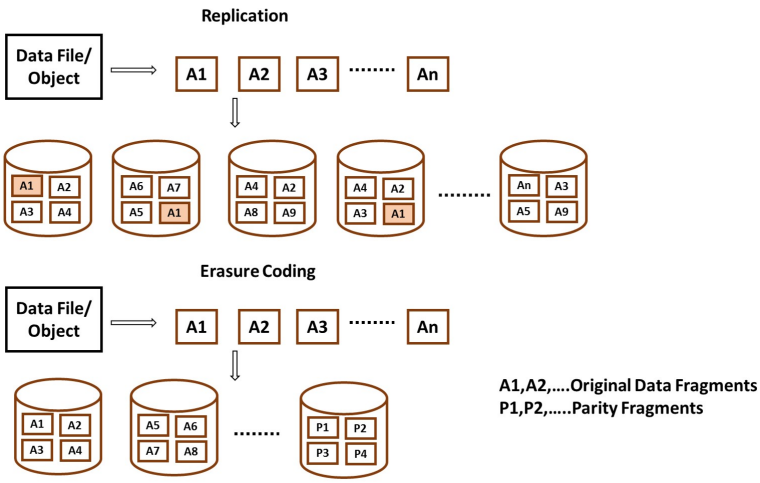


Figure 2.4: Replication and Erasure Coding.

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.

2.3 Erasure Codes

Erasure coding is playing a predominant role in protecting data from failures in large scale storage systems [15]. Before the emergence of cloud computing, erasure coding was used to detect and correct errors in storage and communication systems [36]. 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 2.5 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 [15]. 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 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 [15].

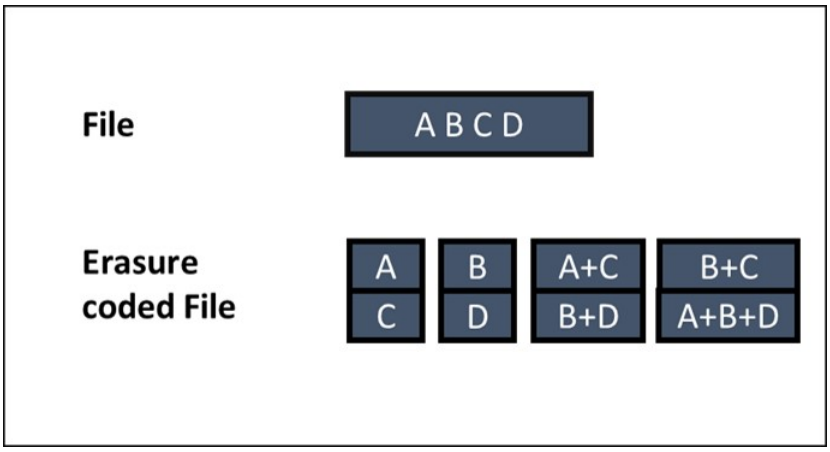


Figure 2.5: Erasure Coding.

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 combinations of m disk failures, if m disks are dedicated to hold parity data. For example, in Figure 2.6.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 2.6.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.

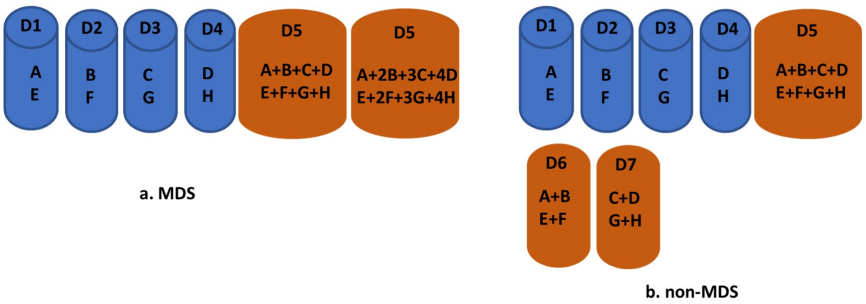


Figure 2.6: Different erasure coding types a. MDS codes b. non-MDS codes.

Examples of simple erasure codes are RAID-6, Array codes, and Reed-Solomon codes [15]. RAID-6 codes are MDS that creates two parity blocks for data blocks such that it can handle two disk failures [37]. 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) [38], EVENODD [39], Blaum-Roth [40] and Liberation codes [41] are array codes for RAID-6 that can tolerate up to two disk failures. Star code is an array code and it can tolerate any combination of three disk failures [42]. Cauchy Reed-Solomon, Generalized EVENODD and Generalized RDP are array codes that can be defined for any values of k and m [15]. 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 [15].

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[7, 16]. 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 researches focus on reducing reconstruction costs of failed data on Reed-Solomon coded storage systems.

This chapter 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). Following sections discuss non-MDS/LRC codes and regeneration codes respectively.

2.3.1 Non-MDS Codes/Locally Repairable Codes

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

Figure 2.7 represents locally repairable codes. Local parity helps blocks with single

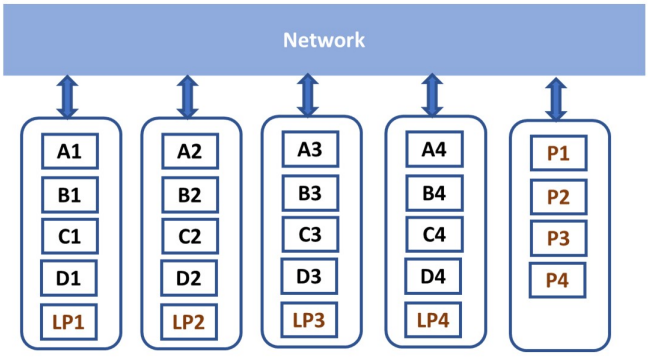


Figure 2.7: Locally Repairable Codes.

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. [43] 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 of generalized pyramid code are calculated using a generator matrix. For erasure codes with MR property, the matching condition becomes sufficient. That means all failure cases satisfying the matching condition are recoverable. Basic Pyramid code in comparison with generalized pyramid code

provides 45% less unrecoverable property [43].

Following this work, Huang et al. presented a new set of non-MDS erasure codes (Local Reconstruction Code) for Microsoft Azure Storage [16]. 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. [2] 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. Xu et al. [44], propose novel family of Concurrent Regeneration codes with Local reconstruction (CRL). This calculates g global parity chunks from all data chunks and divides m data chunks into l groups. CRL also calculates local parity in each group. CRL reduces network bandwidth, disk I/O and reconstruction time.

Plank et al. [45] 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.

Mehrabi et al. [46], proposed a method to construct a class of erasure codes to address update complexity issue of LRC codes which define a strict bound of update complexity. The proposed design algorithm reduces update complexity without sacrificing minimum distance, code rate and locality parameters. Li et al. [47], proposed a novel family of locally repairable codes called Galloper codes, to improve parallelism in existing LRC codes. Galloper codes carefully embed original data into all blocks by considering performance heterogeneity of servers. This improves performance of applications by activating

low disk I/O during reconstructions and by improving I/O parallelism.

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.

2.3.2 Regeneration Codes

Regeneration codes are defined for efficient repair of 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. [48] 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 2.8 [49]. Assume that each data block in the figure is 1GB. Upon failure, the reconstruction needs only 3 GB instead of 4 GB.

Table 2.1: Related work on reducing latency of erasure coded storage systems

Begin of Table					
Author	**Type of storage systems**	**Performance on Data Failure**	**Reliability**	**Energy Efficiency**	**Storage Overhead**
Huang et al. [43]	Cloud	Reduces number of blocks needed to reconstruct failed data	Tolerates any n-k-1 failures, and 86 % of n-k failures	NA	11% additional storage overhead

continuation of Table 2.1					
Author	**Type of storage systems**	**Performance on Data Failure**	**Reliability**	**Energy Efficiency**	**Storage Overhead**
Huang et al. [16]	Windows Azure Storage	Reduces disk I/O and network traffic	Tolerates any n-k-1 failures, and 86% of n-k failures	NA	1.6x of storage overhead
Sathiamoorthy et al. [2]	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. [48]	Distributed Storage	Improved performance in terms of network traffic	Improved Reliability	NA	No
Pei et al. [50]	Distributed Storage	Improved performance in terms of network traffic	Improved Reliability	NA	No
Khan et al. [51]	Cloud	Reduces the number of symbols for recovery and improves performance by 20%	Tolerates arbitrary n-k failures	NA	No
Rashmi et al. [52]	HDFSRAID in Facebook data warehouse	Reduces both network traffic and disk I/O around 25% to 45% compared to Reed-Solomon code	Tolerates arbitrary n-k failures	NA	No

continuation of Table 2.1					
Author	**Type of storage systems**	**Performance on Data Failure**	**Reliability**	**Energy Effi-ciency**	**Storage Over-head**
Li et al. [47]	Cloud	Reduces disk I/O and activate I/O parallelism and reduce the completion time of MapReduce jobs by up to 42.9%	Tolerates arbitrary n-k failures	NA	No
Xu et al. [44]	Cloud	Reduces disk I/O and reconstruction time and improves performance by 0.656x compared to [2]	Mean time to Data loss is high compared to Reed-Solomon code	NA	Similar to [2]
Pradeep et al. [53]	Cloud	Reduces response time and improves performance	Tolerates arbitrary n-k failures	NA	No

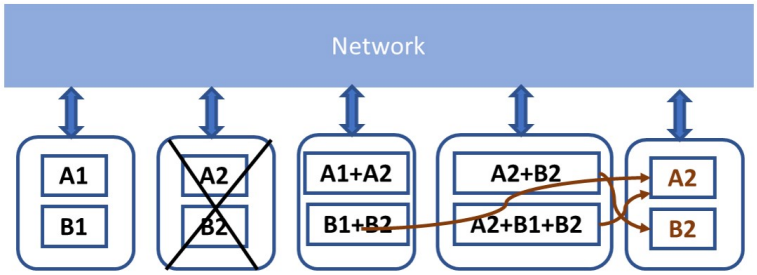


Figure 2.8: Regeneration Codes.

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 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 [54].

Suh and Ramchandran [54] 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. [49] 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 [55, 56, 57]. Hybrid MSR codes with various choices of parameters have been defined in various works[58, 59, 60], which support the exact repair for systematic parts and functional repair for parity parts.

The aforementioned regeneration codes did not consider cross cluster or intra-cluster repair bandwidth. Sohn et al. [61] proposed exact repair MSR codes for cross clustered storage systems. The proposed MSR coding scheme is suggested for repair bandwidth $1/(n-k)$ when the system parameter satisfy $n = Lk$ where L is number of clusters. All MSR and MBR codes focus on storage and bandwidth minimization but may increase disk I/O. The choices of parameters for exact repair remain an open problem.

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. [62] 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 [63, 64]. Exact repair is carried out by transfer MBR codes $(n, k = n-2, d = n-1)$ over finite field of size 2 defined in [65]. Lin and Chung [66] 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 $k \geq 3, \beta < d-k+1$ [67].

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.

Cooperative Recovery Regeneration Codes

Hu et al. [68] 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 [69, 70]. Shum and Hu [71] 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 [72] 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 [73] explain the construction of exact MSCR code for some choices of parameter when $d \geq k = 2$. Pei et al. [50] 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.

Cross-Rack Regeneration Codes

Hou et al. [74], proposed rack-aware regenerating codes (RRC) to achieve optimal trade-off between storage and cross-rack repair bandwidth of rack-based data centers. Two extreme optimal points are derived, namely the Minimum Storage Rack-aware Regeneration (MSRR) and Minimum Bandwidth Rack-aware Regeneration (MBRR) points, to give exact-repair constructions of MSRR codes and MBRR codes. Qu et al. [75], proposed Multi-rack Regeneration Codes(MRC) which repair a failed node by downloading data from nodes in the same rack only. MRC obtain optimal trade-off between storage and bandwidth using common product-matrix framework [49].

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. [38] propose Row Diagonal Optimal Recovery (RDOR) for single disk failure in RDP codes to reduce I/O costs for recovery. The I/O optimal recovery of single disk failure is defined here. Khan et al. [51] 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 ≤ 3.

The following are the system level solution to address the challenges of erasure codes. Rashmi et al. [52] propose Hitchhiker code, which is built on top of RS Code using 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. Silberstein et al. [4] proposed lazy recovery which applies a delay to recover failed data until the number of degraded chunks in a stripe reaches certain threshold and performs parallel reconstruction of degraded chunks. Parallel reconstruction reduces data transfer during data repair and hence reduces recovery bandwidth of erasure codes. It reduces recovery bandwidth up to 76% compared to Reed-Solomon. Li et al. [76] used disk failure prediction and defined proactive replication of data in failure predicted disks.

This method reduces degraded read latency and improves read performance. Li et al. [77] defined a cost effective data reliability management mechanism to ensure reliability of massive data with minimum replication based on a generalized data reliability model. Pradeep et al. [53] proposed a novel recovery mechanism CoARC for degraded read. When the system receives any degraded read request, it proactively recovers all degraded blocks in a strip and caches them. CoARC increases read performance and hence improves the performance of the application.

2.4 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 [78]. 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.

2.4.1 Static Replication

In static replication, the number of replicas and their locations are fixed [78]. 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. [79] 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.

Hassan et al. [80] proposed two multi objective based optimization algorithms namely Multi Objective Evolutionary (MOE) algorithm and Multi-Objective Randomized Greedy (MORG) algorithm for replica management. They determine optimal number of replicas and replica placement in an overlay with an objective with various parameters such as access latency, storage costs and data availability.

Liu and Shen [81] 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. MRR reduces the probability of data loss with low consistency maintenance cost. Long et al. [82] 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 decreases service time, latency and energy consumption.

2.4.2 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. [78], proposed a dynamic cost efficient replication in clouds with consideration of geographical diversity while maintaining high availability and low latency. Bonvin et al. proposed Skute, a key-value store which determines cost efficient position of replicas. Sun et al. [83] 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 [84] propose Resilient, Fault-tolerant and High-efficient global replication

Table 2.2: Related work on improving reliability, cost and efficiency of replicated storage system

Author	**Type of storage systems**	**Replication type**	**Objective**
Cidon et al. [79]	Cloud	Static	To reduce probability of data loss without any additional storage overhead and performance lag.
Hassan et al. [80]	Cloud	Static	To reduce access latency and to improve storage cost and availability without any additional storage overhead and performance lag.
Liu and Shen [81]	Cloud	Static	To improve availability with low storage and maintenance cost.
Long et al. [82]	Cloud	Static	To improve availability with high performance and energy efficiency.
Bonvin et al. [78]	Cloud	Dynamic	To improve availability guarantee at minimum cost.
Sun et al. [83]	Cloud	Dynamic	To improve performance and availability with high storage efficiency.
Qu and Xiong [84]	Cloud	Dynamic	To improve availability with high storage efficiency.
Hussein and Mousa [85]	Cloud	Dynamic	To improve reliability with minimum cost.
Boru et al. [86]	Cloud	Dynamic	To minimize network and energy efficiency.
Li et al. [35]	Cloud	Dynamic	To maintain reliability with low storage overhead.
Qu and Xiong [84]	Cloud	Dynamic	To reduce cost while maintaining high availability storage overhead.
Zeng and Veeravali et al. [87]	Cloud	Dynamic	To reduce response time.
Liu et al. [88]	Cloud	Dynamic	To ensure high durability with low cost.

algorithm (RFH) for distributed Cloud storage systems. The goal of RFH is to hold high replica utilization rate, high query efficiency and low cost while maintaining high availability. To achieve its goal RFH algorithm dynamically changes replica and location to meet different needs. This flexibly replicate data according to the changing query load and it replicates data into the nodes with high forwarding traffic.

Hussein and Mousa [85] 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 those, 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. A data replication technique to optimize energy consumption, network bandwidth and communication delay in cloud data centres are proposed in [86]. 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. [35] 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.

To improve system performance cloud storage systems maintain Meta Data Server (MDS) to perform metadata searching service. Sometimes cloud data centers maintain multiple MDS to improve performance. To determine number of MDS in cloud, Zeng and Veeravali et al. [87], proposed a strategy with an objective to reduce mean response time of metadata requests. Depending on the request rate the arriving at master MDS, the number of metadata replica of each object is determined. The proposed strategy reduces response time and balances MDS load by maintaining minimum replication.

Even though random three replication method is commonly used in cloud storage systems to ensure data durability, it fails to efficiently handle correlated machine failure.

Liu et al. [88] proposed a scheme called popularity-aware multi-failure resilient and cost-effective replication (PMCR) to ensure high durability of cloud storage in the presence of correlated failure while reducing storage and bandwidth cost substantially. PMCR splits cloud storage systems into primary and backup tier. It maintains three replicas of data, but it stores two replicas in primary tier and one in backup tier. It maintains exact copy of third replica in backup tier for hot data. However, it compresses the replicas of warm and cold data to store before backup tier to reduce storage and bandwidth cost.

2.5 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 techniques with respect to various parameters is detailed below.

Figure 2.9 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 2.9.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 2.9.b the reconstruction of a failed disk involves only three disk access to reconstruct three data fragments. However, in Figure 2.9.d reconstruction of the failed disk involves four disk accesses to reconstruct

two fragments. This increases the cost of reconstruction in an erasure coding system. Figure 2.10 depicts storage overhead and Figure 2.11 depicts the reliability in terms of Mean Time to Failure (MTTF) in years with correlated failure for both redundancy policies. Data from [11] are used to depict Figure2.10 and Figure2.11. 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 [89]. 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 [89]. In an erasure coded system, the failed object should be reconstructed from the next available objects, which increases the latency for the read request [89]. Moreover, costs associated with reconstruction is high in terms of bandwidth and disk I/O [90].

Several works compare replication and erasure coding [91, 92, 93, 89, 11]. These comparisons assume independence between parameters. Table 2.3 summarizes the comparison between replication and erasure coding. The keywords high and low are used to represent the superiority of one technique over other.

Figure 2.12 shows quantitative comparison of network bandwidth/traffic, durability/availability and storage overhead. The results shown in Figure 2.12 are generated using ds-sim [4] simulator. Figure 2.12(a) shows recovery network bandwidth and traffic of various redundancy policies and results are normalized against replication with factor 3. Figure 2.12(b) shows reliability of various redundancy policies in terms of number of unavailable and undurable objects during simulation time of ten years. Figure 2.12(c) shows storage overhead of various redundancy techniques.

From Figure 2.12, storage overhead of redundancy polices Reed-Solomon(6, 4) and Reed-Solomon(9, 6) are identical. Recovery bandwidth consumption of Reed-Solomon(6, 4) is less compared to replication. However, availability offered by Reed-Solomon(6, 4) is less than Reed-Solomon(9, 6). Reed-Solomon(14, 10) offers high reliability with less storage overhead. However, recovery bandwidth consumptions are substantially high than

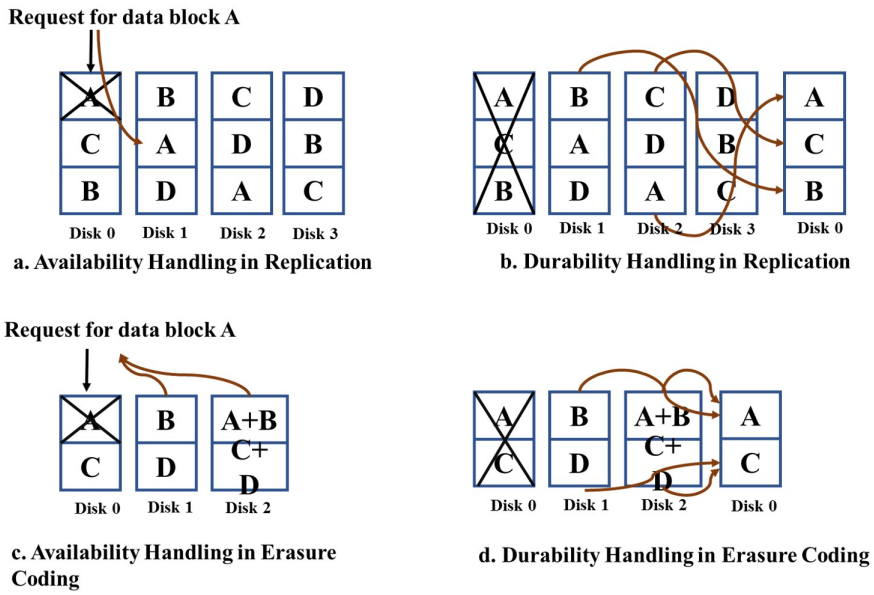


Figure 2.9: Durability and availability handling in replication and erasure coding techniques.

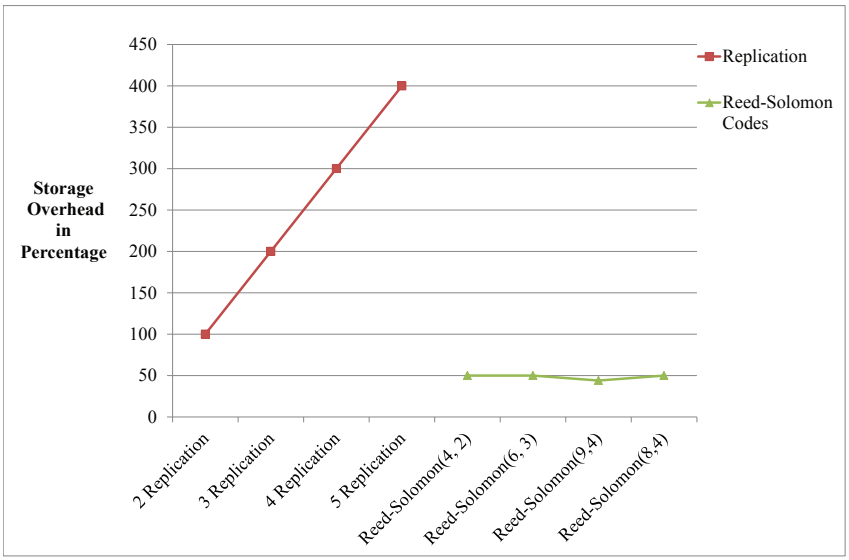


Figure 2.10: Storage overhead (percentage) of various redundancy policies [94].

other redundancy techniques. Storage overhead of 2 replication is 2x which is more than other coding techniques, but still it offers very low reliability compared to other

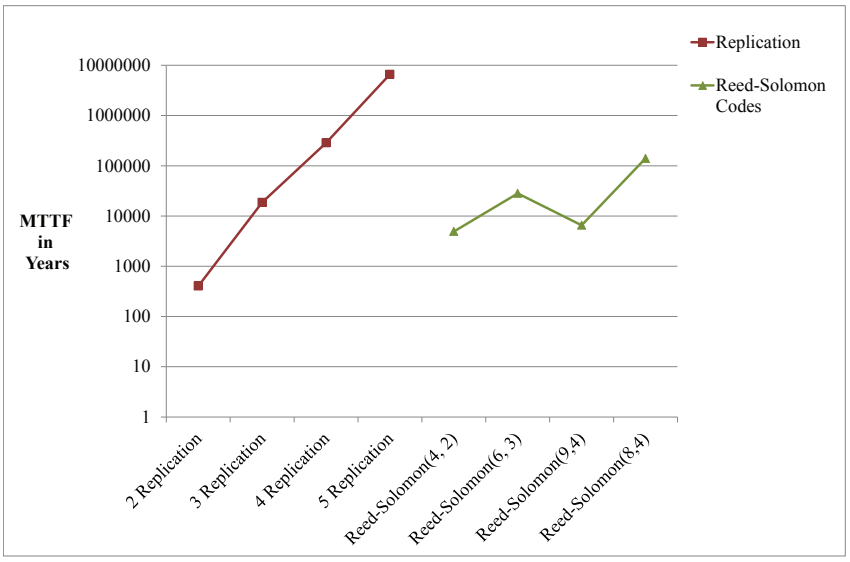


Figure 2.11: MTTF in years with correlated failures for various redundancy policies [94].

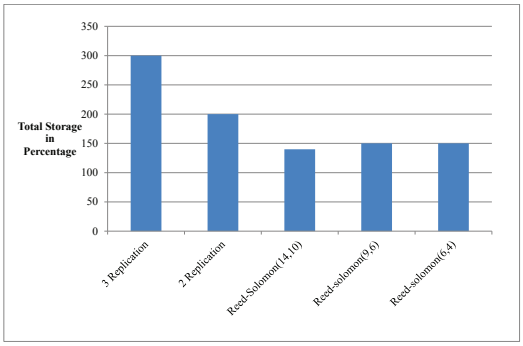
Table 2.3: Comparison between replication and erasure coding

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

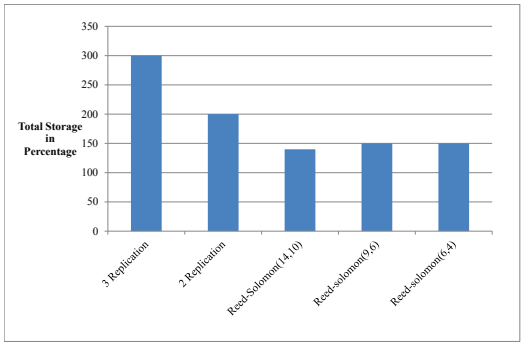
methods. Recovery bandwidth consumption of coding methods are substantially high than replication methods.

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

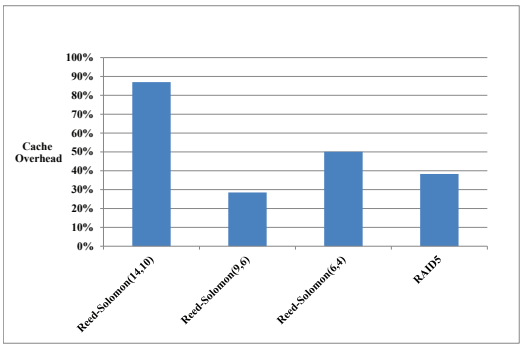
As failures are frequent in cloud storage system, data redundancy is employed in cloud storage systems to handle failures. Replication is simple solution to improve data reliability. But replicating terabytes and petabytes of data increase the storage overhead



(a)



(b)



(c)

Figure 2.12: Quantitative comparison between various redundancy polices (a) Normalized network bandwidth and traffic with respect to Replication (b) Reliability in terms of number of durable degraded and available degraded objects over 10 years (c) Storage overhead of various redundancy policies.

drastically. Nowadays erasure coding is gaining traction because it offers huge savings in terms of storage with 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 2.1) 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 improve performance of erasure codes like the performance achieved with replication for Big Data applications.

Some studies (Table 2.2) have focused on minimizing number of redundancies in replicated storage systems to improve storage efficiencies. None could 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 2.4. Araujo et al. [95] 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. [96] 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, it needs to reconstruct the data upon block unavailability. Li et al. [97] presents proactive erasure coding (ProCode),

Table 2.4: Related work on improving reliability, cost and efficiency of replicated storage system

Author	**Type of storage systems**	**Objective**	**Method**
Araujo et al. [95]	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 in the network.
Ma et al. [96]	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. [97]	Cloud	Reduce latency (In terms of network bandwidth and disk I/O)	Adjust replication factor of data based on drive failure prediction.

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.

2.6.1 Cloud Storage Classes

The most popular cloud storage systems like Amazon, Azure and Google Cloud offers several classes of storage [98, 99]. Pricing of storage classes are different and they define different limits on important metrics of cloud storage such as durability, availability, access latency and throughput. They also enable life cycle policies to automatically migrating data between storage classes. The life cycle management tool migrates data into different storage classes based on the time limit defined in life cycle configuration rules. They support object level migration.

Data objects can be classified as hot, warm or cold based on data access pattern. Data access pattern may change with different applications. Some applications may frequently

change data access pattern or access pattern may not be defined. Amazon Web Services (AWS) have designed a storage class S3 Intelligent-Tiering for the data storage with unknown or changing access pattern [100]. This stores objects in two access tiers: one for frequent access object and other for objects that have infrequent access. Objects in S3 Intelligent-Tiering are monitored for the change in access pattern. According to the access pattern change, it automatically moves data into appropriate tiers.

2.6.2 Energy Efficiency of Cloud Storage Systems

Storage systems are one of the most important energy consuming components in cloud computing [26]. Energy efficiency methods used in data centres save operational costs and help to conserve the environment [101]. The energy efficiency of storage systems is highly dependent on read/write latency [20]. Pinheiro et al. [102] 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 energy related to disk. Harnik et al. [19] 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. [101] 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. [103] 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. [35] proposed a link rate controlled data transfer (LRCDT) strategy for energy efficient data transfer in replication based cloud storage systems.

2.6.3 Research Directions

After extensive and careful literature survey the following directions are derived to proceed with next chapter.

- *Efficient Big Data storage:* Erasure coding can define a storage efficient platform for Big Data. However, it is not bandwidth efficient and performance efficient due to the inevitable failures of storage systems. Replication is not a viable option to improve the reliability of Big Data regardless of its exceptional performance in any existence of failures. An efficient system can be carefully defined to utilize both of its benefits. The system must apply the benefits of replication to address the bandwidth consumption and performance issues of erasure code. The extra replicas should be created to address the data reconstruction issues of erasure codes. To control the extra storage requirements of erasure codes, replicas should be created only when it is required. Failure prediction techniques can be utilized to define such replicas.

- *Cognitive Big Data storage:* Read performance is an important property of Big Data. A storage system must offer exceptional read performance to support velocity property of Big Data. However, data in a storage system may have different access patterns. Based on the access patterns, data can be classified as hot or cold. Data access pattern can be used to classify data as hot or cold. A delay in accessing cold data is acceptable some time. If the client accepts the delay, it can be recorded in client SLA. A cognitive storage can be defined to maximize bandwidth saving by applying lazy recovery to cold data. It must wisely use client SLA and data access pattern to apply lazy recovery.

- *Expeditious Big Data storage:* Cache is a perfect solution to improve the data access speed to support velocity of Big Data. However, cache may be ineffective sometimes. Cache can also be utilized appropriately to improve data access speed when there is a failure in underlying storage. Expeditious storage can be defined with erasure codes by utilizing cache to support data read velocity while also reducing storage overhead to the minimum.

2.7 Summary

Cloud computing is playing a predominant role to serve Big Data applications as it provides cost-effective, on-demand services. Cloud enables storage and computing resources to be scaled up and down rapidly based on client's storage and computation demand. As failures are becoming the norm in cloud storage systems various fault tolerant mechanisms have been employed in cloud storage systems to improve data reliability. Erasure coding is the favourable choice of cloud storage systems to improve reliability of Big Data. However, data reconstructions due to failures demand more resources which affect performance of the applications. This prevents cloud storage systems to move towards erasure coding. In this chapter, state-of-the-art of both techniques is discussed. In erasure coded storage systems, various techniques are highlighted to reduce resource consumption during data repair. In replication storage, several existing researches on improving data reliability with minimum replications are discussed. Also, this chapter highlighted several existing hybrid techniques on improving data reliability.

With highlighted research directions in this chapter, next chapter proposes several novel proactive recovery techniques to mitigate resource usage due to failures in erasure coded storage system. They are defined utilizing hardware failure predictions. We also propose a system to accommodate proposed recovery techniques and select an appropriate recovery technique among them to meet client SLA in the efficient manner.

Chapter 3

Adaptive Bandwidth Efficient Cloud Storage Systems

Replication and erasure codes are the most important data reliability techniques employed in cloud storage systems, but individually they have their own challenges. Challenges of replication and erasure coding were discussed in the last chapter. Subsequently, possible research directions were also highlighted in the last chapter. In this chapter, a novel system is proposed to define a cost effective reliable storage system for Big Data. This reliable storage system is designed to improve storage cost by applying erasure coding and bandwidth cost by applying replications to the necessary erasure coded chunks. Failure predictions are utilized in this system to identify the necessary blocks for replication. To maximize the resource savings, the proposed system employs several novel proactive recovery methods for erasure codes. When the system predicts any hardware failures, it will select one of the most appropriate proactive recovery techniques which can meet client SLA and data access patterns.

This chapter is derived from: Nachiappan, R., Javadi, B., Calheiros, R. N., & Matawie, K. M. (2018, August). Adaptive Bandwidth-Efficient Recovery Techniques in Erasure-Coded Cloud Storage. In European Conference on Parallel Processing (pp. 325-338). Springer, Cham.

3.1 Introduction

Hardware failures (disk failures, machine failures, and latent sector errors) and temporary machine failures are the most common failures that affect durability and availability of data in cloud storage [3]. In order to avoid permanent data loss due to hardware failures, contents in failed nodes or disks have to be restored in an another hardware device, a process that is known as *data recovery*. Data stored in an unavailable machine due to temporary outage will cause temporary data loss. Temporary data loss in erasure code is handled by degraded read. In degraded read, data blocks in the failed node are reconstructed and served using the next available k blocks. In order to avoid unnecessary repairs due to short term transient node failures, data recovery is delayed for a certain amount of time. Google File System (GFS) delays recovery of data from unavailable nodes for 15 minutes. However, this affects availability and degraded read performance [104]. In contrast, when replication is used, degraded read is handled by simply redirecting the request to the next available replica.

Repair network bandwidth hike is one of the most important issues of erasure coding. Repairing a single data block stored using Reed-Solomon(n, k) code requires k data blocks to be transferred over the network. However, repairing a single data block in replication involves the transfer of one data block [105]. Repair network bandwidth is increased by k times in Reed-Solomon(n, k) code compared to replication. The network traffic incurred due to such data movements increase network switch energy consumption resulting in extra costs for cloud service providers. Moreover, network traffic is regulated by network throttling, which affects read performance. All the above facts prevent cloud storage systems to apply erasure codes in large scale.

As both replication and erasure coding have its own advantages, cloud storage systems require hybrid approaches in order to leverage the advantages of both methods. In this chapter, we propose several novel recovery techniques. These techniques replicate certain data chunks of erasure coded data. They utilize data access patterns and hardware failure predictions to improve repair bandwidth savings with minimal storage overhead. We have also showed that the ProDisk method proposed by Li et al. [76] reduces repair network

bandwidth/traffic. All the aforementioned methods must utilize machine and disk failure prediction techniques to predict disk failures and long-time temporary machine outage. When hardware failures (permanent machine/disk failures) are predicted, proposed storage system immediately replicates all the data chunks in failure devices in to permanent storage. When long-term machine failures are predicted, the proposed storage system applies various recovery using data access pattern and client SLA. During proactive recovery of long-term machine failures, data is written into dedicated temporary storage for the quick reference to remove them when it is no longer required.

In a distributed storage system, a data file is dispersed into multitude of interconnected nodes, which serves any end user request by tapping data from multiple nodes. Improving the resilience of distributed storage system with limited storage overhead is desirable. Erasure coding offers high reliability with less storage overhead. Reducing repair network traffic/bandwidth in erasure coding is important to make it more pervasive in cloud storage systems. Applying a delay in erasure code can reduce repair network traffic/bandwidth significantly. However, this may compromise availability and read performance. Proactive replication of failure predicted data in erasure coding can significantly reduce resource usage due to repair but it comes with the cost of additional storage overhead. An object can be categorized as cold, hot or warm according to the data access pattern of that object. For an object in online social network, there is a strong correlation between age and access pattern. An object uploaded to online social network receives more I/O during its early lifetime [106]. Number of I/O will eventually reduce over the time. While hot objects always demand high availability and performance, some relaxation is acceptable for cold objects.

The amount of temporary storage required in the proposed approach is linearly related to the number of long term machine failures predicted over a period of time. To reduce temporary storage overhead due to proactive replication, the proposed system switches between proactive and lazy recovery. The system utilizes the data access pattern to identify hot data. It applies proactive replication to all hot data since read performance of hot data should not be compromised. It checks client SLA before applying lazy recovery to cold data. Even though bandwidth and storage savings can be maximized by applying

lazy recovery to cold data, client SLA must be verified to check the client's acceptance on access delay of cold data caused due to lazy recovery.

Novel block chain-based cloud storage systems like Storj [107] uses consumer storage to serve their customer's storage needs. They suggest, as a means to improve reliability, the use of Reed-Solomon(60, 40) code. This means that, to reconstruct any missing data, 40 surviving data fragments have to be transferred to reconstruct any single failed data fragment. These novel storage systems demand more bandwidth-efficient recovery, which is the focus of this chapter. The proactive recovery techniques proposed in this chapter use several failure prediction methods. As these systems are running on end-users client, it may not be possible to apply existing hardware failures prediction techniques on the user's computers. However, it is possible to predict the availability of user computers using availability logs. Hence it is possible to apply some of the proposed techniques in blockchain-based cloud storage systems.

Using data access pattern of objects, several bandwidth-efficient recovery techniques are defined in this chapter. They use very limited temporary storage overhead. ProMachine, ProHot, ProHot_LazyCold are the novel methods proposed in this research which are the main contribution of this chapter.

3.2 Related Work

A substantial amount of research concentrated on reducing repair bandwidth of erasure codes. Dimakis et al. [108] presented a theoretical framework for regeneration codes that can optimize recovery bandwidth for a given storage. However, exact repair of regeneration codes, matching information theoretic bound, remained unresolved. Following this, several works [3] showed that exact repair is possible for some parameters. Sathiamoorthy et al. [109] proposed Xorbas which reduces network traffic by half compared to Reed-Solomon codes with 14% additional storage overhead [109]. LRC in Windows Azure storage reduces repair network bandwidth significantly with the help of local parities, which have the side effect of increasing storage overhead by 1.33x compared to Reed-Solomon [110]. Hitchhiker code, built on top of Reed-Solomon code using piggybacking framework, reduces network traffic by 35% with some encoding time overhead incurred

[111].

Failure predictions in cloud storage systems offer cloud service providers an efficient proactive failure management in cloud storage. Various statistical and machine learning methods are used to predict failures in cloud storage systems. A few methods [112, 113] are used to predict hard drive failures based on SMART attributes. Li et al. [113] achieved 95% predictions with False Alarm rate less than 0.1%. Many researches had focused on predicting failures in distributed systems based on system logs. Javadi et al. [114] presented failure model as a predictive method of distributed systems availability and unavailability. Agarwal et al. [115] uses log messages to predict failures in Hadoop clusters.

Silberstein et al. [4] proposed lazy recovery to reduce recovery bandwidth in distributed storage by reducing the recovery rate. It reduces recovery bandwidth up to 76% compared to Reed-Solomon. However, applying this method on cloud storage affects read performance and data durability. Li et al. [76] used failure prediction techniques to implement proactive replication in erasure codes for reducing degraded read latency and improving read performance. Li et al. [77] defined a cost effective data reliability management mechanism to ensure reliability of massive data with minimum replication based on a generalized data reliability model. Wu et al. [116, 117] used prediction tools to identify the upcoming events to proactively migrate the data blocks on the degraded device belonging to the hot data zones in the large-scale data centers.

3.3 The Proposed Cloud Storage System

The target system here is an object storage that initially stores data with any appropriate erasure code to reduce storage overhead while maintaining reliability. Consider a distributed cloud storage system composed of a number of disks accommodated in a machine, group of machines in a rack, and several racks in a distributed storage. Data blocks stored in a disk can be determined as a *at-risk block* based on the underlying machine and disk health status. Machine and disk failure prediction algorithms run individually to predict disk or permanent machine failure and machine unavailability. Since rack failures are transitory, the health of data blocks is determined with machine and disks health

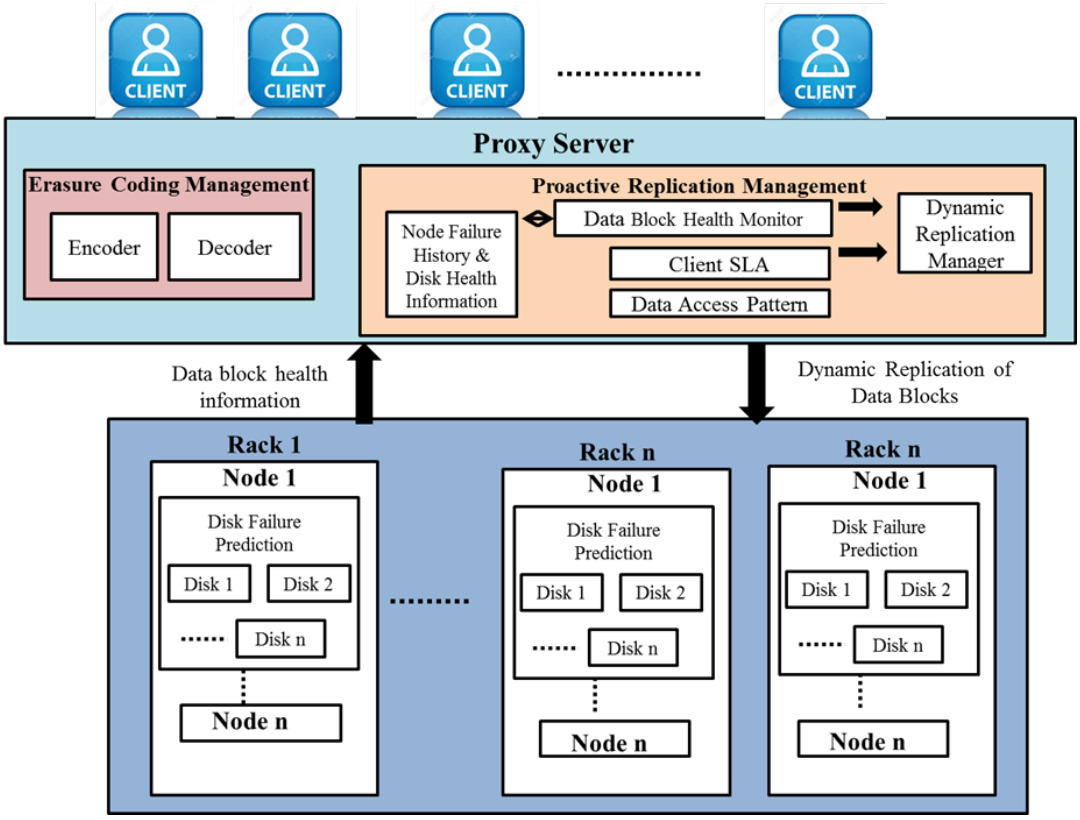


Figure 3.1: The System architecture for proposed recovery techniques.

status. Data blocks that are marked as *at-risk* in this system are proactively replicated before the occurrence of failure based on the client's Service Level Agreement (SLA). Proactive replication reduces the number of blocks required for reconstructions in erasure coded cloud storage system. Hence, the proposed system reduces network traffic with less storage overhead. This system utilizes various recovery schemes to reduce reconstruction bandwidth in erasure coded cloud storage systems.

3.3.1 Architecture and Design

An overview of the system architecture is depicted in Figure 3.1. It is implemented as an extension of regular object storage. Object storage manages data as objects where each object has both data and metadata. A dedicated proxy server extends the support of encoding and decoding erasure codes. It also handles failures in storage systems. The object server stores and retrieves object data. Object server's availability status and

disks health status are reported to the proxy server, which is responsible for increasing or decreasing the data object's replication factor. The system adjusts the replication factor of erasure coded objects when failures are predicted. The components of the architecture are discussed as follows.

Disk Failure Prediction

This module monitors the health status of individual disks and reports prediction results to the Node Failure History & Disk Health Information module in the proxy server. SMART is implemented on disks and it monitors, compares disk attributes and issues warnings. This SMART attributes are used to predict disk health status using various statistical and machine learning techniques [113, 112]. Disk failures are calculated using classification and regression trees methods here [113].

Proactive Replication Management

Popular storage systems like AWS and Azure migrates data into different storage classes according to the changes in data access pattern. Client SLA with important metrics of such as durability, availability, access latency and throughput varies with different storage classes. Since erasure coding is a storage efficient method to improve reliability of cloud storage, instead of moving objects into storage classes, we enforce different recovery methods to improve cost savings of cloud storage. Redundancy of data blocks are adjusted according to node/disk health status, client SLA and data heat.

Node Failure History and Disk Health Information This module collects the information of disk health status and node failure history. Various statistical and machine learning techniques can be used to predict node's Mean Time To Failure (MTTF) and Mean Time To Repair (MTTR). Based on node's predicted MTTF and MTTR, node failures are classified as permanent, long time, or short time failures. Node's MTTF and MTTR are calculated using various statistics of availability and unavailability [114].

Data Block Health Monitor & Client SLA Failure predicted nodes and disks information are collected from Node Failure History and Disk Health Information module. It identifies the disks that are predicted to fail in the underlying storage system. It also

identifies permanent, long term and short term machine failures by predicting machines MTTF and MTTR. Permanent machine failures are handled as disk failures. This module sends failure information to the Dynamic Replication module to take necessary actions. Clients can request various recovery schemes based on their needs. The client can define their requirements as follows:

- High durability, normal availability.
- High durability, high availability.
- High durability, high availability for hot and normal availability for cold data.
- High durability, high availability for hot and low availability for cold data.

Based on the client SLA, the variable that represents different recovery scheme will be set.

Data Access Pattern Data access patterns in a distributed storage can be analysed over a certain period of time to identify the popularity of data blocks in real-time. Based on their popularity, data blocks can be classified as hot, warm, or cold. As the access pattern changes, popularity of data blocks need to be updated. Various researches used popularity-based classification to improve durability, availability, and read performance of cloud storage systems [118]. Our approach combines both failure prediction and data access patterns to decide the recovery type of an object. Data access pattern is used here to define hot data. We assume that data blocks with high access frequency have more chance to be accessed in the future and they are defined as hot. This module uses data access pattern to classify a block as hot data block and they are grouped as a set $H = \{b_1, b_2, ...\}$ where the block b_i in H is hot.

Dynamic Replication Manager This module collects information from Data Block Health Monitor, Client SLA, and Data Access Pattern module and activates various proposed recovery schemes, as follows:

- **ProDisk:** When disk failures or permanent machine failures are predicted, all the data blocks in the failure predicted disks (all disks in failure predicted machine)

are proactively replicated permanently as described in [76]. In the occurrence of failure, the reference is made to the proactively replicated data instead of applying typical reconstruction of erasure codes. This was originally proposed by Li et al. [76] but they only considered the recovery performance not recovery bandwidth. This method improves data durability and will provide limited contribution on improving data availability. It will reduce degraded read due to disk failures and hence it will improve read performance.

- **ProMachine:** Machines are the important components that fail more often in cloud storage [2]. Best bandwidth saving can be achieved by proactively handling long term machine failures. When temporary long term machine failures are predicted with MTTR greater than 15 minutes, data blocks in failure predicted machines are proactively replicated into a dedicated node that is allocated specifically to handle temporary machine failure. In case of any failure, data is accessed from the dedicated node. It will improve durability, availability and read performance.

- **ProHot:** This method periodically identifies hot data blocks and applies proactive recovery only for hot data blocks. When temporary long term machine failures are predicted with MTTR greater than 15 minutes, data identified as hot in failure predicted machine are proactively replicated into the dedicated node. In case of any failure, hot data is accessed from the dedicated node and typical reconstruction is applied to recover cold data. This will improve durability of all objects. This will also improve availability and read performance of hot data.

- **ProHot_LazyCold:** In case of any temporary long term machine failure prediction, it is unnecessary to reconstruct cold data if it is not going to be accessed soon. When temporary long term machine failures are predicted with MTTR greater than 15 minutes, data identified as hot in failure predicted machine is proactively replicated into a dedicated node that is allocated specifically to handle temporary machine failure. In case of any failure, hot data is accessed from the dedicated node and lazy recovery [4] is applied for cold data recovery. Hence it saves temporary storage overhead compared to ProMachine alongside significant bandwidth savings.

This method will improve durability all objects. This method will also improve availability and read performance of hot data. However, it will impact the read performance of cold data since it will apply lazy recovery of cold data.

Dynamic Replication Manager is also responsible for scaling up and down the number of dedicated temporary storage nodes, according to the failure predictions and amount of data need to be stored in temporary storage during a period of time. It is also responsible for allocating highly available node as a temporary storage such that any failure in this temporary storage node is minimal. Any failure prediction in this temporary storage will also lead to proactive replication.

3.3.2 Proactive Recovery Approach

In our target scenario, a cloud storage system initially stores data with any (n, k) erasure code. With the help of disk and machine failure prediction methods employed in cloud storage systems, failure types and MTTR of node failures are predicted. Failures are also identified as disk, permanent machine, temporary long term machine (MTTR>15 minutes), or temporary short term machine (MTTR<15 minutes) failures. The set of data blocks $(b_1, b_2, ..., b_i)$ that is more likely to be accessed soon is defined as the hot data set H. Based on the failure types, hot data blocks, and client SLAs, one of the proposed recovery techniques ProDisk, ProMachine, ProHot, ProHot_LazyCold will be chosen.

When the disk or permanent machine failures are predicted (ProDisk), all the data blocks in the failure predicted disk (all data blocks of each disk in a failure predicted machine) are proactively replicated into the permanent storage as described in Procode [76]. The counter variables of corresponding replicated data blocks are incremented. These counter variables are used to identify if the particular data blocks associates proactive copy. It is also used to delete data blocks against noisy prediction. A delay is applied while deleting data blocks against noisy prediction. Time In Advance (TIA) provided by failure prediction algorithm is used as a time delay to delete the data blocks that are replicated due to noisy prediction. Time delay being larger than TIA is the better choice. However, this will result in extra storage. The choice of time delay varies and depends on the storage system where the system is utilized.

While temporary machine failures are predicted, proactive recovery is activated for either all (ProMachine) or some of the data blocks (ProHot, ProHot_LazyCold) in a failure predicated machine. They are replicated into the dedicated temporary storage. The data blocks that are not replicated proactively are recovered by typical reconstruction of erasure codes or using lazy recovery. While data blocks are proactively replicated into temporary storage, the corresponding data blocks counter variables are incremented. These variables are used to identify if the particular data blocks are replicated already. They are later referred to delete those blocks when that machine recovers from its temporary failures. The dynamic replication module also provisions and adjusts the number of temporary dedicated nodes, based on long term temporary machine failure rate and client SLA. When the failure predicted nodes recover from actual failure provide no further failures are predicted for the same nodes, the proactively replicated data blocks corresponds to those nodes are deleted. Also, any data fragments which have more than one copy in the system are also deleted periodically. In the occurrence of node/disk failure, the reference is made to proactively replicated blocks which reduce number of data reconstructions in erasure coded storage systems.

3.4 Adaptive Proactive Recovery Algorithm

This algorithm introduces different recovery schemes based on client SLA agreement using failure prediction techniques. Temporary machine failures occur more often in large-scale object storage compared to permanent machine/disk failures. Temporary machine failures are contributing more to average recovery bandwidth. Thus, it is necessary to pro-actively handle the recovery due to temporary machine failures.

The algorithm also needs to account for temporary failures. For example, GFS initiates recovery of data on unavailable nodes after 15 minutes in order to reduce unnecessary recovery. Our approach uses predicted MTTR of failure predicted nodes and ignores short-term temporary machine failures. In some applications, the only concern is on the availability/latency of hot data blocks. In this case, the recovery of cold data can be delayed until certain amount of fragment fail (a process called *lazy recovery*) to increase the availability of resources for other operations. The algorithm activates proactive replication

to optimize recovery bandwidth in object storage.

Lines 6-11 in the Algorithm 1 define how disk failures/permanent machine failure predictions are handled in the system. When disk or permanent machine failures are predicted, all the data blocks in the failure predicted machine/disk are pro-actively replicated. Lines 12-39 define how transient machine failures are handled in the proposed system. In the occurrence of machine failure prediction, the algorithm defines several proactive recovery strategies based on client SLAs. It activates proactive recovery for either all or partial data blocks based on client SLA. If the data blocks that are not replicated, it is recovered by typical reconstruction of erasure codes in ProMachine and ProHot. However, they are recovered by applying lazy recovery in ProHot_LazyCold. While data blocks are pro-actively replicated, the corresponding blocks counter variable is incremented. It is used to identify if the particular data blocks are replicated already or to delete blocks when the disk or machine recovers from failures.

The dynamic replication module also provisions and adjusts the number of temporary dedicated nodes based on long term temporary machine failure rate and client SLAs. In the occurrence of node/disk failure, the reference is made to pro-actively replicated blocks. To reduce storage overhead, when the failure predicted machine/disk did not fail or when the node recovers, the copy of the fragment is deleted. Also, if more than one copy of the particular fragment exists in the system, it is also deleted. A delay is applied when deleting data blocks. The choice of time delay varies and depends on the storage system where the system is utilized.

3.5 Performance Analysis

All the methods proposed in this chapter use a combination of proactive, typical and lazy recovery methods. Each proposed recovery methods show various savings in terms of bandwidth and storage. To analyse bandwidth and storage savings of those methods, we will carry out the performance analysis of various reliability in this section.

Algorithm 1 Dynamic Replication Algorithm

```
Predicted Failures,Failure Type, MTTR of node failures
H ← B_i (Set of all hot data blocks)
SLA ← ProDisk, ProMachine, ProHot, ProHot_LazyCold
procedure DYNAMIC REPLICATION(PredictedFailure)
    if Failure Type is disk then
        for each slice s_i in failure predicted disk do
            identify the block b_i to which fragment s_i belongs
            Proactively replicate s_i of block b_i
            copy[s_i]++
        end for
    else if Failure Type is Machine then
        if MTTF >15 minutes and SLA is ProDisk then
            activate normal recovery
        else if MTTF >15 minutes and SLA is ProMachine then
            for each disk in failure predicted machine do
                identify the block b_i to which fragment s_i belongs
                Proactively replicate s_i into temporary node
                temporaryCopy[s_i]++
            end for
        else if  MTTF >15 minutes and SLA is ProHot then
            for each disk in failure predicted machine do
                identify the block b_i in which fragment s_i belong to
                if  block b_i belongs to H then
                    Proactively replicate s_i into temporary node
                    temporaryCopy[s_i]++
                else
                    activate typical reconstruction of erasure codes
                end if
            end for
```

```
30:         else if MTTF >15 minutes and SLA is ProHot_LazyCold then
31:             for each disk in failure predicted machine do
32:                 identify the block b_i to which fragment s_i belongs
33:                 if  block b_i belong to H then
34:                     Proactively replicate s_i into temporary node
35:                     temporaryCopy[s_i]++
36:                 else
37:                     activate lazy reconstruction of erasure codes
38:                 end if
39:             end for
40:         end if
41:     else
42:         activate lazy reconstruction of erasure codes
43:     end if
44: end procedure
```

3.5.1 Bandwidth Analysis

The bandwidth required to reconstruct any missing data is directly proportional to the number of transfers required, which is k times of a chunk size in (n, k) erasure coded storage system. The amount of data transfer required to recover any missing block is calculated as

$$TransferRequired = S * (k + NumberOfMissingBlocks - 1) \tag{3.1}$$

Where S is the chunk size and k is number of fragments needed to reconstruct data. The k is 1 for replication. Hence the recovery bandwidth is calculated as

$$RecoveryBandwidth = TransferRequired/RecoveryTime \tag{3.2}$$

Equation 3.2 shows that the RecoveryBandwidth is directly proportional to TransferRequired. Let us consider Reed-Solomon(14, 10) code with the chunk size of 250MB.

From equation 3.1, TransferRequired can be calculated as 2500MB for recovering a single missing data block in Reed-Solomon(14, 10). However, it is 250MB if the data block is proactively replicated. From this, we can conclude that proactive replication reduces the recovery bandwidth significantly. Lazy recovery delays the recovery of the data fragments until certain amount of data fragments (recovery threshold) are unavailable. For example, in Reed-Solomon(14, 10), if recovery threshold for lazy recovery defined as 12, recovery will not be activated for any objects until number of degraded slices in an object becomes 2. Hence lazy recovery reduces repair rate in erasure codes and reduces recovery bandwidth and traffic. In this chapter, we use lazy recovery only for handling long term temporary machine failures such that it does not impact durability of data. Since the entire predicted disk failures are proactively replicated, it does not affect durability. Furthermore, lazy recovery is activated based on data access pattern of objects and client SLA. If the client needs good read performance only for data identified as hot, it activates lazy recovery only for cold data. Proactive recover is activated for all data blocks that are identified as hot.

3.5.2 Storage Overhead Analysis

Erasure coding offers excellent storage efficiency compared to replication. Proportional increase in storage of various reliability methods is defined as:

$$(systematicdata + originaldata)/originaldata \tag{3.3}$$

The method proposed in this chapter proactively replicates data into a new hardware device when permanent node/disk failures are predicted. Once the predicted device fails, reference will be made to the proactively replicated device. Eventually, there will be wrong predictions about devices failing. When this occurs, it is expected that the storage overhead will suffer a slight increase. False positive for disk failures are calculated as less than 0.1% using classification and regression trees [113]. Hence, the storage overhead will not be significantly increased by wrong predictions. Temporary nodes are dedicated to handle long term node failures. However, data in those temporary nodes are periodically evicted. Hence, temporary node failures will not increase storage overhead permanently.

Moreover, the life time of data that are replicated due to failure predictions are from the time of prediction till actual failures in case of true positive. However, it is from time of prediction till TIA+delay in case of false positive.

3.6 Performance Evaluation

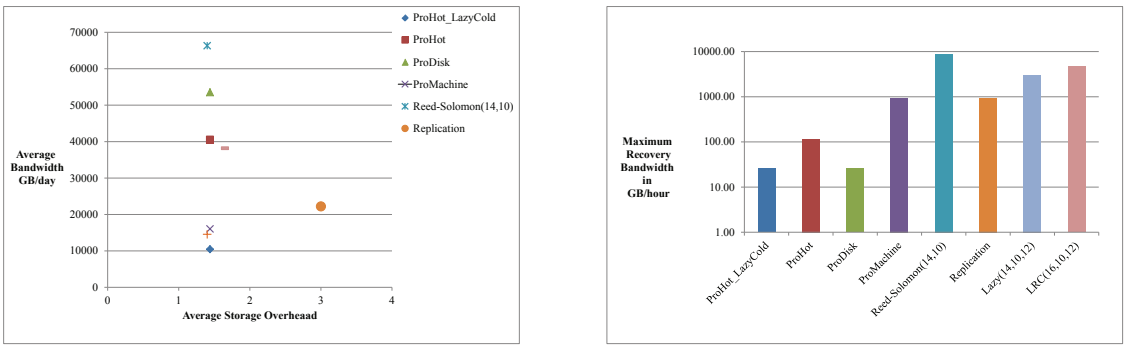
We use ds-sim simulator [4] to compare recovery bandwidth from replication and erasure coding to the various bandwidth efficient recovery techniques proposed in this chapter. The ds-sim is a distributed storage simulator which simulates failures using traces and models. It simulates 3-tier storage components including disks, machines, and racks. Disk failures can be latent or permanent. Latent disk failures are detected and recovered during periodic reads. Permanent disk failures are assumed to be unrecoverable. Machine failures can also be transient or permanent. Recovery from transient failures begins after 15 minutes and immediately for permanent failures. Rack failures are considered as transient. The ds-sim records number of degraded reads and repair bandwidth. We have modified ds-sim to add failure predictions, proactive replication, and hot data prediction. The modified ds-sim calculates repair bandwidth and number of degraded strips of various reliability techniques and proposed proactive recovery methods. The simulator models distributed storage systems of 3 Petabyte of storage for 10 years. Simulation parameters are 11 machines/racks, 20 disks/machines, each disk with capacity 750 GB and maximum recovery bandwidth capacity of 650 TB/day. Also 40% of random data blocks was considered as hot to evaluate ProHot and ProHot_LazyCold recovery methods. For each result we run the simulation with number of iterations and calculated the result with 95% confidence interval.

3.6.1 Results and Discussions

In this section, we compare the bandwidth and reliability of replication, Reed-Solomon(14, 10) and various recovery techniques proposed in this chapter.

Recovery Bandwidth.

We run simulations with the above experimental setup with failure prediction rate 90%, false positive 0.1%, and TIA 24 hours which found reasonable in [113, 115]. Recovery



(a) (b)

Figure 3.2: (a) Average recovery bandwidth in GB per day and (b) Maximum instantaneous recovery bandwidth, in MB/hr, calculated over 10 years.

bandwidth is calculated for each failure event except for machine failures lasting less than 15 minutes. Figure 3.2 shows the comparison of average recovery bandwidth in GB/day versus storage overhead for replication, Reed-Solomon(14,10), Lazy [4], and the various recovery techniques proposed in this chapter. The proposed recovery techniques are applied on Reed-Solomon(14, 10) erasure code in this comparison.

Replication reduces recovery bandwidth in up to 66% compared to Reed-Solomon(14, 10). ProDisk reduces average repair bandwidth up to 19% compared to Reed-Solomon(14, 10). ProHot reduces recovery bandwidth up to 38% whereas ProMachine reduces recovery bandwidth by 75% compared to the same approach. Reduction in recovery bandwidth is directly proportional to number proactive replication for ProDisk, ProMachine and ProHot methods. Data repair activated due to ProDisk is less since number of hard disk failures are limited compared to number of temporary machine failures. Moreover, ProMachine and ProHot apply proactive replication due to any disk failures along with proactive replication due to any machine failures. Bandwidth savings of ProHot_LazyCold is outstanding compared to other methods since it saves bandwidth in two ways. One is bandwidth savings due to proactive replication and other is bandwidth savings in terms of lazy recovery. ProMachine and ProHot_LazyCold outperform replication. This is because in replication, data blocks are distributed among large number of hardware devices. Hence it experiences a large number of recovery events that increases recovery bandwidth

compared to ProMachine and ProHot_LazyCold. ProHot_LazyCold outperform lazy recovery. This is due to failure predicted hot data blocks are replicated proactively which eventually lead to lazy recoveries. However, ProMachine technique increases the temporary storage proportionally to the temporary long term machine failure rate. ProDisk's bandwidth savings compared to LRC is limited, since it proactively replicates less data blocks.

Figure 3.2(b) shows the maximum instantaneous recovery bandwidth, in MB/hr (network traffic) in distributed storage systems over the simulation period of 10 years. The simulation calculates network traffic as follows. Upon each recovery event, instantaneous total recovery bandwidth, in MB/hr is calculated and compared with the previous maximum recovery bandwidth. If the new recovery bandwidth is larger than maximum recovery bandwidth, the new recovery bandwidth becomes the maximum recovery bandwidth.

ProDisk reduces the network traffic to the same level as replication. ProMachine and ProHot reduce network traffic even below replication. However, maximum recovery bandwidth in ProHot and ProMachine is higher compared to ProDisk. This spike is due to the network bandwidth required to proactively replicate all the data blocks in the failure predicted machine.

Reliability.

Durability and availability are always evaluated as hundred percentage for various data recovery approaches and Reed-Solomon(14,10) according to the simulation results. Since reliability of Reed-Solomon is high to calculate very limited compromise on durability and availability, number of iterations should be increased substantially. Reliability of proposed techniques are higher than Reed-Solomon(14,10) due to proactive replication. To evaluate reliability of different approaches using ds-sim, we use the number of durable degraded slices and available degraded slices to compare durability and availability over the mission time. In a distributed storage system, disks are partitioned into units called strip. Set of corresponding strips from n disks that encode and decode together is called stripe [119]. A stripe is termed degraded if one or more systematic blocks is unavailable. The term durable degraded refers the degraded stripe due to permanent failures, whereas

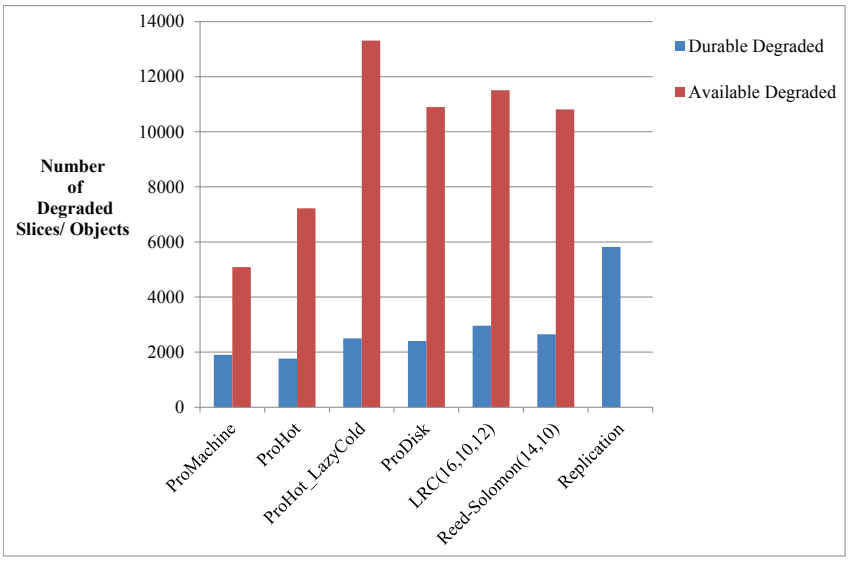


Figure 3.3: Average number of durable degraded and available degraded slices in a day.

available degraded refers to transient failures.

Replication does not increase available degraded slice counts in the system as request to any temporary unavailable slices are redirected to next available replica. Smaller number of durable and available degraded stripes indicates smaller probability of data loss as the system has less number of failure and repair events. Moreover, smaller number of degraded slices reduces access latency and increases the performance of the application running on it. From Figure 3.3, ProHot and ProHot_LazyCold methods do not decrease number of available degraded stripes. However, available degraded slices are increased with respect to cold data. Also, the proposed system predicts and handles disk and node failures separately. ProHot and ProHot_LazyCold methods handle all failure predicted disk failures proactively. Hence, they do not affect durability. Number of degraded slices in LRC(16, 10, 12) is more than Reed-Solomon(14, 10), since number of chunks in an object of LRC is more than Reed-Solomon.

Proactively replicated data blocks reduce number of durable degraded and available degraded slices in the cloud storage systems and hence reduce number of reconstructions. All proactive recovery methods reduce number of data loss events in distributed storage

by reducing number of durable degraded slices count. Figure 3.3 shows that even 90% of disk failure prediction rate does not eliminate degraded slices. Because data degraded due to latent sector errors are not considered in proactive recovery techniques.

3.6.2 Sensitivity Analysis

The proposed recovery techniques are influenced by various important factors such as TIA and failure prediction accuracy. In this section, we examine how disk failure prediction rate affects network traffic and recovery bandwidth with varying TIA.

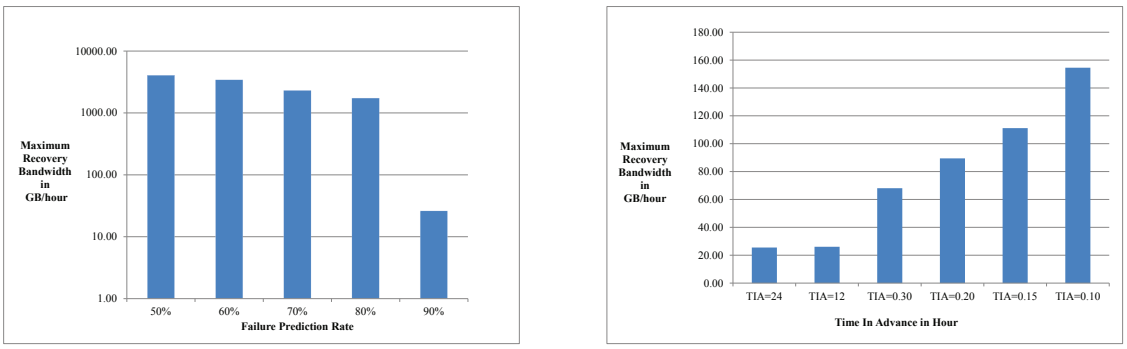
Disk Failure Prediction Rate.

For analyzing how the system is affected by the failure prediction rate, we measured network traffic with varying disk failure prediction rate. Li et al. [113] showed that more than 90% accuracy of disk failure prediction is possible. We run simulation with disk failure prediction accuracy varying from 50% to 90% and calculated recovery network traffic in ProDisk method, as shown in Figure 3.4(a).

The proactive recovery in the storage systems will reduce network traffic (max instantaneous recovery bandwidth in MB/hr) associated with data reconstruction. As expected, network traffic decreases as the failure prediction rate increases. Accurate failure predictions proactively handle failures (transfer one data block instead of 10 data blocks in Reed-Solomon) in storage systems and hence reduce the recovery traffic. Moreover, only in the ProDisk the network traffic varies according to the prediction rate. The rest of the methods are accordance with machine failures. They transfers large amount of data due to proactive recovery compared to ProDisk. Hence it is not showing much variation in network traffic when prediction rates vary.

Time In Advance.

We examine how the failure prediction's TIA affects recovery network traffic of storage systems. Figure 3.4(b) shows how recovery network traffic changes with reduction of TIA of failure prediction in the ProDisk method. Since the maximum recovery bandwidth capacity in these experiments is set to 650 TB/day, reducing TIA from 24 hours to 12 hours does not change average recovery bandwidth drastically. However, reduction in



(a) (b)

Figure 3.4: Maximum instantaneous recovery bandwidth, in GB/hour, calculated over 10 years. (a) for ProDisk with varying failure prediction rates (b) for ProDisk with varying TIA.

TIA below 30 minutes increases network traffic in storage systems. Hence TIA will not affect the recovery bandwidth drastically as it does for network traffic.

Amount of Data Transferred.

To evaluate resource savings due to proactive replication only for hot data, we calculated the total amount of data transferred to the temporary dedicated storage to handle long term temporary machine failure. Amount of data transferred in ProHot/ProHot_LazyCold are directly proportional to the percentage of data determined as hot. Figure 3.5 shows total amount of data transferred in ProMachine. It is approximately twice as ProHot. Recovery methods ProHot and ProHot_LazyCold reduce temporary storage needs.

Figure 3.6 shows average number of replicated slices of popular erasure coding policy Reed-Solomon(14, 10) in a day, with various proactive recovery techniques ProDisk, ProMachine, ProHot and ProHot_LazyCold. Average temporary storage overhead of ProDisk is only 0.001% of total storage. Average temporary storage overhead of ProMachine, ProHot and ProHot_Lazy Cold is 0.089%, 0.089% and 0.05% of total storage, respectively. This shows that proactive recovery techniques provide huge bandwidth savings with very limited additional storage overhead. It is notable that the storage overhead of popular LRC is 14% more than Reed-Solomon (14, 10) and its bandwidth savings are very lim-

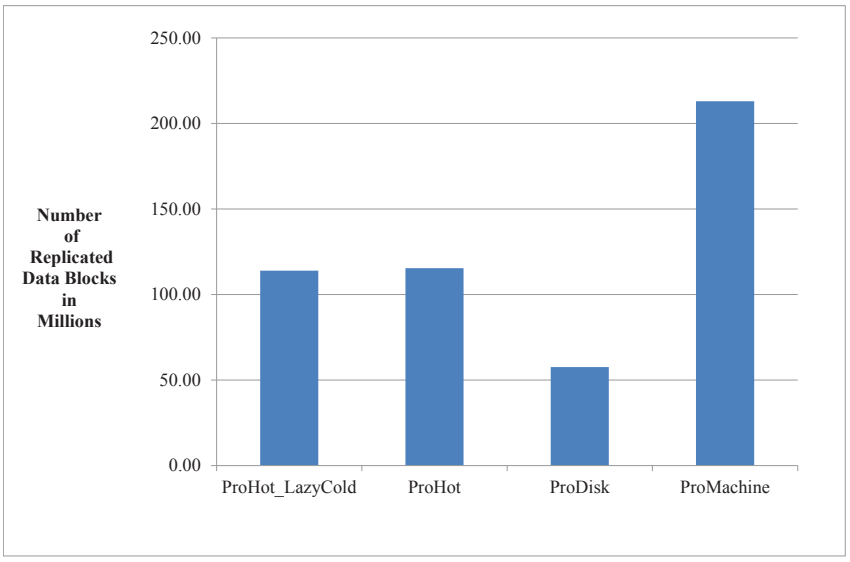


Figure 3.5: Total number of proactively replicated slices due to long term temporary machine failures calculated over 10 years.

ited compared to our proposed storage system. Hence our proposed proactive recovery techniques will offer huge bandwidth savings when it is applied to BigData.

3.7 Summary

The two primary reliability mechanisms employed by cloud storage systems have their own drawbacks. Even though erasure code offers tremendous storage savings compared to replication, reconstructing lost or corrupted data blocks involves large communication overhead. In this chapter, we proposed an approach that utilizes failure prediction techniques to proactively replicate and handle failures in erasure coded storage systems. We defined various recovery techniques with the combination of proactive replication, typical reconstruction of erasure codes, and lazy recovery methods to reduce network bandwidth/traffic in erasure coded cloud storage systems. It uses data blocks hot data status and client SLAs to define an appropriate recovery technique in cloud storage systems. The proposed proactive recovery techniques will improve reliability with exceptional cost savings by improving storage savings and recovery network bandwidth savings. Proposed

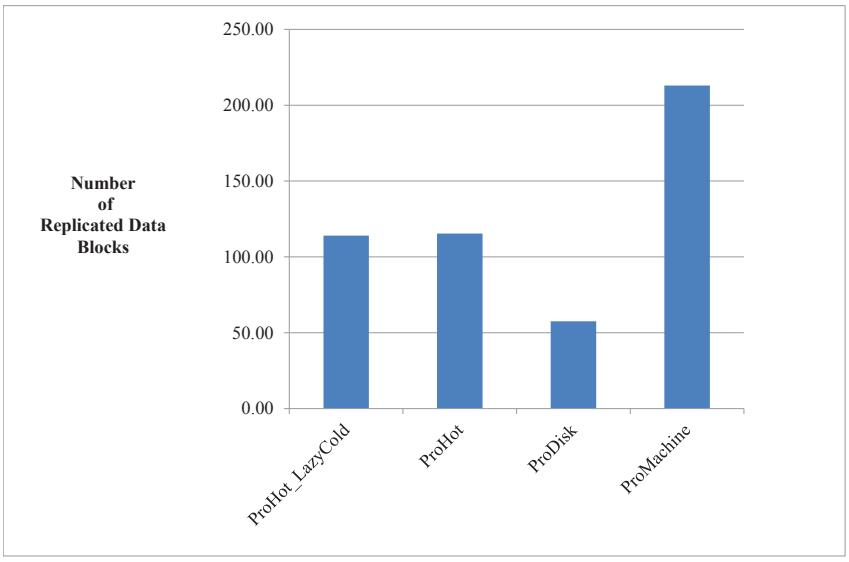


Figure 3.6: Average number of replicated slice in a day for various proactive recovery methods.

proactive recovery techniques also reduce the number of degraded data. Less number of degraded data in erasure coded storage will reduce read latency significantly. Hence the proposed proactive recovery techniques define cost effective solution to improve the reliability Big Data, by reducing the storage overhead significantly while also also supporting the data read in high velocity.

Even though proposed proactive recovery techniques reduce recovery bandwidth substantially, bandwidth savings can be further escalated by optimizing proactive replication. In the next chapter, we will propose an optimization problem using ILP to minimize the number of data blocks replicated during proactive recovery. A novel optimization based proactive recovery is also introduced there. To analyse energy savings of the proposed system and other popular reliability techniques, we have presented the energy estimation module.

Chapter 4

Repair Efficient Erasure-Coded Cloud Storage Systems

Proactive recovery algorithms ProDisk, ProMachine, ProHot and ProHot_LazyCold were proposed in the previous chapter. These algorithms select a set data blocks from failure predicted disk/machine to perform proactive replication. The performance of the proposed methods have been analysed in the previous chapter. The experimental results showed that some of the proposed algorithms can outperform replication, in terms of recovery bandwidth savings. The recovery bandwidth savings of those methods can be further improved by optimizing the selection of data blocks for proactive replication. This chapter presents an optimization technique to eliminate some of the proactive recoveries in the recovery techniques ProDisk, ProMachine, ProHot and ProHot_LazyCold. It utilizes system's current network traffic and data duplication information to optimize the proposed proactive recovery techniques. Applying optimization algorithm on proactive recovery techniques will minimize the number of data blocks to be proactively replicated. Hence optimization reduces temporary storage overhead of proactive replication. A novel recovery technique called Optimized Proactive Recovery (OPR) is also proposed in this chapter. OPR utilizes ILP based optimization to determine appropriate data blocks for

This chapter is derived from: Nachiappan, R., Javadi, B., Calheiros, R. N., & Matawie, K. M. Enhancing Efficiency of Proactive Recovery in Erasure-Coded Cloud Storage Systems. (Submitted to IEEE Transaction on Parallel and Distributed Computing.)

proactive, typical and lazy recovery. We evaluate the optimization algorithm using extensive simulations. Storage systems of cloud play an important role in total energy consumptions of data centres. Erasure coding improves energy savings in terms of storage. However, it can be negated by the energy consumption due to extensive resource consumption during recovery. The proposed proactive recovery methods do not only saves recovery bandwidth but also provides energy savings associated with network bandwidth. However, additional storage space required due to proactive replication will increase energy consumption. To analyse energy savings of various reliability techniques including proposed proactive recovery methods, we propose energy models to measure the energy consumptions of network and storage devices, respectively. A quantitative comparison of energy consumption of replication and several recovery methods are also presented in this chapter.

4.1 Introduction

In the previous chapter, we have proposed a cloud storage system which applies erasure coding to maximize the reliability of data and employs several proactive recovery methods to activate bandwidth efficient repair. The proposed system uses machine and disk failure prediction techniques to predict hardware failures and long-time temporary machine outage. The system proactively handles the failure predicted data blocks. In the event of any disk/machine failure prediction, the client's durability, availability and performance requirements are determined using Service Level Agreement (SLA). When the failures are predicted, system selects an appropriate recovery technique among the proposed proactive recovery techniques ProDisk, ProMachine, ProHot and Prohot_Lazy cold according to the definition of client SLA. According to the selected proactive recovery technique, the system selects a set of data blocks for proactive replication. The experimental results show that the proposed recovery approach improves repair bandwidth efficiency and reduces network traffic in the cloud storage systems. The resource savings of the proposed system vary with respect to the selected proactive recovery technique. To maximize the resource savings, the proposed system must also consider other important system parameters like data duplication and system's current network traffic during proactive recovery.

In previous chapter, experimental results showed that proposed recovery approach improves repair bandwidth efficiency and reduces network traffic in cloud storage systems with limited storage overhead compared to available recovery approaches. The resource savings varies according to the selected proactive recovery technique.

Even though proactive recovery methods ProDisk, ProMachince, ProHot and ProHot_LazyCold significantly reduces repair network bandwidth/traffic in erasure codes, the proposed system simply replicates the failure predicted data blocks according to the selection of a recovery methods and client SLA. It fails to consider important system parameters like data duplication and system's current network traffic during proactive recovery. The temporary storage overhead of proactive replication can be diminished by examining data duplication parameter during proactive recovery. The system's storage and bandwidth efficiency could be further enhanced by eliminating the proactive replication of data blocks that currently have more than one copy. To reduce network throttling, proactive replication should be limited when the system's current instantaneous network bandwidth reaches certain limit.

Improving energy efficiency is another major challenge of cloud data centers. Storage systems consume up to 40% of a data centre's total energy [120]. Read and write latency reduce energy efficiency of storage systems[121]. Energy consumption of cloud storage is influenced by two important factors. They are storage and bandwidth energy consumption [122]. To analyse and compare energy consumption of various recovery techniques, energy consumption of various redundancy techniques is estimated. Energy consumption of those techniques has been estimated in terms of the respective techniques storage overhead and data repair bandwidth. Since intervention of applications running on cloud storage is not considered in this research, energy consumption due to read/write latency is excluded to estimate the energy consumption of cloud storage.

The main contribution of this chapter is as follows,

- An optimization approach is proposed in this chapter to further enhance the efficiency of proactive recovery methods. The optimization aims to minimize the number of data blocks selected for proactive replication. Optimization utilizes system's current

network traffic and data duplication information.

- An optimization algorithm is proposed in this chapter intends to limit proactive replications when the system's instantaneous network bandwidth reaches certain limit. Hence it minimizes bandwidth throttling. It also uses data duplication information to avoid unnecessary replication which may increase temporary storage overhead and recovery bandwidth consumption due to unnecessary replication.

- A novel recovery technique called OPR is proposed to optimize important metrics of storage system such as durability, availability, bandwidth and storage overhead during proactive replication. This method applies ILP based optimization to select appropriate data blocks for proactive and lazy recovery. It intends to apply lazy recovery to minimal data blocks based on the need. Hence it reduces number of degraded data in the system.

- An analysis of energy efficiency of proactive recovery methods is performed. Activating proactive recovery in erasure coding reduces data transfers which can contribute to some energy savings. However, proactive recovery methods suggest additional temporary dedicated storage overhead that may increase system energy consumption. To analyse energy consumption of storage systems, we estimate energy consumption of storage and network devices, respectively.

4.2 Related Work

As Big Data applications demand petabytes of storage, erasure code is becoming an important reliability method in cloud storage systems. Although it improves reliability of Big Data applications with less storage overhead, inefficient data reconstruction issues of erasure code need to be addressed.

Dimakis et al. [123] proposed regeneration codes that reduces network traffic by downloading small amounts of data from higher number nodes than the number of nodes involved in typical reconstruction. However, exact repair of regeneration codes, matching information theoretic bound, remained unresolved. This was followed by several researchers

[124, 125], showing that the exact repair is possible for several other parameters. Another family of codes proposed to reduce repair bandwidth is called LRC [126, 127]. LRC adds local parity such that it reduces number of data blocks accessed during reconstruction. LRC has the side effect of increasing storage overhead by 14% compared to Reed-Solomon [126]. Hitchhiker code [128], built on top of Reed-Solomon code using "piggy-backing" framework, reducing the network traffic by 35% while some encoding time overhead incurred. Even though the above methods reduce repair network bandwidth/traffic, none of them reduced recovery bandwidth as efficient as replication.

Several works in literature suggest system level solutions like delaying data recovery, caching data read during recovery, and proactive replication of data blocks. Silberstein et al. [129] proposed lazy recovery to reduce recovery bandwidth in distributed storage by reducing the recovery rate. This reduces recovery bandwidth up to 76% compared to Reed-Solomon. However, lazy recovery may compromise read performance and data durability. CoARC [53] is a data recovery mechanism which is proposed for handling degraded reads in Hadoop file systems. CoARC activates data recovery not only for the data blocks that requested by clients but also for other unavailable blocks in the same stripe of requested data blocks and caches all recovered data blocks. CoARC reduces network usage in erasure coded Hadoop. This system addressed bandwidth consumption due to degraded read in the HDFS and it fails to address bandwidth consumptions due to data recoveries activated for maintaining reliability requirements.

Li et al. [130] defined a system using failure prediction techniques to implement proactive replication in erasure codes for reducing degraded read latency and improving read performance. However, it did not address machine failures. HP AutoRAID [131] automatically manages migration of data between 2-way replication of active data and RAID 5 for inactive data with the help of access pattern change. The 2-way replications and RAID5 offer limited reliability. Araujo et al. [132] proposed hybrid coding and double coding. Both strategies combine the use of replication and erasure coding. Even though it saves bandwidth upon reconstruction, it reduces storage efficiency. Li et al. [77] defined a cost effective data reliability management mechanism to ensure reliability of massive data with minimum replication using generalized data reliability model. However,

storage savings due to minimum replication can compromise reliability. None of the above works incorporate client's expectation and nature of data to define bandwidth and storage efficient recovery of erasure codes.

Greenan et al. [133] estimates energy consumption of data recovery in erasure coding. It is estimated in terms of power consumption of disks involved in recovery operation. Several researches [133, 134] calculate data reconstruction energy consumption, in terms of participating nodes energy consumption and its active time during data recovery.

The requirements of data durability, availability and read performance of data may vary with respect to client SLA. Client can also refine the requirements with respect to the access frequency of data. Clients requirement are defined in SLA. In previous chapter, we have proposed adaptive cloud storage system that uses failure predictions and access patterns to define several proactive recovery methods. The system employs various recovery methods. The system performs dynamic selection of recovery methods as per the client SLA and applies proactive recovery for a selected set of data blocks from failure predicted disk/machine depending according to the recovery method selected. System applies typical reconstruction to recover data blocks that are not handled proactively for techniques ProDisk, ProMachine and ProHot, whereas it applies lazy recovery for ProHot_LazyCold.

Failure predictions in cloud storage systems enable cloud service providers to apply efficient proactive failure management in cloud storage. Various statistical and machine learning methods are used to predict failures in cloud storage systems. A few methods [135, 136] are used to predict hard drive failures based on SMART attributes. Li et al. [135] had achieved 95% predictions with False Alarm rate less than 0.1%. Many researches focused on predicting failures in distributed systems using system logs. Javadi et al. [137] presented a failure model as a predictive method to measure distributed systems availability and unavailability. Agarwal et al. [138] uses log messages to predict failures in Hadoop clusters. Data access pattern in a distributed storage can be used to identify the popularity of data blocks in real-time over a certain period of time. Based on their popularity, data blocks can be classified as hot, warm or cold. As the access pattern changes, popularity of data blocks have to be updated. Various researches [139, 106, 140]

used popularity-based classification to improve durability, availability, and read performance of cloud storage systems. In this chapter, we propose an optimization algorithm to further enhance the efficiency of proposed proactive recovery methods. A novel optimized proactive recovery technique is also proposed.

4.3 Adaptive Bandwidth Efficient Cloud Storage Systems

An adaptive cloud storage system employs several proactive recovery methods. In the event of any failure prediction, it selects one of the proactive recovery methods which can meet client SLA efficiently. To improve the efficiency of proactive recovery, the proposed system adapts to client SLA and chooses the most suitable method for recovery. Client data can be classified as hot, warm or cold depending on the access frequency. Data recovery can be delayed for cold data that is having less access frequency. The client may also accept a delay in cold data. At the same time, access latency is not acceptable for hot data. Activating proactive recovery of hot data can reduce access latency in the presence of failure. In this chapter, we include an algorithm in the existing adaptive bandwidth efficient cloud storage systems such that it minimizes the number of data blocks replicated during proactive recovery, regardless of the selection of any proactive recovery methods. This optimization algorithm was included in the existing proposed system. It will to further increase the storage and network efficiency.

4.3.1 Architecture and Design

Architecture of the adaptive bandwidth efficient cloud storage system is proposed in last chapter. We have introduced new components called Enhanced Proactive Recovery (EPR) and OPR in the existing architecture, to further enhance the efficiency of proposed system, regardless of the selection of any proactive recovery method using failure prediction. An overview of the system architecture is depicted in Figure 4.1. It is implemented as an extension of regular object storage. Object storage manages data as objects where each object has both data and metadata. A dedicated proxy server supports encoding and

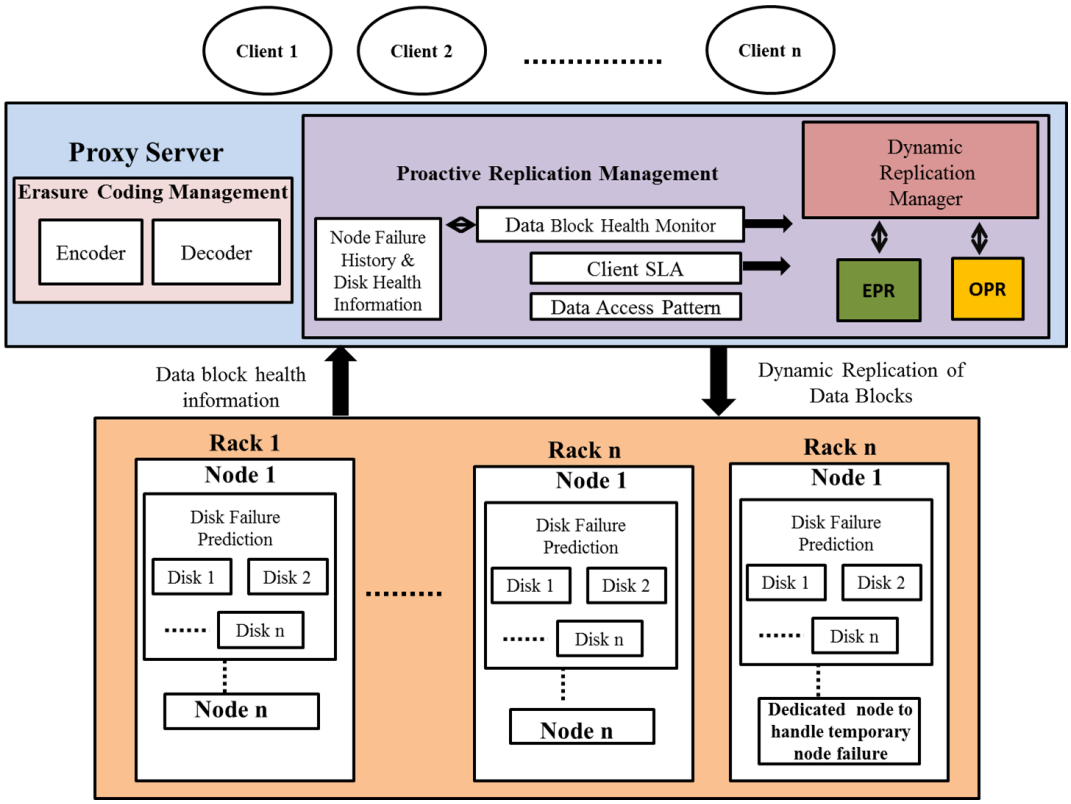


Figure 4.1: Architecture and design of the proposed recovery techniques [141].

decoding of erasure codes. It also handles failures in storage systems. The object server stores and retrieves data as objects. Object server's availability status and disks health status are reported to the proxy server, which is responsible for increasing or decreasing the data object's replication factor. The system adjusts the replication factor of erasure coded objects when failures are predicted.

The component *disk failure prediction* monitors the health status of individual disks, using classification and regression tree methods with information derived from SMART attributes [135]. *Node failure history and disk health information* component collects node failure history and calculates node's MTTF and MTTR using various statistics of availability and unavailability. It also collects disk failure alarms from the component *disk failure prediction*. Data access pattern classifies data blocks as hot based on its popularity over a period of time. Assuming that data blocks with high access frequency have more chance to be accessed in the future, we define those as hot. It is recorded as $H = \{h_{ij}\}$ where h_{ij} is the jth block from disk i that is identified as hot. *Data block health monitor*

collects all information about failure predicted nodes and disks from *node failure history and disk health information module*. It identifies and sets different flags of the data blocks that are predicted for failure due to disk, machine failures. Client's requirements in regard with durability, availability and access latency are recorded in the client SLA. *Dynamic replication manager* chooses one of the best recovery techniques which can meet client SLA with limited resources from recovery methods ProDisk, ProMachine, ProHot and ProHot_LazyCold that are proposed in previous chapter. If the client requires high durability, ordinary availability and access latency, dynamic replication manager will select the recovery technique ProDisk. ProHot will be selected if they require high availability and low access latency of hot data. In case client requests high durability, availability and low latency, the technique ProMachine will be selected. The ProHot_LazyCold will be selected by dynamic replication manager if the client requests high availability and low access latency for hot data and they don't bother about the availability and access latency of cold data. Based on the selection of the recovery technique, it chooses a set of data blocks for proactive recovery.

Enhanced Proactive Recovery module attempts to reduce the number of data blocks elected for proactive replication. EPR selects optimal subset of data, by taking into account the system's current network traffic and data duplication. It also deletes the corresponding replicated data blocks once the failure predicted machine has come back to life or a failure predicted disk does not fail as expected. It is also responsible for scaling up and down the number of dedicated temporary storage nodes, according to the failure predictions and the amount of data to be stored in temporary storage during a period of time. It is also responsible for allocating a highly available node as a temporary storage such that any failure in dedicated temporary storage node is minimal. Any failure prediction of this temporary storage will also lead to proactive replication. Defining an optimization algorithm for EPR is the main contribution of this chapter and it is extensively discussed in the next section.

Optimized Proactive Recovery is designed to support the novel proactive recovery technique OPR. OPR is defined using fine grained optimization to define lazy, typical or proactive recovery for the data blocks from failure predicted machines/disks.

4.4 Enhanced Proactive Recovery

This section explains the optimization of proactive recovery techniques. The problem formulation of applying optimization in proactive recovery techniques is presented here.

4.4.1 Proposed Recovery Approach

The overall functionality of the proposed enhanced adaptive bandwidth efficient cloud storage system and its recovery approach is discussed in this section.

The proposed system initially stores data with any (n, k) erasure code. Utilizing disk/machine failure prediction methods, failure types and MTTR of any node in proposed storage system are predicted. Failures are also identified as disk, permanent machine, temporary long term machine (MTTR>15 minutes), or temporary short term machine (MTTR<15 minutes) failures. A set of data blocks $(b_1, b_2, ..., b_i)$ that is more likely to be accessed soon is defined as a hot data set H. Based on the failure types, and client SLA, one of the appropriate recovery techniques is selected form the recovery techniques ProDisk, ProMachine, ProHot, ProHot_LazyCold.

When the disk/permanent machine failures are predicted, all the data blocks in the failure predicted disk (all data blocks of each disk in a failure predicted machine) are selected for proactive replication by the *dynamic replication manager*. Next, EPR applies minimization on selected set of data blocks by considering system's current network traffic and data duplication. It chooses a subset of data blocks for proactive replication. The selected subset of data are proactively replicated into the permanent storage as described in [130]. The counter variables of corresponding replicated data blocks are incremented. These counter variables are used to identify if the particular data blocks are replicated already. They are also used to delete the data blocks that are replicated due to false predictions. A delay is applied while deleting data blocks that are replicated due to false prediction. The TIA of a failure prediction is used as a time delay to delete the data blocks that are replicated due to false positives. Time delay larger than TIA is the better choice despite this will result in extra storage. The choice of time delay varies and depends on the storage system.

In the event of long term temporary machine failure predictions, *dynamic replication manager* selects all data blocks from failure predicted machine in ProMachine. It selects a certain set of data blocks in recovery methods ProHot and ProHot_LazyCold. EPR selects a subset of it. The subset of data, selected by EPR is replicated into the dedicated temporary storage nodes. While data blocks are proactively replicated into temporary storage, the corresponding data blocks counter variables are incremented. These variables are used to identify if the particular data blocks are already replicated. When the failure predicted nodes recover from actual failure, proactively replicated copy of the data blocks in the corresponding nodes are deleted. In the occurrence of node/disk failures, an appropriate reference is made to proactively replicated block. Hence typical reconstruction of erasure codes is replaced with proactive replication.

4.4.2 Problem Formulation

Let $B = \{b_{ij}\}$ be a set of data blocks stored in the cloud storage system. Let b_{ij} denotes a data block which is stored in the j-th location of disk i. Let DR=$\{dr_{ij}\}$ be a set of counter variables that represents replication count of the corresponding data block b_{ij} from set B. The counter variables in DR are used to represent the data blocks that are replicated due to the occurrence of disk/permanent machine failures. Similarly, MR= $\{mr_{ij}\}$ is a set of counter variables used to represent the data blocks that are replicated due to temporary machine failures. The variable mr_{ij} is incremented if the corresponding data block b_{ij} is replicated on the nodes that are dedicated to handle machine failure. The cardinality of sets DR and MR are equal to the cardinality of set B. The value of mr_{ij} represents the number of copies of block b_{ij} available in the storage node dedicated to handle temporary failures. The variable dr_{ij} is decremented on the actual failure of *disk i*. The variables mr_{ij} are decremented when the machine that holds *disk i* have come back from failure, after the occurrence of actual failure of the same machine. The copy of a data block b_{ij} is also deleted from the temporary dedicated storage node. In case of false positive, the copy of a data block is deleted after applying appropriate time delay.

Upon predicting any disk/permanent machine failures or temporary long term machine failures, a set of data blocks is selected by *dynamic replication manager* for proactive

replication according to the selection of proactive recovery methods. Let $FP = \{b_{ij}\}$ is a set of data blocks selected for proactive replication. EPR has to select an appropriate subset of data for proactive replication from the set FP, by taking into account data duplication and system's current network traffic information. Let $X = \{x_{ij}\}$ be a set of binary decision variable and each variable x_{ij} represents block b_{ij} from FP; $x_{ij} = 1$ if the data block b_{ij} from FP is selected for proactive replication and $x_{ij} = 0$ otherwise.

The module EPR should not select the data block b_{ij} for proactive replication if the system has more than one copy of the data block b_{ij} at the failure prediction time t. This will avoid unnecessary data duplication of the block b_{ij}. For example, consider a disk "disk i" is predicted for failure at time t. The *dynamic replication manager* selects all blocks in "disk i" for replication. Let b_{ij} be a data block belong to "disk i" which is selected for replication by *dynamic replication manager* at time t. The system may already hold a copy of b_{ij} due to the failure prediction of machine which contains the "disk i". In this case, EPR will not select the data block b_{ij} for proactive replication. This will not only save bandwidth of creating extra copy but also storage. On occurrence of actual failure of "disk i", appropriate reference could be made to the copy of data block b_{ij} such that it can handle the failure of "disk i". The scenario is similar when the block b_{ij} is marked for temporary machine failure. Hence we have,

$$x_{ij} = 0 \qquad \forall\ bij \in D \quad if \quad mr_{ij} = 1 \tag{4.1}$$

$$x_{ij} = 0 \qquad \forall\ bij \in M \quad if \quad dr_{ij} = 1 \tag{4.2}$$

Systems network traffic can be effectively managed by eliminating proactive recovery of some failure predictions based on system's Current Recovery Bandwidth (CRB). When the system's current recovery bandwidth reaches system's recovery bandwidth capacity, EPR should avoid proactive recovery of certain failure predictions that may increase the system's recovery bandwidth above certain Bandwidth Limit (BL). BL should be carefully defined such that it does not affect the average recovery bandwidth of the system, and this is formulated in equation 3 below.

$$(S * \sum xij)/TIA + CRB \leq BL \qquad \forall xij \in X \tag{4.3}$$

We formulate the problem of selecting a subset of data blocks for proactive replication, from set FP as a binary integer linear programming as follows,

$$Minimize \sum xij \qquad \forall xij \in X \tag{4.4}$$

Subject to:

$$x_{ij} = 0 \qquad \forall\ bij \in D \quad if \quad mr_{ij} = 1 \tag{4.5}$$

$$x_{ij} = 0 \qquad \forall\ bij \in M \quad if \quad dr_{ij} = 1 \tag{4.6}$$

$$x_{ij} = 1 \qquad \forall\ bij \in D \quad if \quad dr_{ij} = 0 \tag{4.7}$$

$$x_{ij} = 1 \qquad \forall\ bij \in M \quad if \quad mr_{ij} = 0 \tag{4.8}$$

$$(S * \sum xij)/TIA + CRB \leq BL \qquad \forall xij \in X \tag{4.9}$$

$$xij = \{0, 1\} \qquad \forall xij \in X \tag{4.10}$$

The Bandwidth Limit (BL) in constraint 9 is determined according to the system's Recovery Bandwidth Capacity (RBC) at the time of prediction (when proactive recovery starts). In order to determine BL, some important parameters of the system should be analysed. Let S be the size of data blocks, d_n be the average number of data blocks distributed in a disk, d_{max} be the maximum number of data blocks distributed in a disk and m_d be the number of disk in a machine. An average Projected Bandwidth Need (PBN) of any proactive recovery of a machine failure prediction can be calculated as follows,

$$PBN = (S * d_n * m_d)/TIA \tag{4.11}$$

Similarly, the average PBN needed for any disk failure prediction can be calculated as follows,

$$PBN = (S * d_n)/TIA \tag{4.12}$$

The PBN calculation is determined according to the selection of proactive recovery methods. For ProMachine method, PBN is calculated as in equation 11. It is calculated as in equation 12 for the recovery methods ProDisk, ProHot and ProHot_LazyCold. Upon any failure prediction, PBN will be calculated and compared with the system's Recovery Bandwidth Capacity (RBC). If sum of PBN and system's Current Recovery Bandwidth

(CRB) is less than RBC, the system should allow any proactive recovery. On the other hand, if sum of PBN and system's current recovery bandwidth is greater than RBC, or PBN exceeds system's current recovery bandwidth, the system should simultaneously allow the proactive recovery of a single machine and a disk at time t. Hence, BL of constraint 9 is calculated as follows

$$BL = S * d_n * m_d + S * d_{max} \quad if \quad CRB < PBN \tag{4.13}$$

On the other hand, when system's PBN is smaller than RBC, the system has to increase the BL such that it can handle proactive recovery of any single machine or disk failure along with system's CRB. It avoids proactive recovery, if the system is busy with handling any other failures. When CRB exceeds PBN, the BL of constraint 9 is calculated as follows,

$$BL = CRB + S * d_n * m_d + S * d_{max} \quad if \quad RBC < PBN \tag{4.14}$$

In order to maintain the systems network traffic to the level of system's RBC, we can set BL as $BL = RBC$. However, doing this will completely eliminate proactive recovery when system's CRB reaches RBC. As a result, typical reconstruction will be conducted to recover data blocks that were not proactively handled. This may increase system's network traffic substantially.

4.4.3 Enhanced Proactive Recovery Algorithm

To solve the problem of minimizing the number of proactive replicated data blocks, an algorithm called Enhance Proactive Recovery Algorithm (EPRA) is defined. Upon any failure predictions, the proposed algorithm determines the set of data blocks that are needed to be handled proactively by taking in to account system's current network traffic and data duplication information. The EPRA is presented in Algorithm 2.

On receipt of any failure prediction event, the algorithm examines how the system's network traffic will be affected while activating proactive recovery. The required calculations are programmed in Algorithm 3. This algorithm calculates the total Transfers Required (TR) to proactively handle the predicted event. Using TR, the total Projected

Algorithm 2 Enhanced Proactive Recovery Algorithm

INPUT: FP, FT

```
1: if BANDWIDTH_CONSTRAINT (FP)=true then
2:     for each b_ij in FP do
3:         if m_ij ≤ 1 and d_ij ≤ 1 then
4:             x_ij = 1
5:         else if Ft = machine and d_ij ≥ 1 then
6:             x_ij = 0
7:         else if Ft = disk and m_ij ≥ 1 then
8:             m_ij = m_ij − 1
9:             d_ij = d_ij + 1
10:            define disk holding copy of d_ij as permanent
11:            x_ij = 0
12:        end if
13:    end for
14: else
15:    for each b_ij in FP do
16:        x_ij = 0
17:    end for
18: end if
```

OUTPUT: X

Bandwidth Need (PBN) to proactively handle the predicted event is calculated as follows,

$$PBN = TR/TIA \tag{4.15}$$

The algorithm also calculates Projected Network Traffic (PNT) of the system using system's CRB as follows,

$$PNT = CRB + PBN \tag{4.16}$$

Based on the calculated PNT, the system determines whether to proactively handle the predicted failure or not. Following that, data de-duplication is performed in lines 2 to 9.

Algorithm 3 Bandwidth_Constraint(FP)

```
1: procedure BANDWIDTH_CONSTRAINT(FP)
2:     initialize TR=0
3:     initialize PBN=0
4:     for each b_ij in FP do
5:         if m_ij ≤ 1 and d_ij ≤ 1 then
6:             TR = TR + S
7:         end if
8:     end for
9:     PBN = TR/TIA
10:    PNT = CRB + PBN
11:    if PNT ≤ BL then
12:        return true
13:    else
14:        return false
15:    end if
16: end procedure
```

For any failure prediction of disk i, let us consider a data block b_{ij} in *disk i* is already replicated due to the proactive recovery of machine that contains *disk i*. The system should avoid replicating the block b_{ij} due to the prediction of *disk i*. However, an appropriate reference has to be made to the copy of b_{ij} such that it cannot be deleted during the eviction process, which is activated when the machine containing *disk i* recovers from failure. On the other hand, consider a scenario where a data block b_{ij} has to be proactively handled on receipt of a machine failure prediction to which it belongs. If the system already has a copy of b_{ij}, due to failure prediction of *disk i* then it will avoid replicating block b_{ij}. The algorithm gets a set of data blocks in failure predicted machine/disk and the failure type from *dynamic replication manager*. It sets decision variable $x_{ij} = 1$ if the corresponding data block b_{ij} from the failure predicted set has to be replicated. EPRA sends X to *dynamic replication manager*. The *dynamic replication manager* replicates

the data block b_{ij} from set FP provided the corresponding decision variable x_{ij} is 1.

4.5 Optimized Proactive Recovery (OPR)

The novel optimization technique called OPR is discussed in detail in this section. This technique attempts to address the network spike issue of ProMachine when TIA of failure prediction is low and access latency of ProHot_LazyCold due to increased number of degraded data in this technique. OPR applies ILP based optimization to optimize the selection of data blocks for proactive, typical or lazy recovery while taking into account of system's available network traffic. All disk failures are handled as defined in ProDisk in OPR. Upon any machine failure prediction, this technique attempts to maximize the proactive recovery and applies lazy recovery only when it is required.

4.5.1 Problem Formulation

Let $B = \{b_{ij}\}$ be a set of data blocks that are stored in the cloud storage system. Let b_{ij} denote a data block that is stored in the j-th location of disk i. Let $DR = \{dr_{ij}\}$ be a set of variables that keep track of replication factor of corresponding data block b_{ij} in storage cluster. The cardinality of the set DR is equal to the cardinality of set B. The value of dr_{ij} represents, number of copies of block b_{ij} exist in storage cluster. On the occurrence of actual failure of disk$i$, the reference will be made to the replicated copy to act as original copy. The corresponding dr_{ij} is also decremented when the replicated copies reduces. Also, variables dr_{ij} are decremented when the copy of the data block b_{ij} is deleted once the machine that holds disk i have come back from failure after the occurrence of actual failure of same machine. In case of false positive, the copy of a data block is deleted after applying appropriate time delay and dr_{ij} variables are updated accordingly. To regulate the network traffic hike due to proactive replication and to minimize read latency due to lazy recovery, OPR selects optimal data blocks for proactive or lazy recovery from failure predicted machine.

When Dynamic Replication Manager selects OPR as a recovery technique, optimal selection of data blocks for proactive, typical or lazy recovery are performed as follows. When predicted failure is disk, it is handled as in ProDisk regardless of the selected recov-

ery technique. That is regardless of the system's current available network bandwidth; it selects all data blocks from failure predicted disk for proactive replication. Data blocks in unpredicted disk failures are recovered by typical reconstruction of erasure codes. However, when predicted failure is machine, it applies ILP based optimization to determine data blocks that requires proactive or lazy recovery as follows.

Let $FP = \{b_{ij}\}$ is a set of data blocks in failure predicted machine. Let HFP is a subset of FP that holds hot data blocks in failure predicted machine FP. Let $X = \{x_{ij}\}$ be a set of binary decision variable, such that each variable x_{ij} represents block b_{ij} from FP, $x_{ij} = 1$ if the data block b_{ij} from FP is selected for proactive replication. OPR attempts to maximize the proactive recovery of data blocks from set FP by utilizing system's current available recovery bandwidth to the most in the moment of failure predictions. Since proactive replication of hot data is highly important to reduce unavailability of hot data to the minimum, OPR ensures to define proactive replication for hot data that is all data blocks from subset HFP. Also, it selects enough cold data blocks for proactive replication when they can be replicated within system's current available bandwidth. The proactive replication of hot data blocks from failure predicted machine is ensured as follows:

$$\sum x_{ij} = h \quad \forall\, b_{ij} \in HFP \quad where\, h = |HFP| \tag{4.17}$$

Let S be the size of data blocks. System's network traffic hike due to proactive replication can be regulated by defining lazy recovery for appropriate data blocks by considering system's Current Recovery Bandwidth (CRB) and Recovery Bandwidth Capacity (RBC). RBC of storage cluster is determined when the storage cluster is defined. CRB is the amount of recovery bandwidth currently in use in storage cluster. Using this information, the system can avoid proactive replication of certain cold data blocks from failure predicted machine as follows:

$$(S.\sum x_{ij})/TIA + CRB <= RBC \qquad \forall x_{ij} \in X \tag{4.18}$$

OPR will also avoid unnecessary data duplication of the block b_{ij}. For example, consider a machine "machine j" is predicted for failure at time t. Let b_{ij} be a data block belong to "machine j" at failure prediction time t. The system may already hold a copy

of b_{ij} due to the failure prediction of disk "disk i" that contained the "machine j". In this case, OPR will not select the data block b_{ij} for proactive replication. This will not only save bandwidth of creating extra copy but also save storage. Let set Replicated Copy (RC) contains set of all data blocks that have more than one copy. In this case, OPR should not select the data block b_{ij} form the set FP for proactive replication if the system has more than one copy of the data block b_{ij} at the failure prediction time t. Hence, we have,

$$\sum x_{ij} = 0 \quad \forall\ b_{ij} \in RC \tag{4.19}$$

OPR defines lazy recovery for data blocks that are not selected for proactive replication from set FP. For any unpredicted disk failures, it applies typical reconstruction of erasure codes whereas it applies lazy recovery for cold data blocks from unpredicted machine failures.

We formulate the problem of selecting a subset of data blocks for proactive replication, from the set of data blocks with a predicted failure FP as an ILP as follows,

$$Maximize \sum x_{ij} \qquad \forall x_{ij} \in X \tag{4.20}$$

Subject to:

$$\sum x_{ij} = |HFP| \quad \forall\ b_{ij} \in HFP \quad where\, h = |HFP| \tag{4.21}$$

$$(S. \sum x_{ij})/TIA + CRB <= RBC \qquad \forall x_{ij} \in X \tag{4.22}$$

$$\sum x_{ij} = 0 \quad \forall\ b_{ij} \in RC \tag{4.23}$$

$$where \quad x_{ij} = \{0, 1\} \qquad \forall x_{ij} \in X \tag{4.24}$$

After this optimization, for all b_{ij} from FP are selected for proactive replication if corresponding x_{ij} is 1. When x_{ij} is 0 for any b_{ij}, lazy recovery will be activated for corresponding b_{ij} when meta data shows replication factor of b_{ij} is 1. Hence when there is enough recovery bandwidth OPR will replicated all data blocks in failure predicted machine as in ProMachine. While ensuring proactive replication for hot data blocks, it will apply lazy recovery for appropriate data blocks when CRB is high. Therefore, OPR selects dynamic set of data blocks for proactive, typical and lazy recovery for each failure

predictions. OPR will handle all disk failures as defined ProDisk except it avoids proactive replication of data blocks that have more than one copy.

Using this optimization, system's network traffic can be maintained to the level of system's RBC. However, when the failure prediction accuracy are very low, high number data blocks from failed disks must be recovered using typical reconstruction of erasure codes which may increase system's network traffic substantially. Even when failure prediction accuracy is very low, OPR will provide significant bandwidth savings by intelligently activating lazy recovery.

4.5.2 Optimized Proactive Recovery Algorithm

To optimize the selection of data blocks for proactive, typical and lazy recovery, we present Algorithm 1. OPR optimization problem attempts to maximize proactive replication with in system's available network bandwidth. When data blocks that require proactive replication are identified, appropriate recovery is activated as defined in *Optimized Proactive Recovery Algorithm*. Algorithm 1 represents how it selects recovery types for data blocks in failure predicted machines and disks. This algorithm accepts following as input. The set of data blocks from failure predicted device (FP), Failure Type (FT) such as disk/ machine failures, failure prediction TIA, data block size (S), system's Current Recovery Bandwidth (CRB), Recovery Bandwidth Capacity (RBC), set of hot data blocks in corresponding failure predicted device (HFP) and set of data blocks in corresponding failure predicted device that have more than one copy RC. Upon each machine failure prediction, ILP based optimization will be defined as presented in lines 2-9 of the Algorithm 1. Once optimization defines appropriate values for each x_{ij} in X, type of recovery for corresponding blocks b_{ij} is defined. After optimization, if x_{ij} holds value 1, proactive recovery is defined for corresponding data block b_{ij} in FP. Lazy recovery is opted for data blocks b_{ij} if it's corresponding x_{ij} is evaluated to 0 and the corresponding data block b_{ij} does not belong to the set RC. When the predicted failures are disk, OPR will apply proactive replication for all b_{ij} in FP except for all b_{ij}'s that belong to RC. When a b_{ij} in FP belongs to RC, appropriate reference should be made. Otherwise, it may get deleted when a machine whose prediction made a copy, has come back to life. The complexity of

this algorithm is O(N), where N is the cardinality of the set FP.

4.6 Energy Consumption Analysis

Several metrics are used to measure energy consumption of storage systems. Some metrics use energy consumption of hardware/software components to calculate the energy consumption of the storage systems. Some others measure the energy consumption of storage systems by measuring application's usage of physical resources like storage, network and memory. In order to compare energy efficiency form replication, erasure coding to various proactive recovery techniques, we estimate energy consumption of storage and network devices. Energy consumption of the storage systems are estimated in terms of power consumptions of disks. We calculate the energy consumption of the network devices in terms of the amount of data transferred via top of rack switch during recovery.

Since we use these energy models to compare the energy consumption of various recovery methods, we do not consider the energy consumption of the intervening applications running on top of the storage system.

4.6.1 Energy consumption of storage devices

As mentioned above, we will estimate the storage energy by calculating energy consumption of the disk drives in the storage system. Even though several other devices like machine, rack and cooling systems are involved in energy consumption of storage systems, disk remains the most important component of the storage systems, and the energy consumption of the storage system is directly proportional to the amount of disks employed in storage system. Since we use this model for the system comparison for various recovery methods, we will ignore the energy consumption of other devices. Hence, energy consumption of the storage devices is expressed as follows,

$$E_t = D_t * T_t * U \tag{4.25}$$

Where E_t is the energy consumption of storage system during the time period t as the product of number of disks active D_t, the amount of time active T_t and the energy consumption of the disk per unit time U.

Number of disks used in replication is relatively high compared to erasure coding since it stores more number of data blocks to ensure reliability. Proactive replication in erasure coding uses additional disks for a certain period of time. Disk failure/permanent machine failure predictions use the additional disk from the time of prediction till the actual occurrence predicted disk failure. Temporary machine failure uses the additional storage from the time of prediction until the recovery of corresponding machine. Number of disks used in case of disk failure/permanent machine failure predictions is equal to number of disks predicted for failure, whereas it varies with respect to the selection of recovery methods in the event of temporary machine failure predictions.

4.6.2 Energy consumption of network devices

We estimate data recovery energy consumption by calculating the amount of data transferred through the router during recovery of each failure event. We have calculated energy consumption of data recovery using the model proposed by Viswanath et al. [17].

$$E_t = E_p * R_t/S + E_{st} * R_t \tag{4.26}$$

We have calculated energy consumption of router as a sum of energy consumption of processing and storing the data blocks that were involved in recovery during time t. Energy consumption caused by data processing in routers is expressed as the product of per packet processing energy E_p and the incoming data rate during recovery time t. Input data rate can be calculated using data transfer rate (amount of data transferred per second), R_t and data block size S as R_t/S. Data storage energy in the router is calculated as a product of per byte storage energy E_{st} and the input data rate R_t. Total energy consumption of router during time T is calculated as the sum of all individual recovery energy consumptions E_t during the time T. Since we use equation 18 to measure energy consumption of routers during recovery, router's idle power is not considered. Energy consumption of data transfer due to the intervention of applications running on top the storage is not considered.

Finally, the total energy consumption of the system, during time T is calculated as the sum of storage and network devices energy consumption during that time period.

4.7 Performance Evaluation

We have used ds-sim [4] simulator to evaluate the efficiency of proactive recovery methods after applying optimization. Significant amount of codes have been added in ds-sim to implement failure prediction, proactive recovery, optimization of proactive recovery and energy consumption.

The ds-sim simulates 3-tier storage components including disks, machines and racks. The ds-sim stores data in blocks and multiple data blocks form a stripe. A stripe is composed of a set of original and parity data blocks such that any data block in a stripe can be reconstructed using a subset of data blocks in a stripe. In n way replication, a stripe consists of all replicas of a block, whereas it is comprised of k original and $n - k$ parity blocks for Reed-Solomon(n, k) erasure code. The ds-sim randomly chooses n racks to store n blocks of a stripe such that no two blocks in stripe are placed on nodes in the same rack.

The ds-sim generates failure and recovery events for all hardware components using either synthetic probability distributions or failure traces. Each storage component disk, machine and rack is incorporated with separate failure and recovery distribution. Disk failures include both latent sector failure and permanent disk failures. Latent sector failures are detected and recovered using a technique called *scrubbing*. Machine failures include both transient and permanent failures. The ds-sim starts the recovery of any permanent machine failure immediately; whereas it applies 15 mins delay for initiating the recovery of any transient failures. Rack failures are assumed to be transient. The ds-sim performs a runtime simulation and records all instantaneous properties of the system, including repair bandwidth, repair energy, repair storage overhead for proactive recovery and the number of degraded stripes. Table 4.1 lists the values of simulation parameters. The choices of energy consumption of disk and router's storage are made from [142] and [143], respectively.

In order to carefully compare and evaluate the efficiency due to optimization, we have used the same set of failure events for heuristics and optimization. This eliminates any difference in metrics due to the variation of failure events generated by the simulator.

Table 4.1: Simulation Parameters

Parameter	Value
Total data	3 petabyte
Duration	10 years
Disk Capacity	750 GB
Recovery bandwidth capacity	650 TB/day
Disks/machine	20
Machines/rack	11
Disk Energy Consumption while operating	43.2 kilojoule/hour
Router's per byte storage energy	14 nJ
Packet processing energy	1375 nJ
Prediction percentage	90
Hot data threshold	40% of total data
Number of iterations	50

4.7.1 Performance Analysis

In this section, we analyse and compare the energy consumption of replication, most popular erasure code Reed-Solomon(14, 10) and various proactive recovery techniques. The trade-off between dedicated temporary storage overhead and bandwidth savings of various proactive recovery methods have also been discussed in this section.

System Energy consumption

The simulations are conducted with the configuration parameters and failure prediction rate, as listed in Table 4.1. We use TIA of 24 hours, which is found reasonable in [113] and [115]. As the failures are generated by the simulator, the recovery energy consumption and energy consumption of dedicated temporary storage are calculated for each failure event,

except for machine failures that stay less than 15 minutes. Since we do not consider the intervention of any application running on top of the storage, we have calculated the storage energy of disks by assuming that disks are active during the entire period of simulation. The shutting down of the inactive disks is not considered in simulation. They usually enhance energy savings of the storage systems. In case of any disk failure prediction, additional disks are activated to support proactive recovery. Those incur some additional energy consumption. It is calculated by assuming that the storage system activates an extra disk at the time of failure prediction and it is active from the time of failure prediction till the actual occurrence of failure. In the event of temporary machine failure prediction, the system will activate number of disks proportional to the number of data blocks that are selected for proactive recovery. Those disks will be active from the time of failure prediction till recovery of the failure predicted machine and energy consumption is calculated accordingly.

Figure 4.2 shows the comparison of average energy consumption in KJ/day for replication, Reed-Solomon(14, 10) and proactive recovery methods. The figure also shows average energy consumption of the system, with individual split ups for energy consumption of storage, recovery bandwidth and temporary dedicated storage for proactive recovery, respectively. Reed-Solomon(14, 10) saves system's overall energy consumption up to 51.7% compared to replication. ProDisk reduces energy consumption up to 51.8% compared to Replication. ProHot reduces energy consumption up to 51.9%, ProMachine reduces energy consumption by 51.8% and ProHot_LazyCold reduces energy consumption by 52% compared to the same approach. The energy savings of proactive recovery methods are very limited. This is due to the fact that energy consumption of dedicated storage compensates the energy savings of recovery bandwidth. Figure 4.3 shows data reconstruction and storage energy for various coding scheme and proposed recovery methods. They are normalized against replication. Energy consumption of temporary storage overhead of proactive recovery methods are calculated for TIA 12 hours. Figure 4.3 shows that storage energy consumption overhead of proactive recovery methods are the least. The power consumption can be further reduced by carefully scheduling the proactive replication with appropriate TIA, which is one of the promising future research directions.

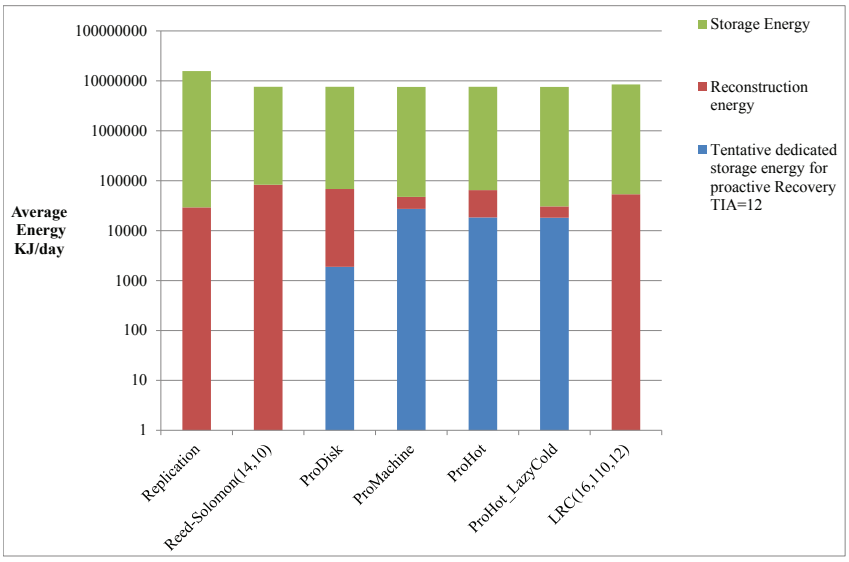


Figure 4.2: Storage system's average energy consumption in KJ per day.

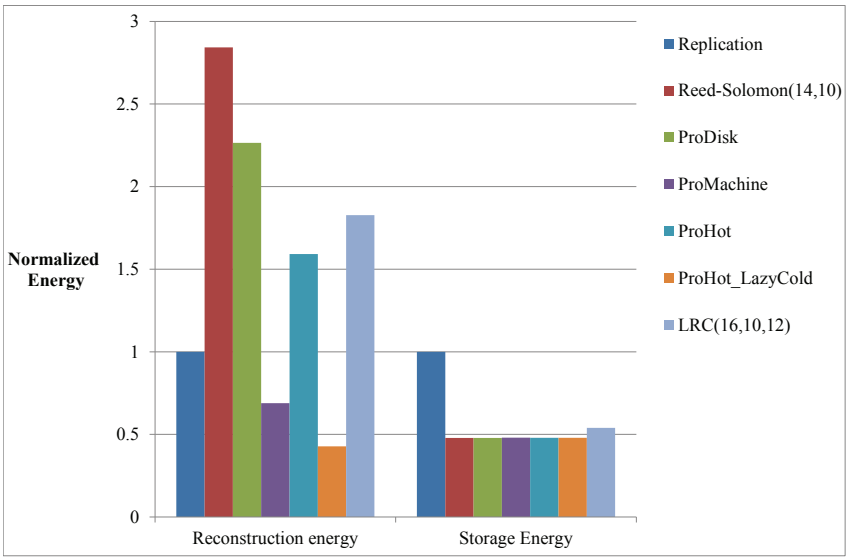


Figure 4.3: Average storage and recovery energy consumption in KJ per day.

Temporary dedicated storage overhead

To evaluate resource savings from proactive replication, the average number of data blocks replicated per day is calculated. Figure 4.4, shows the trade-off between recovery band-

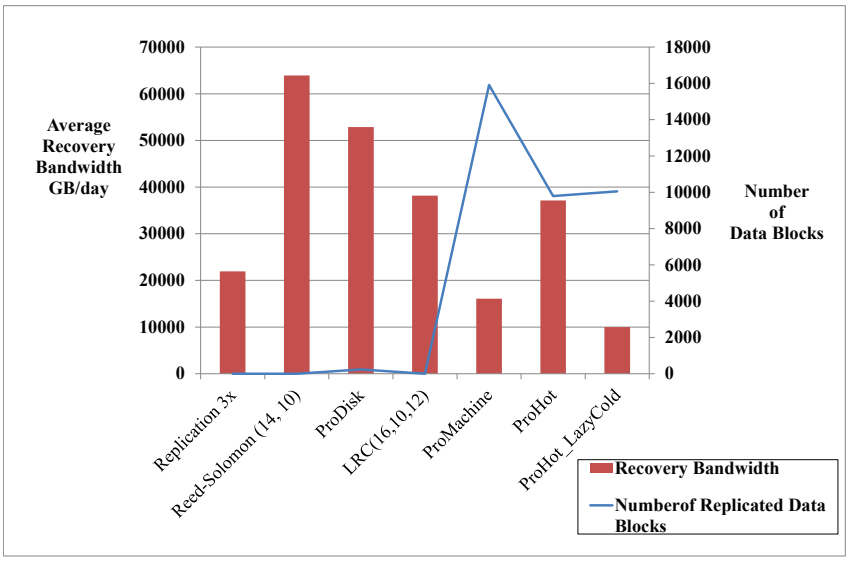


Figure 4.4: Storage overhead and average recovery bandwidth.

width and temporary dedicated storage for various proactive recovery methods. ProMachine reduces recovery bandwidth of Reed-Solomon(14, 10) up to 75% with approximately 1.3% storage overhead compared to Reed-Solomon. ProHot reduces recovery bandwidth up to 41% where as ProHot_ LazyCold reduces recovery bandwidth by 85% with 0.75% additional storage savings compared to Reed-Solomon. The storage overhead of replication, ProDisk, ProMachine, ProHot and ProHot_LazyCold are 90%, 0.01%, 1.3%, 0.75% and 0.75%, respectively compared to Reed-solomon. Figure 4.4 shows that proactive recovery methods offer excellent bandwidth savings as a compensation of dedicated temporary storage overhead due to proactive replication.

4.7.2 Enhanced Proactive Recovery

In this section, we investigate how the optimization of proactive recovery further improves the efficiency of storage systems. To eliminate any differences in measurement, due to variations of events generated by the simulator, we apply optimization on the same set of events that we used for evaluating heuristic proactive recovery techniques. We compare various measures such as repair network traffic/bandwidth, energy and storage overhead

of heuristic proactive recovery techniques against optimized proactive recovery techniques.

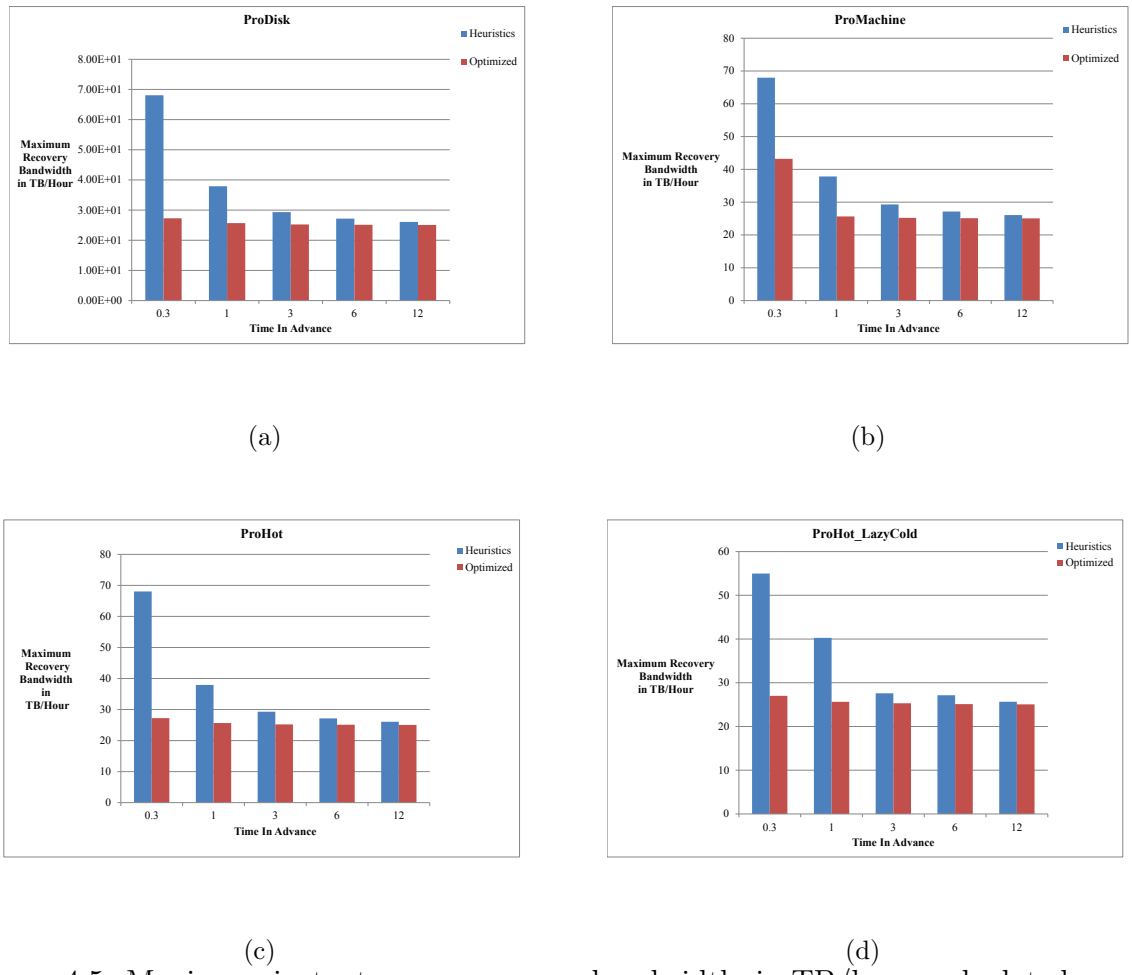
Repair network traffic

We examine how the optimization of proactive recovery reduces repair network traffic compared to its respective heuristics. The results of examining network traffic of proactive recovery methods, with varying TIA of failure predictions are presented in Figure 4.5. Since recovery network bandwidth is inversely proportional to recovery time, reduction of TIA increases network traffic of storage systems. In this experiment, maximum recovery bandwidth capacity is set to 650 TB/day. When TIA is set to 12 hours, optimization does not show much savings. When TIA is set to 30 minutes, optimization shows significant savings in network traffic.

The optimization of recovery methods ProDisk, ProHot and ProHot_LazyCold reduces network traffic up to 60%, 37%, 60% and 49%, respectively, compared to its corresponding heuristic methods. Similarly, optimized ProDisk, ProMachine, ProHot and ProHot_LazyCold recovery methods reduces network traffic around 4%, 4%, 4% and 3.6%, respectively, when TIA is 12 hours. This shows that the network savings due to proactive recovery increases, as TIA decreases. Applying optimization on proactive recovery methods reduces recovery bandwidth to the level of systems recovery bandwidth capacity. However, ProMachine's network traffic reduction due to optimization is very limited, while TIA is 30 mins. This is due to the fact that this method handles large amount of data compared to other methods, during proactive recovery. If we attempt to further reduce the recovery traffic of ProMachine, by adjusting the parameter BL in constraint 3, it will end up in less proactive recovery of data due to machine failures. As a result, it will further increase average network bandwidth and the traffic of the system because it should apply typical reconstruction for the data blocks that are not proactively handled.

Temporary dedicated storage overhead

To evaluate temporary storage savings due to the optimization of proactive replication, we have calculated the total amount of data transferred during the simulation period. Figure 4.6 shows the comparison of total amount of data transferred for both heuristics and optimization, respectively for proposed proactive recovery methods. ProDisk provides up



(a) (b)

(c) (d)

Figure 4.5: Maximum instantaneous recovery bandwidth, in TB/hour, calculated over 10 years. (a) ProDisk (b) ProMachine (c)ProHot and (d)ProHot_LazyCold.

to 46% storage savings due to optimization. Similarly, optimizing ProMachine, ProHot and ProHot_LazyCold offer up to 10%, 12% and 12% of storage savings, respectively compared to its respective heuristics. Storage savings of proactive recovery methods are increased by applying optimization. The storage savings due to optimization of each method is different because BL in constraint 9 varies with respect to the recovery methods.

Repair Bandwidth

To examine, recovery bandwidth savings due to optimization of different proactive recovery methods, we have estimated average recovery bandwidth in a day. Figure 4.7 shows

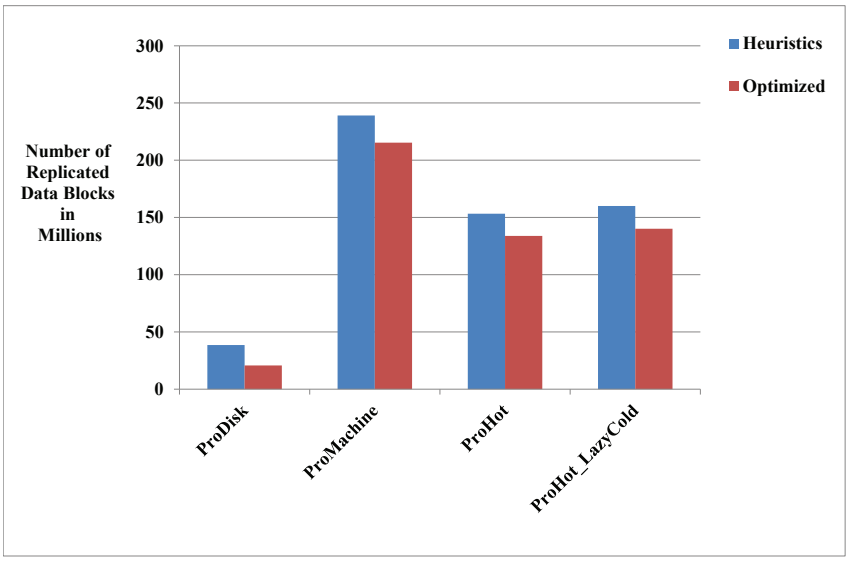


Figure 4.6: Total number of proactively replicated slices due to proactive recovery calculated over 10 years.

the bandwidth savings due to optimization of various recovery methods ProDisk, ProMachine, ProHot and ProHot_LazyCold. Optimizing the proactive recovery of methods such as ProDisk, ProMachine, ProHot and ProHot_LazyCold can offer up to 3%, 9.5%, 4% and 12% of savings, respectively. Bandwidth savings of proactive recovery methods due to optimization are increased since it saves bandwidth by avoiding data duplication. Also, constraint 9 increases recovery rate of the storage systems, which will further save recovery bandwidth.

Energy Consumption

To examine energy savings due to optimization of various proactive recovery methods, we have estimated the average recovery bandwidth in a day. Figure 4.8, shows the energy savings due to optimization compared to respective heuristics recovery methods. Since optimization eliminates data duplication, it saves energy in terms of temporary storage overhead and recovery bandwidth. Applying optimization on ProDisk, ProMachine, ProHot and ProHot_LazyCold offered up to 3%, 5%, 3.5% and 4% of bandwidth savings compared to its respective heuristics.

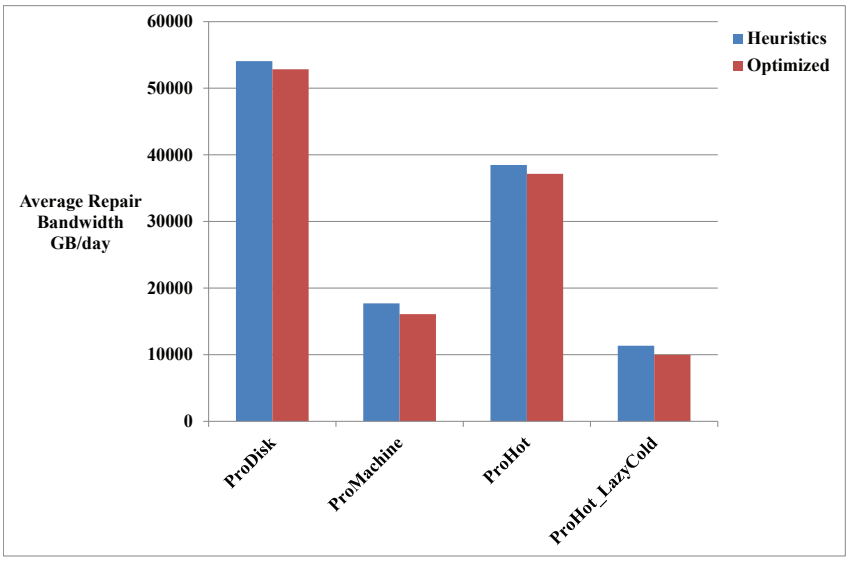


Figure 4.7: Average recovery bandwidth in GB per day.

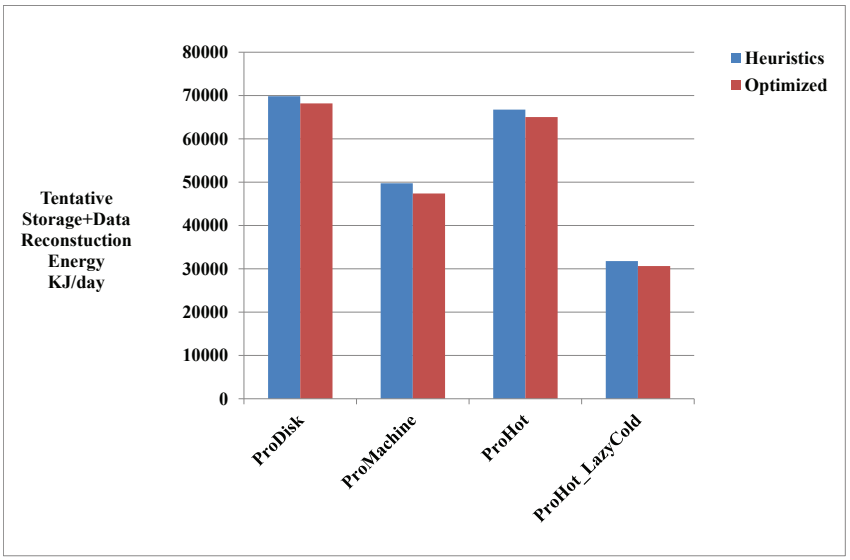


Figure 4.8: Average energy consumption in KJ per day.

Reliability

To examine how the optimization of proactive recovery affects reliability of the storage system, the average durable degraded and available degraded slices in a day is estimated.

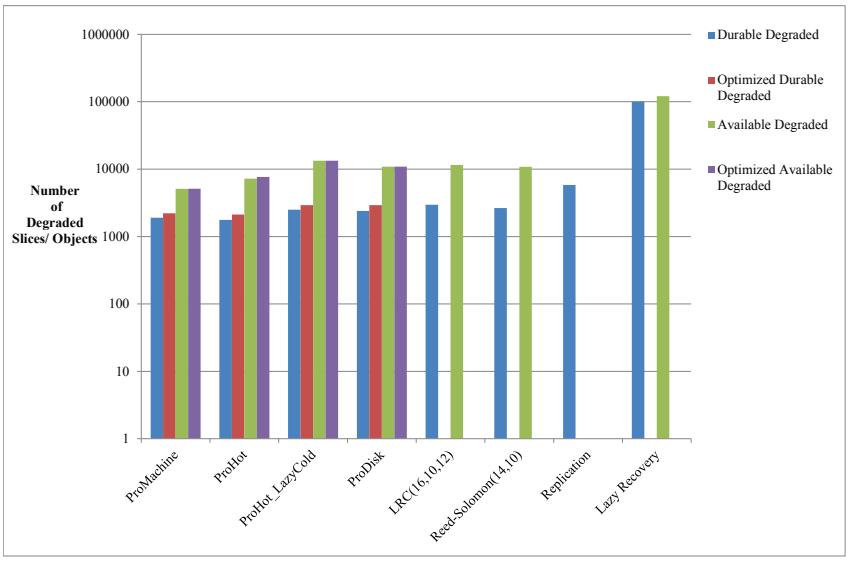


Figure 4.9: Average number of durable degraded and available degraded slices per day.

Figure 4.9 shows number of degraded slices for various reliability techniques. It also shows the impact of optimization on reliability compared to its respective heuristic proactive recovery methods. Since optimization attempts to minimize recovery network traffic, it slightly increases number of degraded slices compared to heuristics. However, numbers of degraded slices in optimized proactive recovery techniques are still much better than the native methods.

Sensitivity Analysis

To determine how EPRA is influenced by failure prediction rate, we have measured parameters such network traffic/bandwidth, storage overhead, and energy consumption, with varying failure prediction rate.

For analyzing how the system is affected by the failure prediction rate, we have measured network traffic with varying disk failure prediction rate. Li et al. [135] showed that more than 90% accuracy of disk failure prediction is possible. We run simulation with failure prediction accuracy varying from 60% to 90%. We have calculated the recovery network bandwidth and traffic of ProMachine method with TIA equals to 12 hours. They are depicted in Figure. 4.10 and Figure 4.11. Figure 4.10 depicts bandwidth savings at

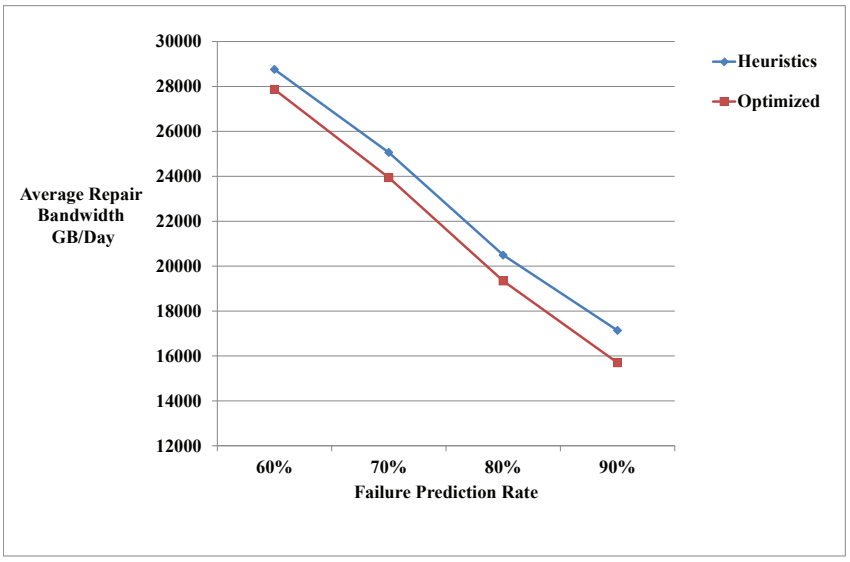


Figure 4.10: Average repair bandwidth, in GB/day with varying prediction rate.

failure predictions rate of 90%, 80%, 70% and 60% of ProMachine method with TIA 12 hours. They are 75%, 68%, 61% and 56% compared to Reed-Solomon(14, 10). Optimizing proactive recovery in the storage systems provides bandwidth savings up to 8%, 6%, 4% and 3% compared to its heuristics for prediction percentage 90%, 80%, 70% and 60%, respectively. Bandwidth savings due to optimization reduces as the failure prediction rate decreases. Even though optimization offers excellent bandwidth savings with prediction percentage of 90%, savings in terms of network traffic is becoming very limited as the failure prediction rate decreases. When the prediction rate is low, the system's network traffic increases as it activates typical reconstruction of unpredicted failures. Optimizing proactive recovery in the storage systems reduces network traffic (max instantaneous recovery bandwidth in MB/h) by eliminating proactive recovery for some failure predictions. However, with less failure prediction, this has minimum effect.

Hence optimization of proactive recovery provides significant improvements on repair network bandwidth, energy consumption and temporary dedicated storage even with smaller failure prediction percentage. However, network traffic savings due to optimizations are limited when failure prediction percentage is minimal.

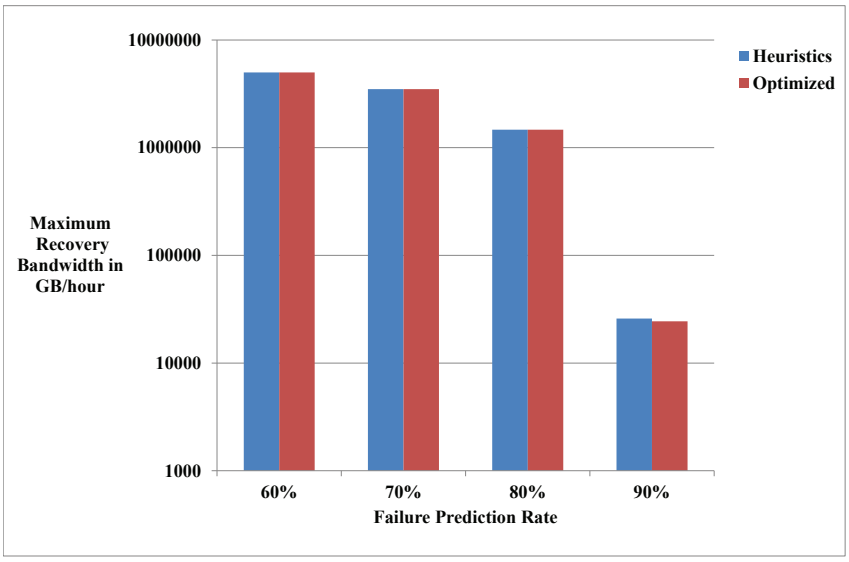


Figure 4.11: Maximum instantaneous recovery bandwidth, in GB/h, calculated over 10 years with varying prediction percentage.

4.7.3 Optimized Proactive Recovery

Recovery Network

In this section, we have analysed the reduction of recovery network traffic in erasure coded storage due to proactive recovery. To analyse the effectiveness of proposed proactive recovery methods, we have evaluated storage overhead and average recovery bandwidth of various storage systems such as replication, erasure codes and several proposed proactive recovery methods. Since proposed proactive recovery techniques are defined using failure predictions, we have also analysed the impact of recovery network traffic when TIA of failure predictions reduces.

Recovery Network Bandwidth and Storage Overhead

To evaluate additional storage overhead due to proactive replication, we have calculated average number of data blocks replicated per day. Figure 4.12, shows the trade-off between recovery bandwidth and storage overhead of various storage systems. ProMachine reduces recovery bandwidth of Reed-Solomon (14, 10) up to 75% with approximately 1.3% storage

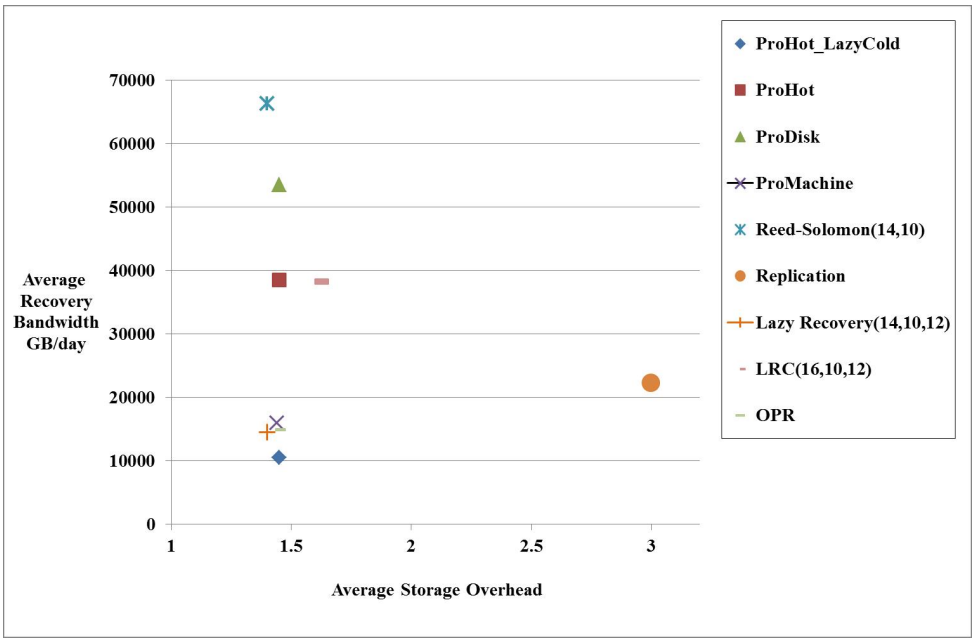


Figure 4.12: Storage overhead VS average recovery bandwidth in GB/day.

overhead compared to Reed-Solomon when TIA of failure prediction is 12 hours. ProHot reduces recovery bandwidth up to 41% where as ProHot_ LazyCold reduces recovery bandwidth by 85% with 0.75% additional storage savings compared to Reed-Solomon. The storage overhead of replication, ProDisk, ProMachine, ProHot, ProHot_LazyCold and OPR are 90%, 0.01%, 1.3%, 0.75% and 0.75%, 1.09% respectively compared to Reed-Solomon.

Figure 4.12 shows that proactive recovery methods offer excellent bandwidth savings as compensation of dedicated temporary storage overhead due to proactive replication. Our novel proactive recovery method OPR saves recovery bandwidth better than ProMachine but not as good as ProHot_LazyCold when TIA is 12. However, this will vary when TIA and failure prediction accuracy reduces. OPR tries to handle more amounts of data blocks pro-actively compared to ProHot_LazyCold to reduce amount of slices degraded due to lazy recovery which is the reason for increased network bandwidth. In turn, OPR will improve system's read performance. Even though OPR tries to handle more data blocks pro-actively like ProMachine, it always offers better bandwidth savings than ProMachine as it defines lazy recovery for necessary data blocks and also avoids data duplication.

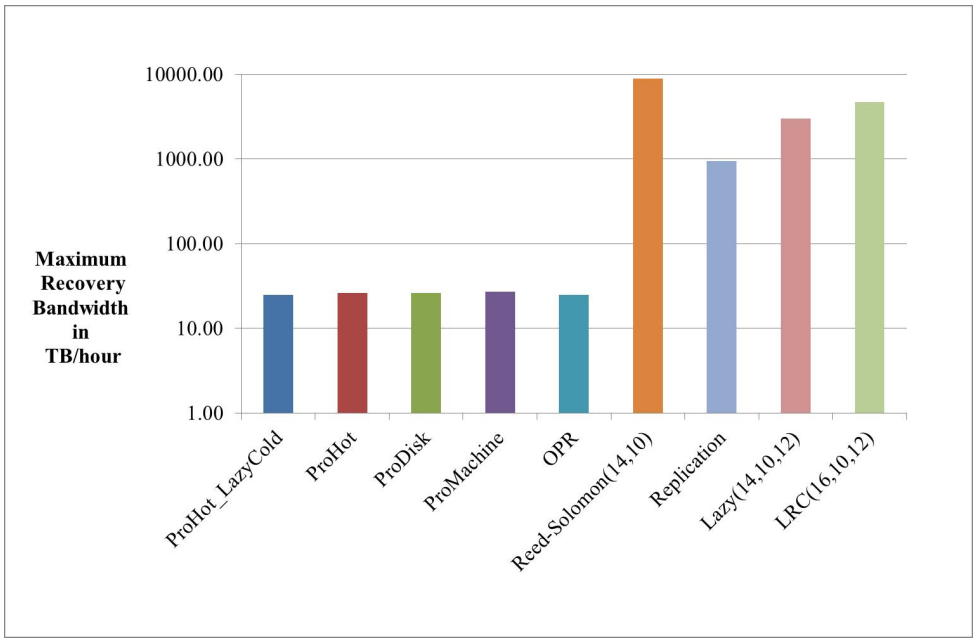


Figure 4.13: Maximum instantaneous recovery bandwidth, in TB/day, calculated over 10 years.

Repair network traffic

To estimate repair network traffic of storage systems, we have estimated maximum instantaneous network bandwidth for the simulation time period of 10 years. Figure 4.13 shows recovery network traffic of various storage systems. It shows, all proposed proactive recovery techniques reduces network traffic significantly compared to other existing reliability techniques replication, Reed-Solomon(14, 10) and LRC.

Figure 4.14 shows network traffic of proactive recovery methods with varying TIA of failure predictions. Results showed that recovery network traffic increases as TIA reduces for proactive recovery methods ProDisk, ProMachine, ProHot and ProHot_LazyCold since they attempt to transfer static amount of data blocks with in the time period of TIA. However, our novel technique OPR does not increase network traffic significantly when TIA reduces. OPR optimally selects data blocks for proactive replication by considering system's current network bandwidth usage. By optimizing the selection, it defines perfect balance between proactive and lazy recovery and hence it maintains recovery network traffic to the minimum.

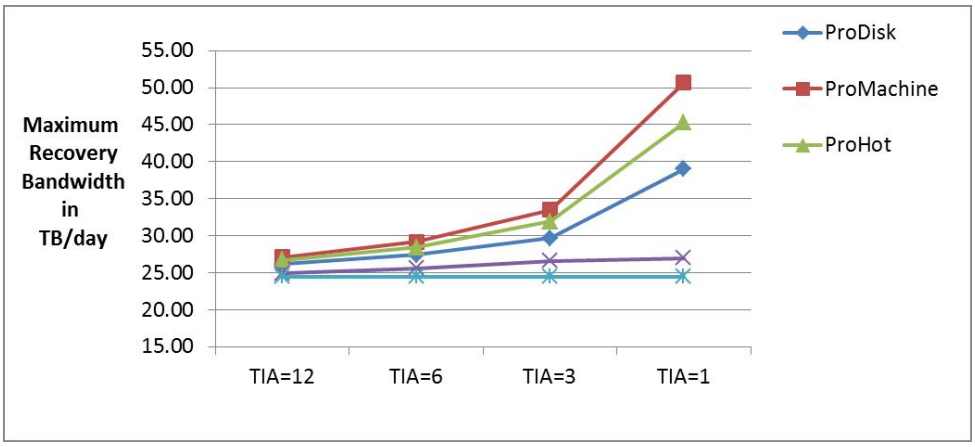


Figure 4.14: Maximum instantaneous recovery bandwidth, in TB/day, calculated over 10 years with varying TIA.

4.7.4 System Energy Consumption

Figure 4.15 shows the comparison of average energy consumption in KJ/ day for various storage schemes such as replication, (14,10) Reed-Solomon, LRC and (14,10) Reed-Solomon with different proposed proactive recovery methods. This figure depicts storage system's average energy consumptions break down in storage, recovery bandwidth and associated temporary dedicated storage for proactive recovery. Reed-Solomon (14, 10) and LRC (16, 10, 12) saves system's overall energy consumption up to 51.7% and 45% respectively compared to replication. Proposed proactive recovery techniques reduce network traffic/ bandwidth by dedicating additional temporary storage overhead. Hence energy savings of proactive recovery techniques due to reduced recovery bandwidth is compromised by the energy consumption of additional temporary storage overhead of those methods. Energy savings of ProDisk, ProMachine, ProHot, ProHot_LazyCold and OPR are approximately up to 51.8%, 51.8%, 51.9%, 52% and 52.4% respectively compared to replication. Energy savings of proactive recovery are estimated with TIA 12 hours while failure prediction accuracy is of 90% in Reed-Solomon (14, 10). Due to the trade-off between recovery bandwidth and temporary storage overhead energy consumptions in various proactive recovery schemes, energy savings of proposed proactive recovery are very limited compared to Reed-Solomon (14, 10) with traditional reconstruction. Among the proposed proactive recovery techniques, OPR offers best energy savings. OPR improves

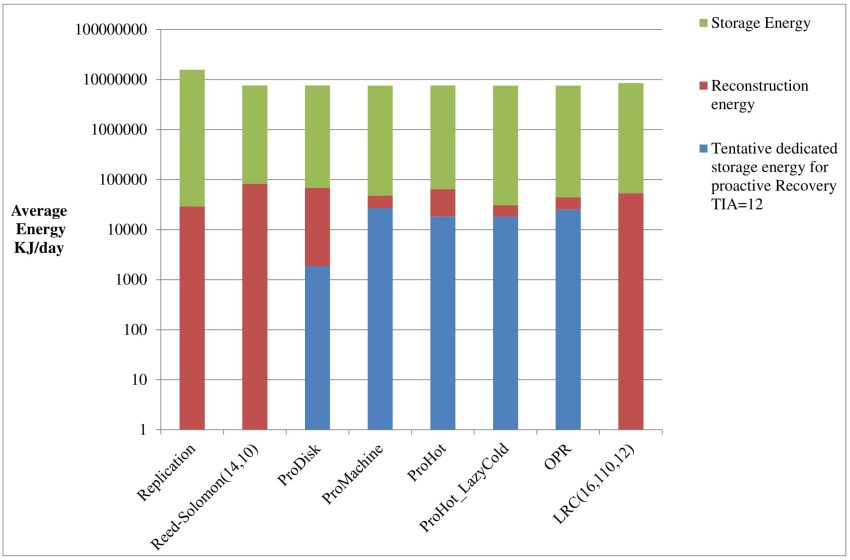


Figure 4.15: Storage system's average energy consumption in KJ per day.

energy savings by reducing temporary storage overhead due to duplication of replicated blocks and also by improving recovery bandwidth savings with lazy recovery.

Figure 4.16 shows data reconstruction and storage energy for various coding scheme and recovery methods that are normalized against replication. As in Table 1, in our simulation, we have used substantially high per byte storage energy compared to per byte processing energy of the router. This measures may vary with storage systems and routers used in cloud storage cluster. Also, Figure 4.16 shows that storage energy consumption overhead of various proactive recovery methods when TIA is 12 hours. Energy consumption of proactive recovery techniques can be varied with respect to TIA, disks per byte storage consumption and router's per byte processing energy consumption. Hence energy consumption of storage system with proactive recovery can be maximized by optimizing and scheduling proactive replication with respect to time such that temporary storage and recovery bandwidth energy consumptions are optimized. This is one of the possible future directions of this research.

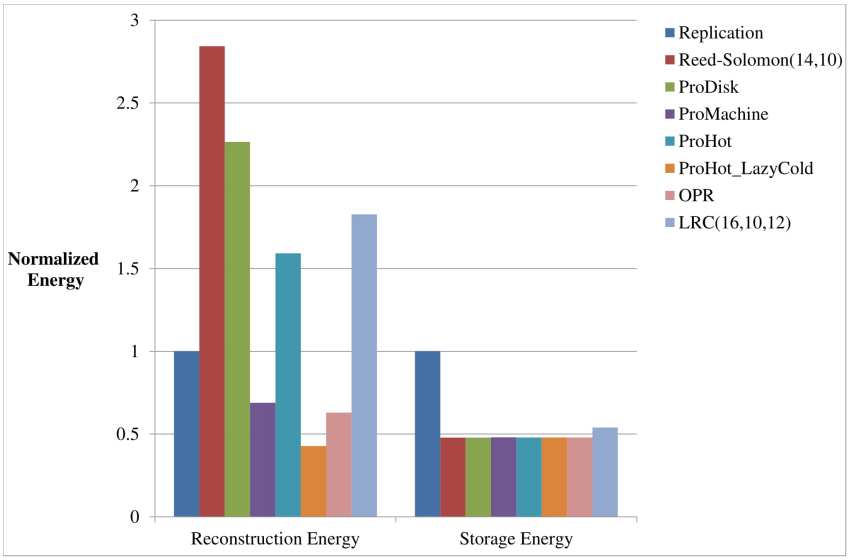


Figure 4.16: Average storage and recovery energy consumption in KJ per day.

Sensitivity Analysis

We run simulation with failure prediction accuracy varying from 50% to 90% and calculated recovery network bandwidth of all proposed proactive recovery methods when TIA of failure predictions is 12 hours. The results are depicted in Figure. 4.17. From Figure 4.17 bandwidth savings at failure predictions rate of 90%, 80%, 70% , 60% and 50% of ProMachine method with TIA 12 hours are 75%, 68%, 61% and 56% compared to (14,10) Reed-Solomon. Bandwidth savings due to proactive replication reduces as the failure prediction rate decreases for the methods ProDisk, ProMachine, ProHot and ProHot_LazyCold whereas OPR shows increased bandwidth savings when failure predictions decreases. OPR defines proactive recovery for more data blocks when failure prediction is high and actives lazy recovery for large data blocks when failure prediction accuracy is low according to the optimization problem defined in section 4. This will maintain network traffic to minimum. Low failure prediction accuracy will activate to low proactive recovery and high typical reconstruction in Reed-Solomon (14,0). Hence when failure prediction accuracy is low, network traffic also got increased. However, our experimental result showed that OPR maintained network traffic almost to the level system's allocated

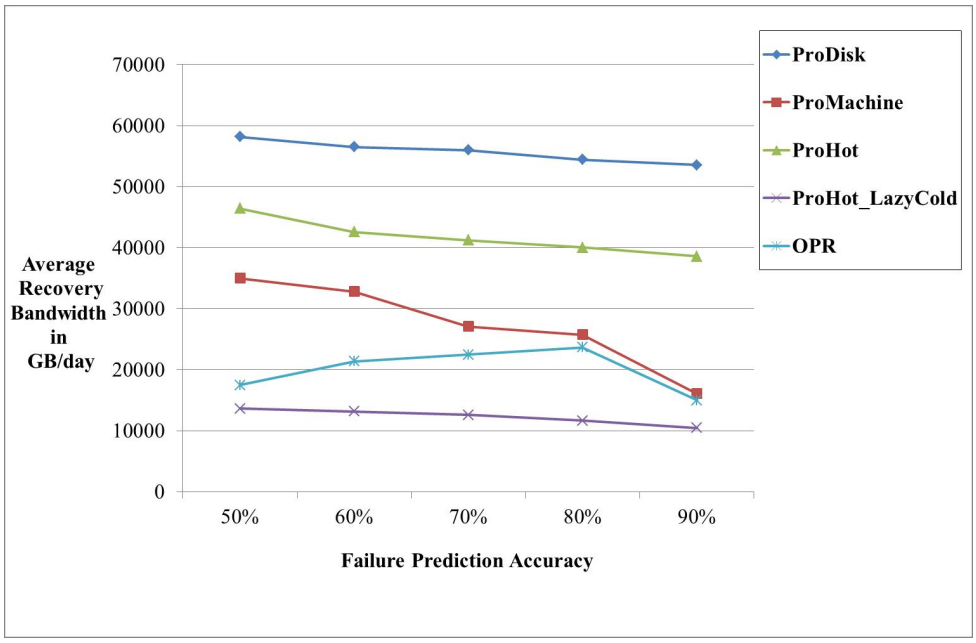


Figure 4.17: Average recovery bandwidth, in GB/day with varying prediction percentage.

network capacity. However, it also increased number of degraded slices.

Regardless of the variations in failure prediction accuracy, there is no doubt that all the proposed proactive recovery techniques reduce network bandwidth and traffic significantly compared to the typical reconstructions of erasure code. Proactive recovery also increases system's reliability with limited temporary storage overhead. Even though proactive recovery methods offers limited energy savings compared to typical reconstructions, it can be maximized using appropriate scheduling algorithms. Scheduling algorithms can be defined to minimize the usage duration of temporary storage overhead.

4.8 Summary

The two primary reliability mechanisms—replication and erasure coding—employed in cloud storage systems have their own drawbacks. Even though erasure code offers tremendous storage savings compared to replication, reconstructing lost or corrupted data blocks incur large communication overhead.

In the previous chapter, to achieve maximum recovery bandwidth savings in erasure codes, we have proposed, several failure-prediction-based, novel proactive recovery techniques. They are defined with the combinations of replications, erasure coding and lazy

recovery. As an extension of this, we propose an optimization approach and an algorithm, in this chapter. Optimization attempts to minimize the number of data blocks to be replicated during proactive recovery. This optimization contributes to increase resource savings of the storage systems. A novel proactive recovery called OPR is proposed. We have also analysed the energy consumptions of replication, erasure coding and erasure coding with several recovery approaches. Experiments showed that the applying optimization on proactive recovery techniques have further increased the resource savings in cloud storage. It is also notable that proactive recovery methods energy savings are almost close to native erasure codes. The storage and bandwidth savings due to optimization helps to improve reliability of Big Data with additional cost savings while also supporting the data read in high velocity.

The proposed proactive recovery methods reduce number of degraded read in the storage system. Hence it can reduce degraded read performance of the system significantly. However, they required to change references in physical storage and metadata. Applying such changes to an existing storage system will lead to unnecessary chaos. To reduce degraded read performance in existing erasure coded storage, we propose a novel cache based solution in the next chapter. They do not suggest defining any such references.

Algorithm 4 Optimized Proactive Recovery Algorithm

INPUT: FP, FT, S, TIA, CRB, RBC, HFP, RC

```
1: if FT="Machine" then
2:     for each b_ij in FP do
3:         Generate decision variables set X= x_ij
4:     end for
5:     Set objective as Maximize ∑ x_ij ∀x_ij ∈ X
6:     Add constraint as ∑ x_ij = |HFP| ∀ b_ij ∈ HFP where h = |HFP|
7:     Add constraint as (S * ∑ x_ij)/TIA + CRB <= RBC ∀x_ij ∈ X
8:     Add constraint as ∑ x_ij = 0 ∀ b_ij ∈ RC
9:     Optimize ILP problem for evaluating x to 0 or 1
10:    for each b_ij in FP do
11:        if x_ij == 1 then
12:            Define proactive recovery for block b_ij
13:        else
14:            if b_ij ∉ RC then
15:                Define lazy recovery for block b_ij
16:            else
17:                Do not handle block b_ij
18:            end if
19:        end if
20:    end for
21: else if FT="Disk" then
22:    for each b_ij in FP do
23:        if b_ij ∉ RC then
24:            Define proactive recovery for block b_ij
25:        else
26:            Define appropriate reference to block b_ij such that block is not deleted
27:        end if
28:    end for
29: end if
```

Chapter 5

On Reducing Degraded Read Latency of Erasure Coded Cloud Storage

Erasure coding is gaining attraction in cloud storage systems since it improves data reliability with huge cost savings in terms of storage. However, data recovery in erasure codes includes high disk I/O, network traffic and complex decoding that impacts degraded read latency. Data access latency is one of the most important metrics to determine Quality of Service. Hence reducing degraded latency in erasure coding is vital to improve user performance. Proactive recovery techniques proposed in previous chapters can reduce degraded read latency as they reduce number of degraded reads. Hence they can improve degraded read performance of the system proposed in previous chapter. The improvement on degraded read performance may vary with respect to the selection of the proactive recovery method. However, they require modifying metadata. To reduce degraded read latency of

Nachiappan, R., Javadi, B., Neves Calheiros, R., & Matawie, K. M. (2019). ProactiveCache: on reducing degraded read latency of erasure coded cloud storage. In Proceedings of the 11th IEEE International Conference on Cloud Computing Technology and Science (CloudCom 2019), the 19th IEEE International Conference on Computer and Information Technology (CIT 2019), the 2019 International Workshop on Resource brokering with blockchain (RBchain 2019), and the 2019 Asia-Pacific Services Computing Conference (APSCC 2019), 11-13 December 2019, Sydney, Australia (pp. 223-230).

any existing erasure coded storage, in this chapter, we propose a cache based technique called ProactiveCache. It proactively copies objects from failure predicted machine into cache. On accurate failure predictions, ProactiveCache eliminates degraded read latency. ProactiveCache is evaluated using a system prototype in Ceph object store.

5.1 Introduction

Any failure in cloud storage degrades objects that are resided in the failed zone. This is applicable for both replication and erasure coding. To avoid any unnecessary repair, a delay is applied to recover data that are degraded due to failure [3]. Any read request to a degraded data in a replicated storage is handled by redirecting them to next available replica. In erasure coding, degraded data is reconstructed on the fly before it is served. Data recovery in replication is as simple as copying data from next available replica. On the other hand, in (n, k) erasure coding, data should be recovered by performing decoding using any k available chunks.

Replication maintains several copies of data on different failure zones. During temporary or permanent node/disk failure, degraded read requests are managed by redirecting read requests to next available node/disk. Hence replication will not increase access latency of degraded data. In erasure coding, any degraded read request requires to access data from k different nodes/disks and decodes them on the fly. This in turn increases access latency of degraded data.

Access latency is one of the most important metrics of Quality of Service. According to an observation form Google, Microsoft and Amazon, a service delay about 400ms can cause huge business disruption and revenue loss [144]. Erasure coding improves data reliability with huge cost savings. Addressing degraded read latency issue of erasure coding can enable erasure coding to be more pervasive in cloud storage. This can bring extensive cost savings to cloud storage especially for Big Data applications.

Several researches have focused on reducing disk I/O and network traffic during recovery process. Several works attempted to define new set of encoding and decoding methods [16, 52] to reduce recovery disk I/O and network bandwidth. However, they impact reconstruction time as reducing network traffic. Several other researches [145, 146] have

focused on parallel reconstruction. They reduce degraded read latency by applying parallel disk I/O. Few researches have focused on optimizing data read during degraded read operation. They evaluate storage node using network topology, performance, bandwidth to optimize degraded read.

Some of the recent researches have focused on defining proactive recovery methods to reduce degraded read latency of erasure codes. Many among them [53] use proactive recovery methods when the failure occurs; [147, 141, 97] have activated proactive recovery even before the occurrence of failure using failure predictions. Proactive failure handling methods are defined either with [141, 97] or without additional temporary storage overhead [147].

A study on cloud hardware reliability revealed that hard disks are the most replaced component in cloud infrastructure [30]. Experiencing simultaneous disk failures are possible in the cloud storage systems since it is composed of large number of storage disks. Predicting disk failures in advance helps to efficiently handle failures in cloud storage systems by applying some proactive failure handling mechanisms. Several researches have used some statistical and machine learning methods to improve failure prediction accuracy. Several hard disk failure prediction models are defined with range of statistical and machine learning techniques using SMART attributes [135, 136, 29].

In this chapter, we propose a system called ProactiveCache. It proactively copies failure predicted data into cache tier. Several existing cache replacement algorithms in literature suggest to keep either recently accessed data or frequently accessed data on cache. Hence existing algorithms may not help to reduce access latency for applications that do not follow any specific access pattern. It is also applicable for the applications that changes data access pattern more often. Traditional cache may not store most of the degraded objects provided an application follows a specific access pattern. ProactiveCache is designed to reduce degraded read latency by proactively copying data into cache when the failures are predicted. In case of any degraded read, data can be fetched from cache tier instead of performing data reconstruction on the fly. ProactiveCache is designed either to adapt with an existing cache or standalone (when a cluster does not have any existing cache tier).

5.2 Related Work

Inefficient data recoveries of erasure coding prevent it being more pervasive in cloud. A considerable amount of researches are constantly attempting to address the inefficient data reconstruction issues of erasure codes.

- **Coding Approach** Several researches have focused on defining new encoding methods to reduce reconstruction bandwidth and disk I/O, thereby improving read performance. Rashmi al.[52] proposed Hitchhiker system which defines new encoding and decoding technique on most popular Reed-Solomon code. Hitchhiker system uses Piggybacking framework to define a new family of code. Piggybacking framework maintains arbitrary functions. Arbitrary functions are constructed by adding functions of data form one node to another such that it reduces the amount of data transfer required for data recovery; thereby Hitchhiker reduces degraded read latency and leads to faster recovery from failure. Huang et al. [16] proposed Local Reconstruction Codes (LRC), which divides data fragments into two equal groups and maintains local parities for each group along with global parities. Local parities help to reduce number of data fragments to be read during recovery. Hence it reduces disk I/O, network bandwidth and data reconstruction time. Osama et al. [51] proposed a code called Rotated Reed-Solomon to improve I/O performance. Rotated Reed-Solomon suggests to maintain number of parities less than or equal to three.

- **Optimizing Degraded Read** Several researches [148, 149] have attempted to optimize the selection of data blocks for data reconstruction. Some researches use topology aware degraded read optimization for reducing network bandwidth. Few researches [146, 150] have attempted to assign appropriate weight for each node using various performance metrics such as capacity and speed of disks on each node or by analyzing historic response time of each node. They optimize degraded reads using the assigned weight of each node. Xingjun et al. [151] proposed degraded read optimization strategy NADE. NADE optimizes degraded read using node evaluation

method (node's weight) and distance calculation (network topology). NADE evaluate node's performance by combining a metrics choice and an analytic hierarchy process. They use network topology for the distance calculation.

- **Parallel Reconstruction** Peng et al. [145] proposed Collective Reconstruction Read (CRR) method, which utilizes parallel reconstruction to reduce degraded read latency. In CRR, data read, transfer and decode are shared among all participating nodes in parallel. Hence it reduces time complexity of degraded read from linear to logrithamic. Yunfeng et al. [146] propose a system FastDR that uses I/O parallelism to reduce degraded read latency. FastDR utilizes greedy algorithm to seek data form surviving nodes for degraded read. Hence it reduces data transfer cost.

- **Proactive Recovery** Some of the recent researches have focused on defining proactive recovery methods to reduce the degraded read latency of erasure codes. Pradeep et al. [53] proposed a novel recovery mechanism called CoARC to handle degraded read. CoARC recovers all unavailable blocks in a stripe and caches them on a separate node. CoARC reduces the job run time as it increases read performance. Peng et al. [97] proposed *Procode* which utilizes disk failure prediction methods to predict failures on HDFS. Procode proactively replicates data from failure predicted disk into a healthy disk. Peng et al. [147] proposed a proactive data migration technique, in the event of any disk failure prediction. It reduces data reconstruction time and degraded read latency.

Several researches have used some statistical and machine learning methods to improve failure prediction accuracy. Several hard disk failure prediction models use statistical and machine methods on SMART attributes [135, 136, 29]. Li et al. [113] proposed a disk drive prediction model that predicts more than 95% of failures with False Alarm Rate (FAR) under 0.1%. They have used smart attributes such as Power on Hours, Reported Uncorrectable Errors, Temperature Celsius, Spin Up Time and Seek Error Rate. They have proposed a classification tree model using aforementioned attributes to classify a disk as *good* or *failed*. They have also proposed the regression model to evaluate drive's health degree. After analysing, 1 million SATA disks, Ao et al. [32] revealed that reallocated

sector's count reflects the disk reliability deterioration. They have also proved that disk failures are predictable using reallocated sector measurements. They have also proposed RAID SHIELD, an active defence mechanism, which reconstructs failing disks before it's too late. They have developed PLATE, to provide proactive protection against single disk failure; AMOR for proactive RAID protection.

To reduce degraded read latency in cloud storage systems, in this chapter, we propose a novel caching technique using failure predictions. This technique proactively caches objects from failure predicted devices. Literature [113] shows failure predictions accuracy up to 95% with reasonable TIA. ProactiveCache configures cache tier with respect to failure predictions. ProactiveCache proactively copies failure predicted objects into cache. Hence it reduces degraded read latency. Proactive cache may also amplify read performance of storage system since fast/expensive storage devices are used for cache tier.

5.3 Background and Motivation

Nowadays object storage is a popular choice of cloud storage since this provides simple put/get interface to store and retrieve data. Netflix uses Amazon s3 which is object storage. Inefficient load balancing management and data reconstruction are important reasons that increase data access latency in erasure coding. Increased access latency of erasure coding prevents it from being more pervasive. This chapter will not address I/O latency incurred due to load imbalance. However, it will address I/O latency due to failures in object storage. In particular, this research focus on reducing degraded read latency in erasure coded storage systems.

Configuring a cache tier over an object store provides non-blocking end-to-end connectivity at cloud scale as it reduces I/O latency. Caching can improve I/O performance in both replicated and erasure coded storage systems. When there is a failure in object storage, all objects resided in failed device will enter into degraded state. They remain degraded until they are recovered. Any read request to the degraded objects are served by performing data reconstruction on the fly. Data reconstructions in erasure coding increase disk I/O and network bandwidth. This in turn will increase degraded read latency. We propose a cache based solution to reduce degraded read latency.

Traditional caching system are designed to maintain only least recently used or least frequently used data to reduce I/O latency in erasure coded storage system. However, traditional caching techniques may not be effective on reducing degraded read latency for the following reasons;

- Workloads from Facebook and Microsoft production clusters have shown that the top 5% of objects are seven times more popular than the bottom 75% [8]. This observation implies that a small number of objects are more likely to be accessed, which will get benefited from caching [152]. In this case, cache may not hold considerable amount of degraded data.
- Some scientific applications may run on different input datasets each time and it will read different objects each time. In this case, traditional caching systems are ineffective in reducing degraded read latency.
- Traditional cache tier is ineffective when an application changes data access pattern frequently. It is also applicable to the applications that do not follow any access pattern.
- Traditional caching systems need to identify most frequently used data by analysing data access pattern. Hence it may take effect only after a considerable amount of time once after it is introduced.

To handle the aforementioned cases, we have designed a new caching system called ProactiveCache. This is the novel caching technique, which utilizes various existing device failure prediction methods to forecast the devices that will fail soon. ProactiveCache caches objects form failure predicted device. Hence ProactiveCache reduces or eliminates the degraded read latency when the failure prediction accuracy is high.

5.4 ProactiveCache- A novel caching method on reducing degraded read latency

In this section, we present the design of ProactiveCache. ProactiveCache is defined on object storage and it aims to minimize or eliminate degraded read latency. If a storage sys-

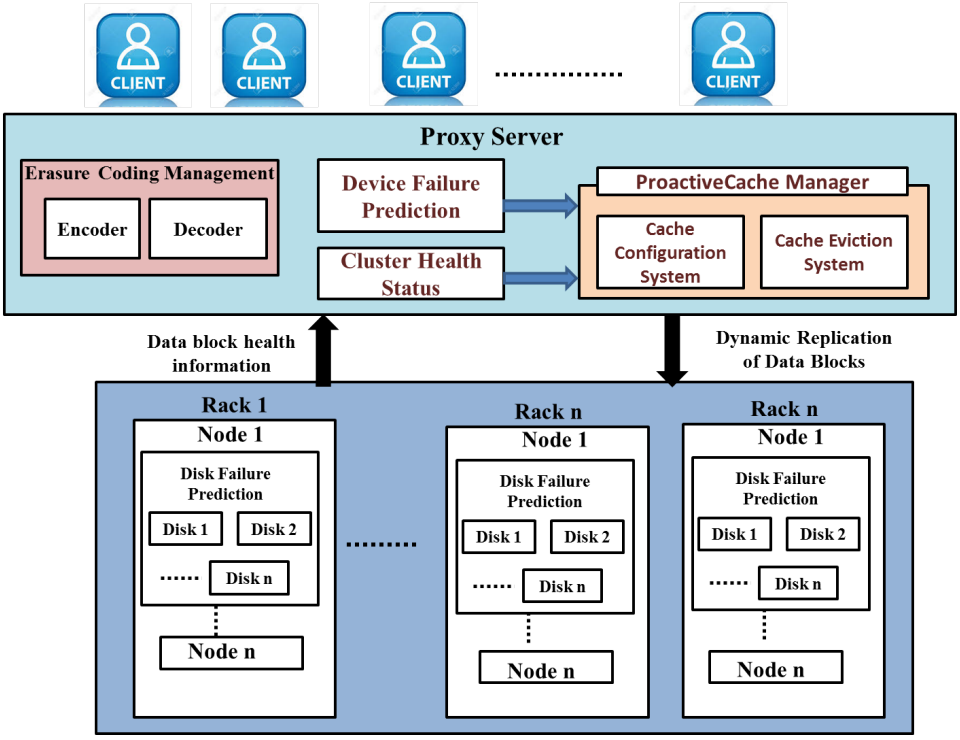


Figure 5.1: ProactiveCache: A novel caching system on eliminating degraded read latency.

tem is already configured with traditional cache, it can be upgraded with ProactiveCache method to reduce degraded read latency. If a storage system is not already configured with a cache tier, ProactiveCache will add and configure cache tier to reduce degraded read latency. An appropriate cache eviction technique is also defined to reduce storage overhead of cache tier to the minimum.

5.4.1 System Architecture

The architecture of ProactiveCache is illustrated in Figure 5.1. We have introduced several modules in existing object storage to accommodate ProactiveCache.

Data Reliability Manager

This module is designed to ensure reliability of an object in a storage system. When erasure coding is chosen to maintain data reliability, this module has to define appropriate configuration parameters to meet client SLA efficiently. The most important configuration parameters involved in erasure coding are *n, k, erasure coding technique and failure*

domain. The number of original data blocks is represented by $n - k$ and k represents the number of parity blocks. Hence any k failures from the failure domain are tolerated. Failure domain can be defined as node, rack and so on. It is important to carefully select the aforementioned parameters to minimize storage overhead and to improve reliability.

Encoder According to the selected configuration parameters, encoder segregates data into k original data fragments. It calculates $n - k$ parity fragments. It distributes all n fragments into appropriate locations such that any k failures form selected failure domain is tolerated.

Decoder On the occurrence of failures, objects in the failed devices will turn degraded and remain degraded until they are recovered. Decoder will perform data reconstruction using any $n - k$ data blocks to recover degraded object data. It migrates all recovered data into appropriate location according to the failure domain defined.

Device Failure Prediction

This module tracks hardware storage devices (SSD/HDD) and collects health metrics of those devices to predict hardware failure. Health metrics are collected using a standard called SMART for hard disks. SMART provides devices internal information like unrecoverable read errors, duration of power on and power cycles. Using those health metrics, device failures are predicted. After prediction, appropriate health alerts are generated.

Cluster Health Status

Storage cluster may contain thousands of storage nodes and it stores data as objects on storage nodes. When the cluster is operating, the health status of storage nodes and disks can be evaluated. This module continuously checks cluster health. On the occurrence of failure, it reports to the *proactive cache manager*.

Proactive Cache Manager

Proactive cache manager is designed to configure the cache tier with objects from failure predicted devices such that it reduces degraded read latency. It copies necessary objects into cache tier on the receipt of any failure predictions. This also evicts objects, when the failure predicted device has come back to life.

Cache Configuration System When a device failure prediction is reported to this module, it will copy either all or some objects from failure predicted device into cache. It uses application's data access pattern and client SLA to configure cache as follows:

- All the objects in failure predicted disk will be copied to cache, when an application does not follow any access pattern provided if SLA demands low access latency.
- Apart from reducing degraded read latency, this cache tier can also be configured to reduce access latency when there is a need for planned maintenance. During planned maintenance, this caches either the data that are likely to be accessed soon or all data in a node which undergoes maintenance.

When a storage system does not need to be configured with traditional cache tier, ProactiveCache adds and removes cache tier as required. ProactiveCache can be easily configured with existing erasure coded cloud storage without making any changes in the underlying storage systems. Hence the technique of adding a cache tier is a better solution than enforcing changes in encoding or applying proactive recoveries.

This module will also be responsible to increase storage size of cache tier as required. ProactiveCache is an elastic cache. It adjusts the capacity of the cache with respect to the rate of failures in cloud storage. Even though the cache tier must be defined with a fixed capacity, it can be expanded on the receipt of any failure predictions. *Proactive cache manager* resizes cache tier with respect to the number of objects resided in failure predicted devices. Cache will be turned back to original fixed size when the cluster has recovered all objects that are degraded due to failure. To classify objects that are cached due to the failure predictions, this module defines a unique flag. This flag is used during cache eviction process, to evict the objects that are copied due to failure predictions.

Cache Eviction System In traditional cache, eviction is performed as follows,

- Least Recently Used (LRU): This eviction method assumes that the objects which have been accessed recently are more likely to be accessed soon. When the cache is full, LRU evicts objects that are least recently accessed.

- Least Frequently Used (LFU): In this method, the most popular objects are maintained in cache over the least popular objects. This method uses access history to identify popular object. This is suitable for applications that do not change access pattern for a prolonged period of time.

Since ProactiveCache is designed to adapt with traditional caching, it performs any of the aforementioned cache eviction methods, to evict the objects that are cached due to the traditional methods when the cache tier gets full. In addition to that, it also evicts objects that are cached due to failure predictions as follows. The objects that are cached due to failure predictions are identified using the unique flag. The objects with the flags are deleted when the failure predicted machine has come back to life. In case of any false positive in failure predictions, the corresponding objects will be deleted after applying some time delay. The eviction of objects that are cached due to failure predictions will only be deleted in the aforementioned conditions. They will not be evicted when the cache is full.

5.4.2 ProactiveCache Algorithm

ProactiveCache is a novel caching technique defined in object storage to reduce degraded read latency. This cache is different from traditional cache since it is designed to cache objects from failure predicted disk. Algorithm 4 defines a layer between the objects that are cached using the traditional algorithms and ProactiveCache. A flag is used to determine if an object is cached due to the ProactiveCache technique. This flag is used to protect the objects that are cached due to ProactiveCache being deleted during typical eviction process. The objects with the flags are deleted only when the failure predicted device is recovered from failure. In case of any false positive predictions, they are deleted after applying some time delay. Mostly the time delay is defined to be greater than TIA of prediction. In this way, those objects are protected from being deleted in advance. The algorithm accepts Failure Predicted Device FPD information, aggregated total size of objects that it holds DB and current unused capacity of cache CC. Algorithm increases the cache size if it is required. Following that, it caches appropriate objects by assigning appropriate flags. If an object from failure predicted device is already available in cache,

Algorithm 5 Proactive Caching Algorithm

INPUT: FPD, DB,CC

```
1:  if cache tier exists then
2:      if ACS < DB then
3:          increase cache size
4:          for each obj_i in storage do
5:              if any fragment of obj_i contained in FPD then
6:                  copy obj_i to cache tier
7:                  set flag f_i for obj_i
8:              end if
9:          end for
10:     end if
11: else
12:     create cache tier
13:     for each obj_i in storage do
14:         if any fragment of obj_i contained in FPD then
15:             copy obj_i to cache tier
16:         end if
17:     end for
18: end if
```

it sets unique flag, to protect them being deleted form typical cache eviction algorithm.

Algorithm 5 prevents objects that are cached due the failure predictions being deleted during normal eviction process. When the cache is full, it deletes objects using typical eviction process only for the objects that are not cached due to the failure prediction. The objects replicated due to failure predictions are only deleted when the failure predicted device has come back from failure or by applying some delay in case of any false positive. While deleting those objects, it checks the flag to make sure that it does not delete objects cached by typical caching method. It also reduces cache size to default cache size when the failure predicted device has come back to life provided no further failures are predicted.

Algorithm 6 Cache Eviction Algorithm

INPUT: $Clusterhealthinformation$, $Flag$

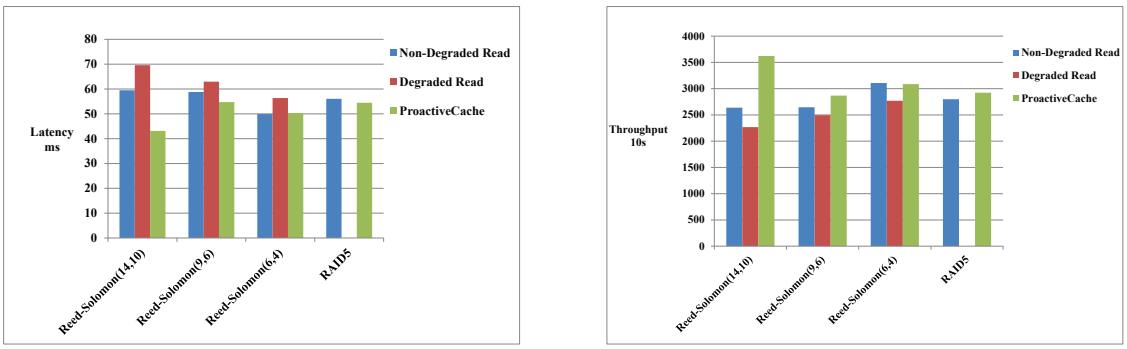
```
1: if cache is full then
2:     if Data Recovery Completed for Failure Predicted Disk then
3:         for each object obj_i in cache do
4:             if f_i is set then
5:                 evict obj_i
6:             end if
7:         end for
8:         reduce cache size to typical cache size
9:     else
10:        for all object obj_i in cache for which f_i is unset do
11:            perform eviction using LRU or LFU
12:        end for
13:    end if
14: end if
```

5.5 Performance Evaluation

To evaluate the effectiveness of ProactiveCache, we have developed a system prototype in a Ceph storage cluster [153]. This explains the cluster set up which we have used to evaluate the proposed method. In order to compare the effectiveness ProactiveCache, for each experiment, we have added an empty cache tier. To disregard any read performance gain due to traditional caching system, we have copied only the failure predicted data into cache. Once all objects from failure predicted device are copied into cache, we have changed the mode of the cache such that it avoids any further caching objects due to any I/O operation. Hence we have configured cache tier only with the failure predicted data.

5.5.1 Performance Analysis

In this section, we analyse and compare the impact of failure, on data access latency. For performance analysis, we have conducted all our experiments on the most popular



(a) (b)

Figure 5.2: Average latency and throughput of various reliability methods (a) Average latency and (b) Average throughput.

Reed-Solomon code. We will also evaluate the performance of ProactiveCache on reducing degraded read latency.

Latency and Throughput

In case of any failure in a storage system, all the objects resided in the failed device will enter into degraded state and they will remain degraded until recovered completely. Any data read request to the degraded objects are served by performing data reconstruction. Figure 5.2(a) represents a comparison of non-degraded read latency, degraded read latency in typical erasure coded storage and the latency of degraded read in Proactive cache. In Reed-Solomon(14, 10), Reed-Solomon(9, 6), Reed-Solomon(6, 4) and RAID5 degraded read latency is increased by 15% and it is 6%, 11% and indeterminate respectively, compared to its respective read latency when no objects are degraded. Degraded objects increase latency because any read request to the degraded objects should be served by performing data reconstruction. RAID5 can tolerate a single device failure. However, a single disk failure in RAID5 shows latency as indeterminate. This is due to the fact that, it has to wait indeterminate to access a data block from a disk that experiences latent sector error which may cause deadlock.

Figure 5.2 (b) shows that degraded read in erasure coded storage. A system with degraded objects reduce throughput. Figure 5.2 (b) also shows ProactiveCache improves

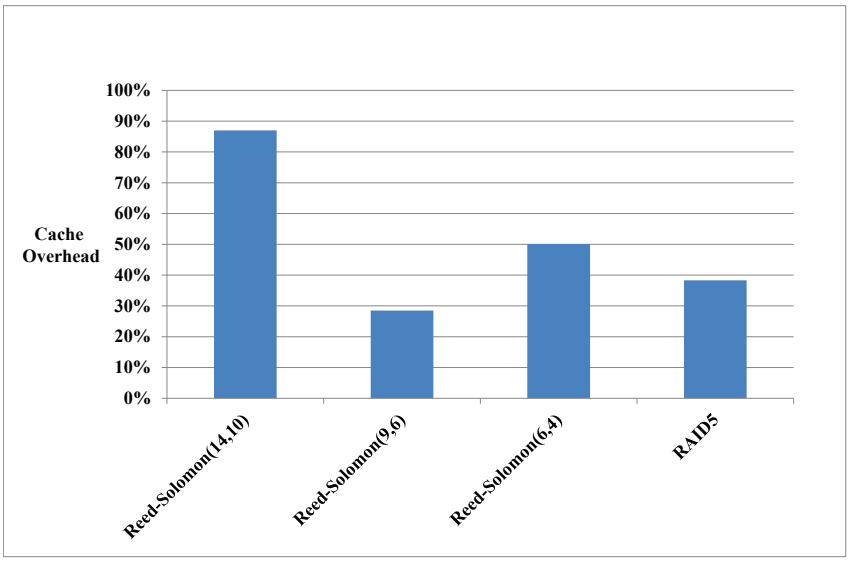
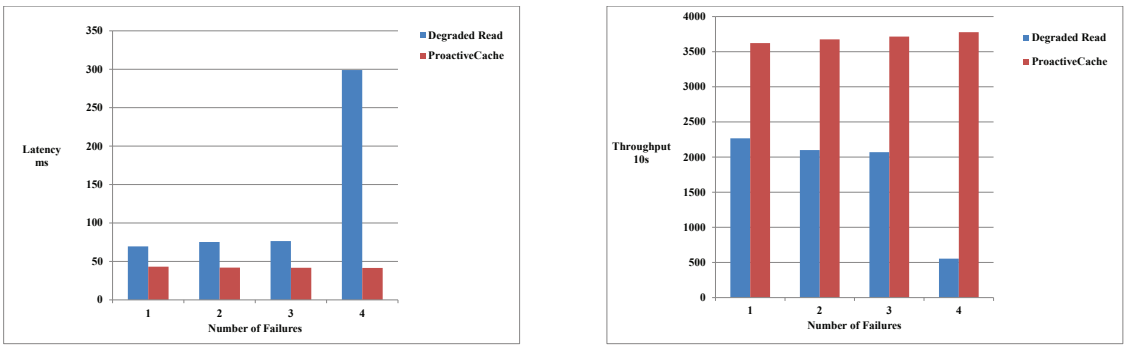


Figure 5.3: Cache Overhead.

throughput despite the existence of degraded objects.

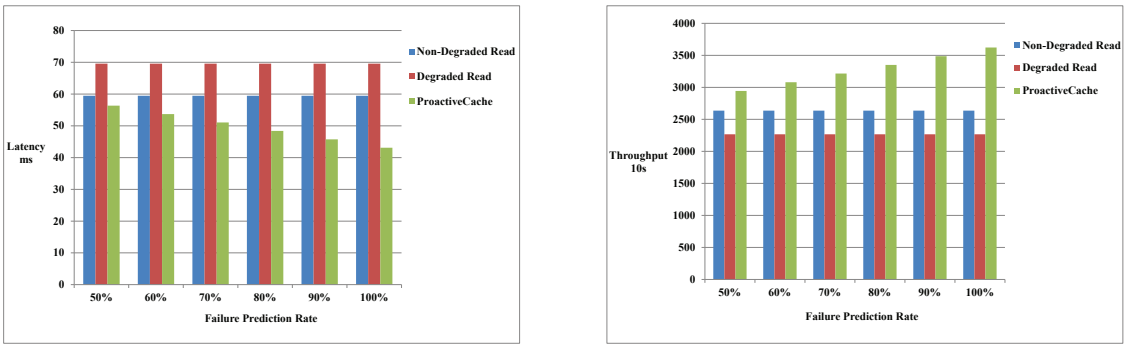
Figure 5.2 (b) shows latency of Reed-Solomon(14, 10) is more than Reed-Solomon(9, 6). In Reed-Solomon(14, 10), data has to be distributed into large number of disks compared to other method. The cluster set up used for our experiment has only 15 OSDs. Hence total of 14 chunks in Reed-Solomon(14,10) is distributed among 15 OSDs. It has to reconstruct data using 10 fragments from other OSDs. The methods Reed-Solomon(14, 10), Reed-Solomon(9, 6), Reed-Solomon(6, 4) and RAID5 decreases degraded read throughput approximately by 14% and it is 6%, 11% and indeterminate respectively compared to non-degraded read. In Figure 5.2 latency is steadily decreasing with respect to the reduction in number of data blocks required for degraded read. However, it does not apply for RAID 5, because it does not provide many choices and hence it increases I/O congestion.

Figure 5.2 also shows ProactiveCache performance during degraded read in a storage cluster. ProactiveCache reduces access latency for the methods Reed-Solomon(14, 10), Reed-Solomon(9, 6), Reed-Solomon(6, 4) and RAID5 approximately upto 38%, 13%, 11% and 100% respectively compared to those typical degraded read latency. Latency of RAID



(a) (b)

Figure 5.4: Average latency and throughput with varying number of failures. (a) Average latency and (b) Average throughput.



(a) (b)

Figure 5.5: Average latency and throughput with varying failure prediction rate. (a) Average latency and (b) Average throughput.

is indeterminate for typical degraded read. ProactiveCache has reduced latency almost equal to level of non-degraded read latency. Hence Proactive cache has reduced the latency of RAID from indeterminate to 55ms, which is 100% reduction. Reed-Solomon(14, 10) reduces access latency up to 38% since it copies 87% of data into cache as in Figure 5.3. Our experimental cluster has only 15 OSDs and Reed-Solomon(14,10) distributes an object into 14 disks. Hence it has 87% of object's data fragment in a failure predicted disk. The amount objects copied to cache tier will be reduced for the same method when the cluster size is large. ProactiveCache eliminates I/O dead lock in RAID and it improves latency by 3% compared to non-degraded read latency of RAID. ProactiveCache increases throughput for the methods Reed-Solomon(14, 10), Reed-Solomon(9, 6), Reed-Solomon(6, 4) and RAID5 approximately up to 37%, 13%, 10% and 100% respectively compared to its own methods when there is a failure. Hence there is no doubt that ProactiveCache reduces degraded read latency and improves throughput in the existence of failures.

Figure 5.4 represents how the latency and throughput are affected in Reed-Solomon(14, 10) when the number of device failure increases. It also shows the the performance improvement due to ProactiveCache. Figure 5.4shows that the latency increases when the number of failures increases. ProactiveCache method helps to reduce the latency when the number of failures increases. ProactiveCache also increases throughput when number of failures increases. ProactiveCache copies more objects into cache tier as the number of failures increases and hence it increases throughput. When there are n-k failures in (n, k) erasure code, the experiments results show a sudden peak in latency compared to n-k-1 failures. Any disk sector error could be the reason for such delay. Throughput during failures is steadily reduced as number of failures increases. However, ProactiveCache holds throughput regardless of the number of failures. ProactiveCache reduces access latency significantly when there are simultaneous failures. However, cost of copying data into cache will also increase exponentially when there are simultaneous failures.

Sensitivity Analysis

The performance of ProactiveCache will vary with respect to failure prediction accuracy. To analyse the sensitivity of ProactiveCache, we have measured latency and throughput with varying prediction rates.

Li et al. [135] showed that more than 90% accuracy of disk failure prediction is possible. We run simulation with failure prediction accuracy varying from 50% to 100% and calculated latency and throughput as shown in Figure 5.5.

From Figure 5.5, the reduction in latency due to ProactiveCache at the failure predictions rate of 100%, 90%, 80%, 70%,60% and 50% are 38%, 34%, 30%, 26%, 22% and 19% when there is a single OSD failure in Reed-Solomon (14, 10). ProactiveCache increases throughput compared to typical degraded read of Reed-Solomon (14, 10) as the failure predictions increases. However, this comes with the cost of additional cache overhead.

5.6 Summary

Latency is one of the most important metrics in cloud storage systems. Latency increase due to degraded read in replication is very limited compared to erasure coding. Even though an erasure code can define cost efficient reliable storage, degraded objects in erasure code increases data access latency. To address the degraded read latency issues of erasure codes, in this chapter, we have proposed a novel cache based solution. The novel ProactiveCache method suggests copying all objects in a failure predicted disk into cache tier in a proactive manner. ProactiveCache reduces the degraded read latency of erasure codes significantly. It does not suggest performing any changes in the underlying physical storage. It also does not suggest any changes encoding and decoding methods. It can be applied on any existing erasure codes. By reducing degraded read latency of erasure code, ProactiveCache defines cost effective reliable storage that supports data read in high velocity. ProactiveCache enables erasure coding to be a perfect solution for improving reliability of Big Data.

The next chapter explains the framework, which we have used to analyse the performance of various data reliability techniques.

Chapter 6

Framework of Efficient Fault-tolerant Cloud Storage

Replication is a repair efficient solution to improve fault tolerance in cloud storage system. However, reliability is directly proportional to the dedicated storage overhead in replication. Erasure coding is a storage efficient alternative, but it is not a repair efficient solution. To bring together the benefits of both methods, we propose a framework. This framework simulates distributed storage. It initially stores data with erasure coding. It performs proactive replication according to the failure predictions. This chapter proposes a framework called "ds-sim_Hybrid", which is implemented by extending the classes of ds-sim simulator [4] to simulate a cloud storage systems with hybrid reliability techniques. Hybrid reliability techniques are defined using replication and erasure coding. This framework is designed to conduct performance analysis of various recovery techniques that are proposed in chapter 3 and 4.

6.1 Introduction

Simulator is a better choice to estimate the expected number of data loss events that can occur over the time period of 10 years. The ds-sim simulator [4] is used in this research to estimate reliability of various storage systems. It simulates a three tier, tree structure of storage components of racks, machines and disks. It randomly chooses racks to store data

blocks in different failure domains, according to standard practices in production setting [94]. It also simulates disk, machine and rack failures. Disk failures can be latent or permanent. Latent errors are detected and recovered during periodic reads. Permanent disk failures are assumed to be unrecoverable. Machine failures can be of transient or permanent. Recovery from transient failures, begins after 15 minutes, where as it is immediate for permanent machine failures. Rack failures are considered as transient. The ds-sim records the number of degraded reads and repair bandwidth over the simulation period of 10 years.

In this research, several recovery approaches had been defined to reduce repair network, traffic in cloud storage systems. They are defined as the combinations of proactive, typical and lazy recovery. Proactive recovery suggests performing proactive replication of data from a failure predicted device. Even though this method sounds promising to reduce repair network traffic of erasure codes, this increases temporary storage overhead. On the other hand, lazy recovery applies some delay in repairing data that are degraded, due to any failures in cloud storage system. Lazy recovery activates repair, only after a certain number of data from a stripe is degraded. Collective repair in lazy recovery reduces repair bandwidth by minimizing the repair rate. However, the delay in data reconstruction affects durability and availability of data.

Both lazy and proactive recoveries can be introduced in erasure coded cloud storage systems, depending on the nature of the applications that are running on cloud storage. Some applications may demand high availability on whole. Other may demand high availability only for hot data. Some applications may even accept a delay in accessing cold data. The nature of applications can be recorded in client SLA. While a cloud storage system attempts to apply lazy recovery, it is essential to make sure that it does not compromise the durability and availability of data. Durability of data is an important metric in cloud storage system that should not be compromised at any cost. Hence an appropriate threshold should be defined while activating lazy recovery. Since different storage system selects several parameter choices of erasure coding, it is hard to define a threshold. For the applications that demand proactive recovery, it is important to decide how much additional storage is required to perform proactive replication. Failure history

can be analysed to understand the nature of failures in a cloud storage system. This analysis in cloud storage system will help to provision storage resources.

In order to analyse durability, availability, recovery network bandwidth/traffic, recovery energy consumption/energy consumption of the storage systems and temporary storage overhead due to proactive recovery, we propose a framework "ds-sim_Hybrid". This framework is an extension of ds-sim [4]. The ds-sim takes system configuration parameters such as number of original data blocks and parities. It simulates different failures latent block, machine replacement and also failures from different failure domains disk, machine and rack. It calculates recovery bandwidth, system durability, availability, durable degraded block count and available degraded block counts during a given period of simulation time. We have added several classes in an existing ds-sim to accommodate proactive recovery techniques proposed in the chapter 3. It measures the energy consumption of the storage systems. The main contribution of "ds-sim_Hybrid" is as follows,

- A component has been added to calculate system's energy consumption. Total energy consumption of the storage is calculated as a sum of energy consumption of individual disks. Energy consumption of data repair is calculated as the sum of energy consumed due to data transfer of each recovery.
- We have added components to calculate recovery energy/bandwidth for each proposed recovery methods with varying failure predictions accuracy. We have also calculated total temporary storage overhead due to the proactive replications.
- On configuration parameters, user can input different proactive recovery methods according to client SLAs.

6.2 Implementation of "ds-sim_Hybrid"

"ds-sim_Hybrid" is an extension of the simulator "ds-sim" and several components are introduced to estimate energy consumption and several proactive recovery techniques proposed in this research. The ds-sim is developed using java language. The ds-sim

accepts hardware configurations, failure/recovery parameters and failure traces as input. It distributes data according to the hardware configurations. It also generates failures and recovery events for various components using failure parameters and failure traces. Figure 6.1 shows the architecture of "ds-sim_Hybrid". The components highlighted in grey are introduced or altered in existing "ds-sim", to preform proactive replication and energy measurement.

In cloud storage cluster, client's data are stored with different reliability methods and configuration parameters. They are defined by the administrators to meet client's requirement with minimal resource usage of storage cluster. Using "ds-sim_Hybrid" system, cloud administrator can ensure that the client's reliability requirements can be satisfied for any defined configuration parameters. It helps them with analysing the resource usage in cloud storage systems for any selected parameters.

6.2.1 User Code

User code represents the inputs passed to "ds-sim_Hybrid" to simulate a storage cluster. It represents the reliability techniques, configuration parameters and recovery techniques. To define a storage cluster, the important parameters like data size, number of disks in a machine and number of disks per rack are passed. The configuration parameters of reliability methods, such as, number of original fragments and total number of fragments, including original and redundant fragments are passed. In 3 way replication, number of original fragments and total fragments are one and three, respectively. To distribute the data blocks such that the system can tolerate any defined number of failures, the total number of racks in a storage cluster is calculated on runtime, using aforementioned parameters. Various recovery methods, including proposed proactive recovery methods can be defined with "ds-sim_Hybrid" to simulate erasure coded system. Also, in order to generate various failure and recovery events, for various domains such as machines, racks and disks, the failure generators and its parameters are passed. Failures and recovery events of disk and rack are generated using Weibull generator [154]. The failure and recovery events for machines are generated using Weibull generator and real time failure traces. Apart from the aforementioned input parameters, the system will accept a parameter for

recovery type. Recovery type can be defined based on client SLA based.

Several SLA based recovery methods called ProDisk, ProMachine, ProHot and ProHot_LazyCold are proposed in this research for erasure codes. They apply different combination of recoveries using proactive replication, lazy recovery and typical reconstruction of erasure codes. Recovery methods ProHot and ProHot_LazyCold enforce different recoveries for hot and cold data. Different percentages of data are defined as *hot* in a storage cluster, depending on the behaviour of the application running on top of it. To adapt with different application's behaviour, an input parameter is defined in "ds-sim_Hybrid" to classify different amount of data *hot*. Among the storage data, a specific amount of data (according to the input parameter) is identified as hot.

6.2.2 System Generation

Once the simulator receives user inputs, it calculates required number of racks according to the configuration parameters of reliability methods and storage capacity. It also checks if the calculated number of racks is capable of maintaining each chunk of an object in different racks. According to the configuration parameter *total amount of data to be stored*, total number of objects to be maintained in the storage is calculated. Some other parameters such as chunk size and number of original fragments are also considered during this process. The ds-sim uses 256MB as chunk size. After determining number of objects, it distributes objects into disks such that the simulated storage system tolerates $n - k$ of failures. This also keeps tracks several variables to calculate average durable degraded, available degraded and latent defect of objects in a day.

6.2.3 Failure and Recovery Events Generation

This module generates failure and recovery events for components such as rack, machine and disk for the simulated period of time. Failure and recovery events of various components are generated using various generators. Failure and recovery of disk, rack, disk latent error and scrub is generated using Weibull generator. Each component uses its own value for shape, size and location parameters of Weibull. For the component machine, events such as temporary long, temporary short and permanent machine failure rates are

generated using real time traces or Weibull generator. The recovery events for the same component are generated by calculating fail fractions.

In the rest of this section, we will see how proactive recovery is implemented in the existing ds-sim. We will also see how the events are simulated to calculate durability, availability, network bandwidth/traffic, energy consumption of storage devices, energy consumption due to recovery events and temporary storage overhead due to proactive recoveries.

6.2.4 Proactive Recovery

In this section, we will see how various failure predictions are identified and proactive replications are performed, to accomplish various proactive recovery methods.

Disk/Machine Failure Predictions

This module enables the system to enforce failure predictions with different prediction accuracy. According to the user input for prediction accuracy, the system marks random amount of failures as predicted according to the input parameter of failure predictions. Failure predictions are assumed with the use of a variable. User can also input TIA of the predictions. According to the specified TIA, the prediction events are generated. Even though each prediction may use different TIA, for reducing the complexity, we use same TIA for all failure predictions.

Hot Data Predictions

To enforce the proactive recovery techniques ProHot and ProHot_LazyCold that are defined in this research, the system has to identify certain percentage of data as hot data. We designed the system such that the hot data percentage can vary according to the user input. We randomly identify user specified percentage of data as hot. We use a variable to identify hot and cold objects.

Proactive Recovery

On the event of failure predictions, all the objects in the failure predicted machine/disk are considered for proactive replication, according to the selection of a recovery method. When the system selects an object to be proactively replicated, it sets a variable to repre-

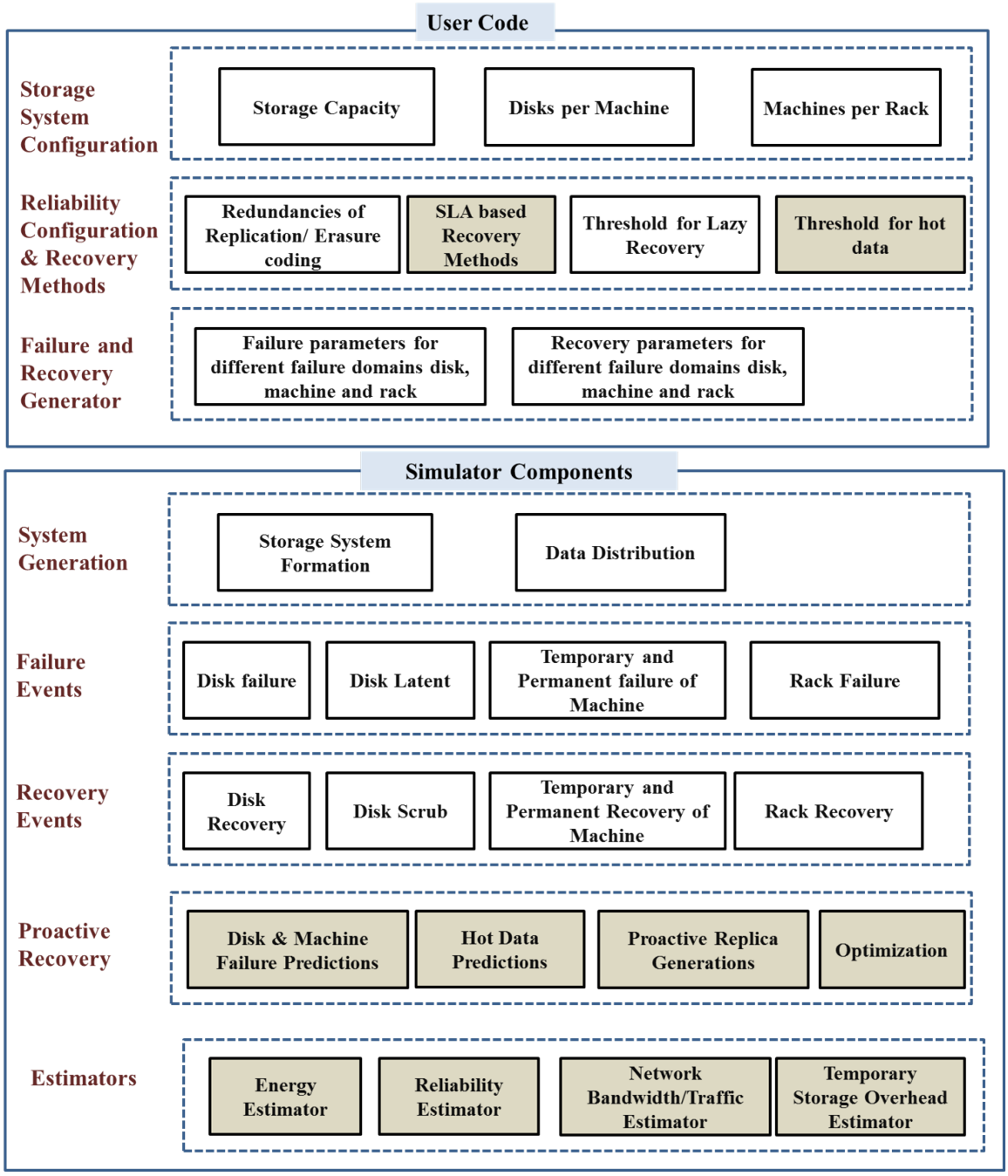


Figure 6.1: Architecture of ds-sim_Hybrid.

sent that it has an extra copy. On the occurrence of an actual failure, the aforementioned variable is examined to determine if a particular chunk of an object have an extra copy. The actual recovery of those chunks is omitted. This variable is unset on the recovery of that machine or disk. Hence the system ensures that the copy of an object is maintained, only form the time of prediction till the recovery of the same machine.

Optimization

This module is designed to implement the optimization problem, which has been defined in chapter 5. In a distributed storage, it is possible to encounter a machine failure when a disk in the machine is already predicted for a failure. In this scenario, proactive recovery could end up in maintaining two additional copies for a same chunk. It must be avoided to improve the storage efficiency of the system. When system network traffic is already high, it is optimum to avoid some proactive recovery. An optimization module has been introduced to optimize proactive recovery. In the event of failure predictions, this module ensures the system holds one a single additional replica of each in the failure predicted device. This module also calculates the projected bandwidth need for each proactive recovery to forecast the network traffic due that proactive recovery.

6.2.5 Estimators

This section sheds some lights on estimating bandwidth, energy, reliability and storage overhead of various storage systems.

Network Bandwidth/Traffic Estimator

This module calculates network bandwidth/traffic for each simulated events. For permanent machine and disk failure events, this module calculates recovery bandwidth need for current failure by calculating the amount of data to be transferred. This will vary for according to the reliability methods. For example, it is equal to chunk size for replication and k times of chunk size for erasure coding. In lazy recovery method recovery of data is delayed and amount of data has to be transferred for recovering one chunk is k, two chunks are k+1 etc. Recovery rate will also be reduced significantly in proactive recovery. This module calculates network traffic of the current failure event by summing up the recovery bandwidth of all failure events in that specific time period. When a recovery is completed, it reduces network traffic accordingly. The maximum network traffic for the simulation period is recorded as well. Recovery bandwidth of each recovery is also recorded to calculate average network bandwidth in a day. Activating proactive replication in erasure coding with 100% prediction accuracy will reduce recovery bandwidth

better than replication because erasure coding uses less number of storage devices than replication. Hence it encounters less number of failure events than replication.

Energy Estimator

Total energy consumption of the storage system is calculated by summing up the energy consumption of each disk in the storage system and energy consumption of router due to data recovery. To calculate the energy consumption of the storage devices, we have assumed that the storage devices are always active during the total simulation period. We did not consider any intervention of an application, running on top of it. Since replication needs more number of storage devices than erasure coding, replication consumes more energy, in terms of storage devices. Proactive recovery in erasure coding maintains extra copy of data from failure predicted devices. Extra copy is maintained from the time of predictions till the occurrence of actual failure of the same device. To calculate the energy consumption of the devices that hold an extra copy, every time when the predictions are encountered, energy consumption of extra devices that hold additional copies are calculated. During each failure predictions, the amounts of energy consumed by the temporary storage devices are recorded, to calculate average energy consumptions in a day due to additional temporary storage overhead. Energy consumption during data recovery is calculated in terms of amount of data transferred due to this. It is also applicable for latent sector recovery. Energy consumed during each recovery event are recorded. Using this, average energy consumption in a day, due to data recovery is calculated.

Reliability Estimator

This module calculates the reliability of data. For each failure, recovery, latent defect and scrub events, the corresponding variables such as *durableCount*, *availableCount* and *latentDefect* are changed to keep track of the reliability of data. Initially, they hold the value total number of replicas. The variable *durableCount* is reduced for disk failures and incremented after recovery. The variable *availableCount* is decremented for both machine and disk failures and they are incremented after recovery. The variable *latentDefect* is decremented for latent defect events and it is incremented after scrub. By incorporating all three variables, on occurrence each event, unavailability and undurability is calculated.

When *availableCount* or *durableCount* fall below the number of original fragments, corresponding variables *unavailable* and *undurable* are incremented. For calculating the number of durable degraded and available degraded (when a single fragment to (k+1) number of fragments are lost due to machine or disk failures) stripes in a day, the variables such as *currentSliceDegraded* and *currentAvailabSliceDegraded* are maintained. They are incremented and decremented during failure and recovery events accordingly. Using all the aforementioned variables, the durability of data is calculated for the simulated period of time.

Temporary Storage Overhead Estimator

This module calculates the temporary storage overhead due to proactive recovery. On the occurrence of any failure prediction event, according to the proactive recovery method, an extra copy of chunks will be created. An extra copy of chunks in a failure predicted machine/disk will be maintained from the time of prediction till the actual occurrence of failure of the same machine. In the event of actual failure of predicted machine/disk, an appropriate reference will be made to the existing copy. To calculate the temporary storage overhead, each time when the failure predictions are encountered, the amount of extra copies created are calculated. This will vary according to the proactive recovery methods defined for storage system. For every prediction, the total size of replicated data is recorded.

6.2.6 Sequence Diagrams

To present the work flow of various proactive recovery methods that are proposed in this research, the sequence diagram of each method is depicted. Figure 6.2, 6.3, 6.4 represent the implementation of failure handling in typical erasure coded storage and failure handling in proactive recovery methods ProDisk, ProMachine, respectively. Typical failure handling in erasure codes applies necessary changes to the variables those keep track of durable degraded, available degraded, unavailable and undurable in reliability estimator. Bandwidth estimator calculates bandwidth involved in recovering data blocks that are lost during failures. Also, energy estimator will calculate energy that is consumed due

to the data transfer during repair process. In ProDisk and ProMachine methods, when failures are predicted, reliability estimator will update respective variables to represent the extra copy of data fragments. It will also update Metadata for the extra copy which is generated due to failure prediction. Bandwidth and reliability estimators will calculate bandwidth and reliability. On the actual occurrence of failure, it will update Metadata to represent the extra replicated copy as an original fragment of the object. Hence recovery events of the predicted failures can be skipped.

Figure 6.5 and 6.6 depicts the work flow of recovery methods ProHot and ProHot_LazyCold, respectively. ProHot applies proactive replication only for hot data. ProHot_LazyCold method applies proactive replication for hot data and will apply a delay in recovery for cold data. ProHot_LazyCold will update only reliability estimator, in the event of failure prediction. Bandwidth and energy estimators will be updated when it activates lazy recovery. All proposed proactive recovery methods handle disk failure predictions in the same way. Since machine failures are mostly transient in cloud storage system, we have proposed different recoveries using client SLA and hot data status of the object. All the unpredicted failures are handled by the typical recovery of erasure code.

Upon failure predictions, the data in failure predicted machine is marked as copied. Metadata is also updated accordingly. Upon failure, using Metadata, the system determines if there is an extra copy of a chunk from failed machine. During recovery of the failed machine, instead of performing actual reconstruction, it makes some changes in Metadata such that the pre-copied data becomes the original chunk of the object. In our experiments, the system marks 40% of random data as hot. If the system does not have a proactive copy for a data chunk, it is recovered using original data reconstruction in ProHot recovery method or by applying a delay for ProHot_LazyCold method.

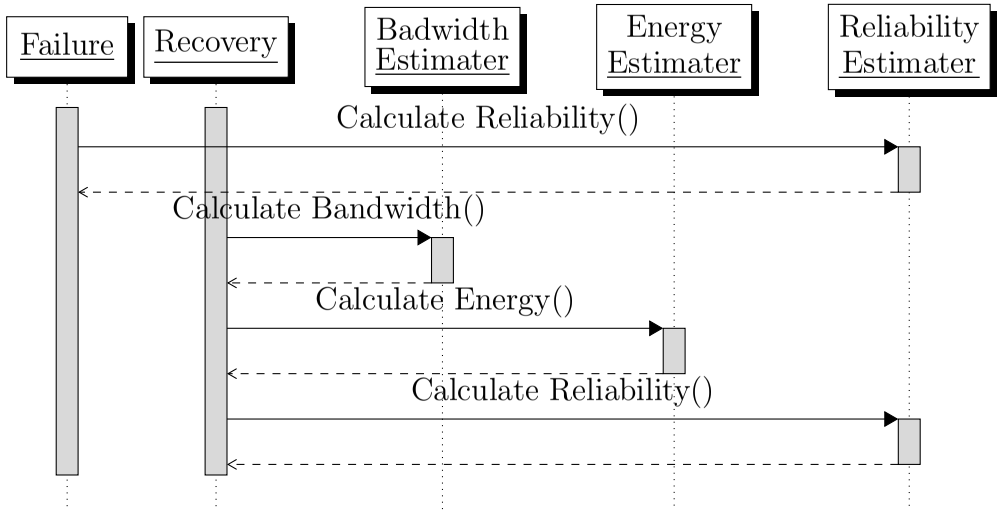


Figure 6.2: Sequence diagram of bandwidth, reliability and energy estimation in erasure coding.

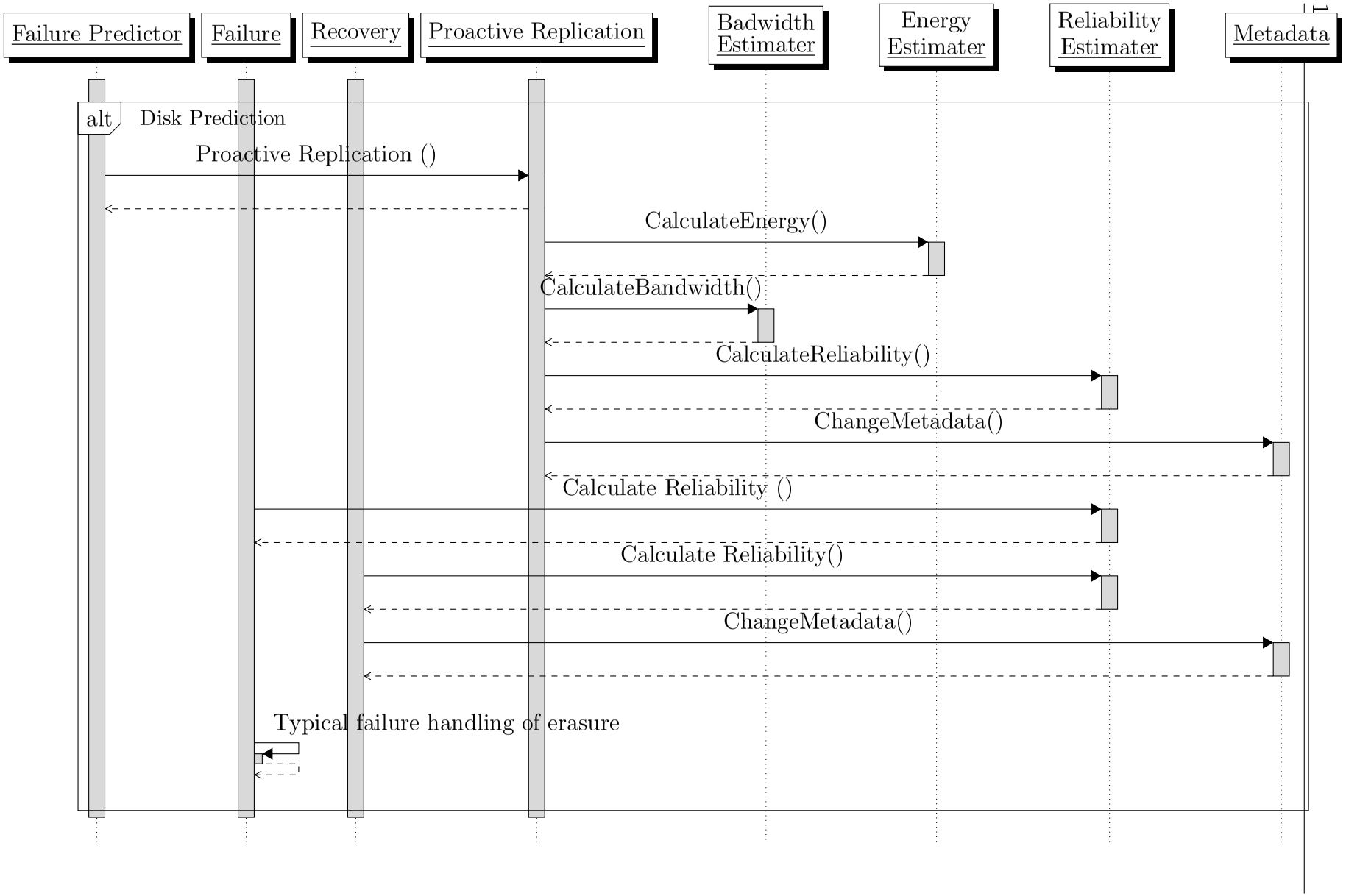


Figure 6.3: Sequence diagram of bandwidth, reliability and energy estimation in ProDisk.

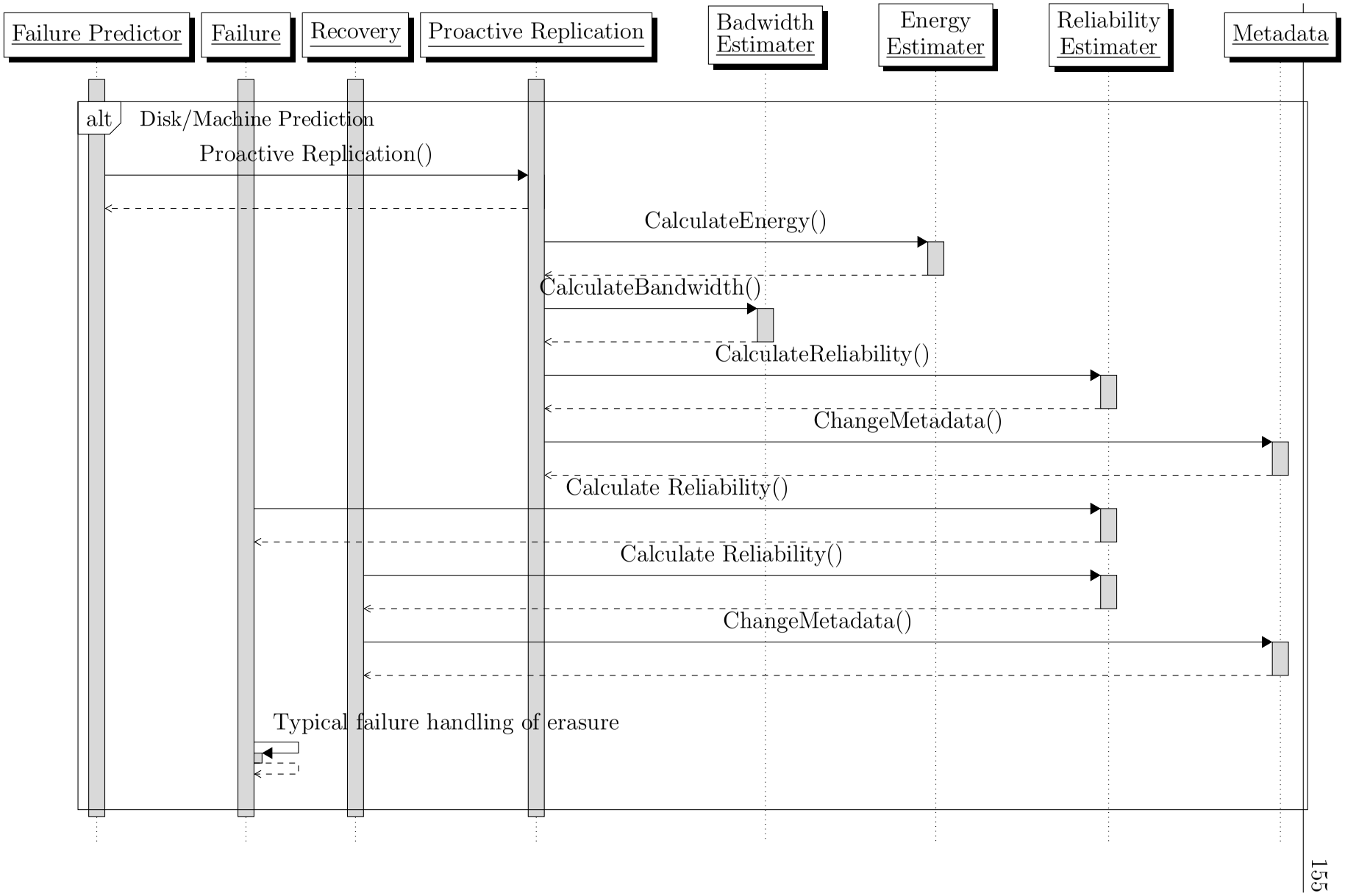


Figure 6.4: Sequence diagram of bandwidth, reliability and energy estimation in ProMachine.

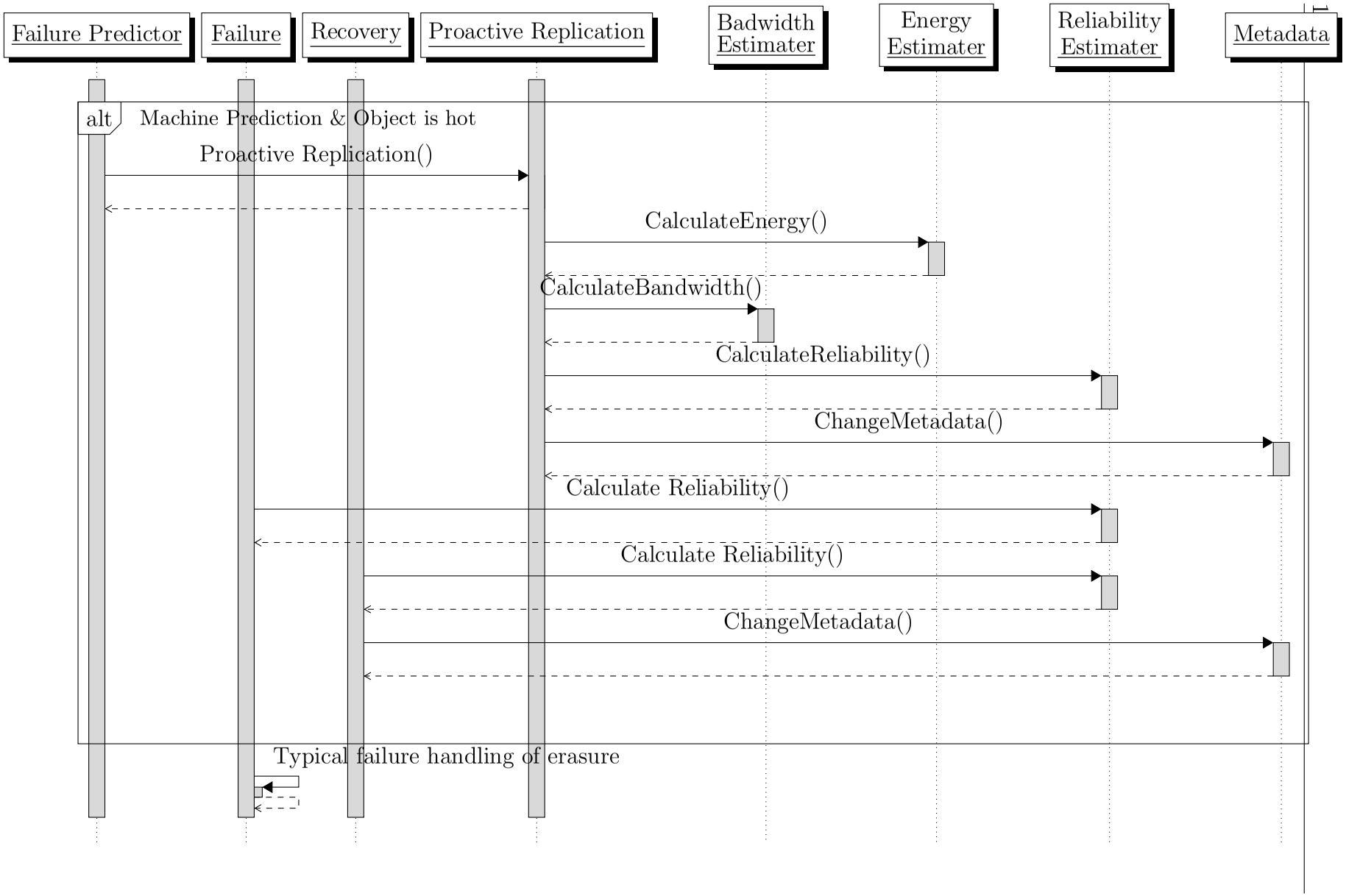


Figure 6.5: Sequence diagram of bandwidth, reliability and energy estimation in ProHot.

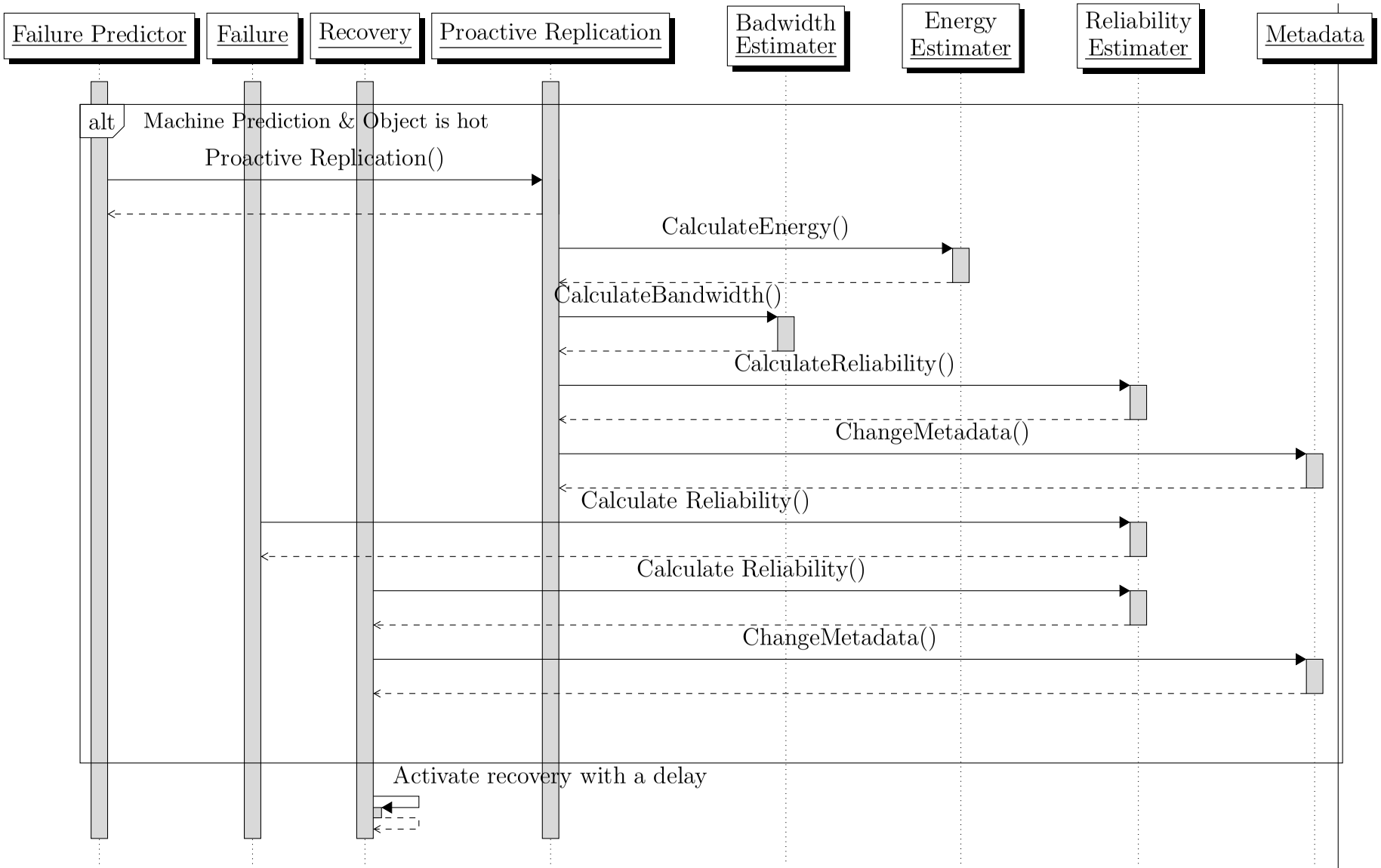


Figure 6.6: Sequence diagram of bandwidth, reliability and energy estimation in ProHot_LazyCold.

6.3 Validation

Researchers from Facebook and Google collaborated and developed ds-sim [4]. The modules of the simulator were designed to enable accurate simulation of failures in distributed storage environments [4]. Due to this, ds-sim has been selected for conducting the performance evaluation of the methods proposed in chapters 3 and 4. Several components are modified or added in ds-sim to implement ds-sim_Hybrid. They are highlighted in grey in Figure 6.1. Among them the modules such as Energy Estimator, Network Bandwidth/Traffic Estimator, Temporary Storage Overhead Estimator and Reliability Estimators are used to analyse the efficiency of cloud storage systems and they need to be validated for their accuracy. Remaining modules highlighted in grey are introduced or modified to improve the input requirements of the simulator. All modules, except Energy Estimator are already available in ds-sim and they are modified in ds-sim_Hybrid. The modules Network Bandwidth/Traffic Estimator, Temporary Storage Overhead Estimator and Reliability Estimators should have been modified to adopt proposed proactive recovery techniques.

Energy Estimator module has been introduced in ds-sim_Hybrid and it is used to compare energy consumptions of various popular erasure codes, replication and hybrid techniques that are proposed in this thesis. However, the motive of this research is to compare the energy consumption of the various reliability techniques. Therefore, energy consumption due to recovery network bandwidth and storage overhead are estimated using Energy Estimator module. Energy estimation model is presented in detail in chapter 4 in section 4.6.

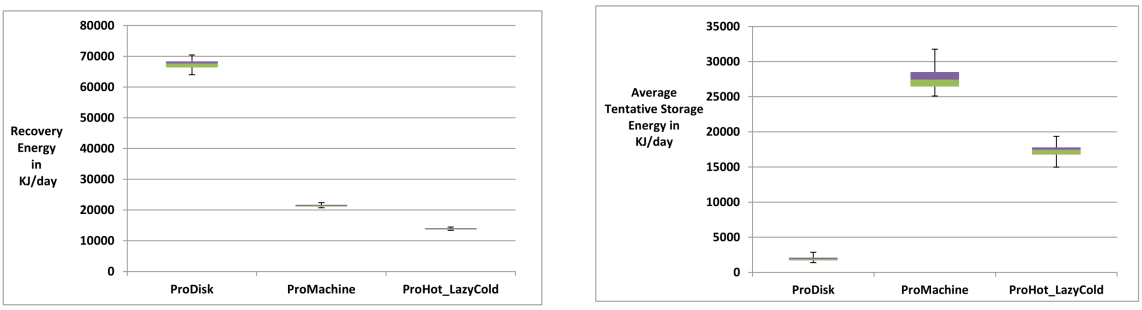
It is certain that a spike in recovery network bandwidth will eventually increase energy consumption of the storage system. This is applicable for storage overhead as well. Since real data is not available to validate Energy Estimation module, sensitivity analysis as described in [155] is used to validate this module. As defined in chapter 4, two separate energy estimators are formulated to estimate recovery and storage energy. Hence sensitivity analysis is conducted with the extreme values of network bandwidth and storage, using Analysis Of Variance (ANOVA) [156].

To analyse recovery energy estimator using ANOVA, results of ProDisk, ProHot and ProHot_LazyCold methods are utilized. The energy consumption of these methods are estimated using the parameters listed in Table 4.1 of chapter 4. The results from the aforementioned methods are used to perform ANOVA test since they show extreme variation in recovery bandwidth as well as temporary storage overhead, as discussed in chapter 3. The ANOVA test results of *recovery energy estimator* is presented in Table 6.1. The summary section of this table shows the information about each group. The *count* shows the number of samples in each group. The *average* represents the sample mean of each group. This shows the mean of different groups are lying in different range. The section *ANOVA* shows the details of *ANOVA* test. The *source of variation* depicts variation *between groups*, *within groups* and *total* re variation between and within groups. The column *sum of squares* shows sum of square values between groups, within groups and total sample. The sum of squares between groups is calculated by summing the squared differences between each group. The sum of squares within group is calculated by summing the squared differences between each observation and its group mean. The total sum of squares is calculated by summing the squared differences between each observation and the overall sample mean. The next column is *Degrees of freedom*. The degrees of freedom *between groups* are one less than the number of groups. For *within groups*, it is the difference between total sample size and number of groups. For *total*, it is one less than number of groups. Next column is mean squares. It is computed as the ratio between *sum of squares* and *degrees of freedom*. The next column is F statistic and it is computed by taking the ratio of mean squares of *between groups* and *within groups*. The *F critical* value can be found in the table probabilities of F distribution. F distribution is the distribution of all possible values of *f statistic*. Since F-value is greater than F-critical in the conducted ANOVA test, we can conclude that the mean of three aforementioned groups are different. Hence we can conclude that the results produced by the energy estimator module are statistically valid.

Similarly, in order to validate storage energy estimation, the ANOVA test has been conducted with the results of ProDisk, ProMachine and ProHot_LazyCold. Results of the test are presented in Table 6.2. Using the same method, we can conclude that the results

Table 6.1: ANOVA test results of recovery energy estimator

SUMMARY					
Groups			**Count**	**Sum**	**Average**
ProDisk			30	2025233.048	67507.76
ProMachine			30	643145.19	21438.17
ProHot_LazyCold			30	415728.18	13857.60
ANOVA					
Source of Variation	**Sum of Squares**	**Degrees of Freedom**	**Mean Square**	**F**	**F critical**
Between Groups	5.06E+10	2	1E+10	30822.57	3.101
Within Groups	71386726.01	87	820537.083		
Total	5.07E+10	89			



(a) (b)

Figure 6.7: Energy consumption in KJ per day (a) Recovery energy (b) Storage energy

produced by the energy estimator model are statistically valid.

For better understanding of how the data from three extreme groups ProDisk, ProMachine and ProHot_LazyCold has been distributed, it is graphically represented using boxplot. Figure 6.7(a) represents the data spread of energy consumption due to recovery bandwidth and Figure 6.7(b) depicts storage energy consumption. From both the figures, we can understand that the mean of the extreme groups are different and the data from each group are highly skewed.

The ANOVA test results show that the experimental results of Energy Estimator are valid. The energy savings of various reliability techniques are likely to be caused by the differences in performance of different strategies.

Table 6.2: ANOVA test results of storage energy estimator

SUMMARY					
Groups			**Count**	**Sum**	**Average**
ProDisk			30	56216.80	1873.89
ProMachine			30	826740.70	27558.02
ProHot_LazyCold			30	515279.27	17175.97
ANOVA					
Source of Variation	**Sum of Squares**	**Degrees of Freedom**	**Mean Square**	**F**	**F critical**
Between Groups	1E+10	2	5.01E+09	4955.34	3.101
Within Groups	87925869.87	87	1010642.182		
Total	10104077560.19	89			

6.4 Summary

In this section, architecture of "ds-sim_Hybrid" simulator is discussed in detail. To analyse the performance of proposed proactive recovery methods and energy consumption of storage systems, several components are added into the existing ds-sim to design "ds-sim_Hybrid". The functionality of components in "ds-sim" and "ds-sim_Hybrid" are elaborated here to understand how the various metrics such as reliability, network bandwidth/traffic, temporary storage overhead and energy consumptions are calculated while simulating failures in a distributed storage. This chapter also throws some light on implementing proposed proactive recovery methods in real cloud storage systems.

To address data recovery issues of erasure coding, several proactive recovery methods are proposed in this thesis. The proposed recovery techniques are evaluated for their performance. Experiment results showed that they substantially increase resource savings in erasure codes. Nowadays, cloud storage systems apply erasure coding only for cold data due to the data recovery issues of erasure coding. Upon selecting appropriate recovery techniques from proposed proactive recovery techniques, erasure coding can be applied for any storage systems that hold hot or cold data. Proactive recovery techniques can be applied effortlessly to existing erasure coded storage systems since it does not suggest any changes in encoding techniques.

6.5 Software Availability

ds-sim_Hybrid presented in this chapter available to download on GitHub website:

`https://github.com/umarekha/ds-sim_Hybrid`

Chapter 7

Conclusions and Future Research Directions

This chapter summarizes the research conducted in this thesis in the area of improving data reliability of cloud storage system. It also highlights the contributions of this thesis. Future research directions are also discussed in this chapter.

7.1 Conclusions

Cloud service providers are consistently striving to provide efficient and reliable service, to their client's Big Data storage need. Replication is a simple and flexible method to ensure reliability and availability of data. However, it is not an efficient solution for Big Data since it always scales in terabytes and petabytes. Hence erasure coding is gaining traction despite its shortcomings. Deploying erasure coding in cloud storage confronts several challenges like encoding/decoding complexity, load balancing, exponential resource consumption due to data repair and read latency. This thesis has addressed many challenges among them. Even though data durability and availability should not be compromised for any reason, client's requirements on read performance (access latency) may vary with the nature of data and its access pattern behaviour. Access latency is one of the important metrics and latency acceptance range can be recorded in the client's SLA. Several proactive recovery methods, for erasure codes are proposed in this research, to reduce

resource consumption due to recovery. Also, a novel cache based solution is proposed to mitigate the access latency issue of erasure coding.

Chapter 2 presented a comprehensive literature survey in the area of improving data reliability of cloud storage systems. Existing fault tolerance techniques employed in cloud storage systems were highlighted. The pitfalls of the existing methods on improving Big Data reliability, in literature were highlighted. The advantages and drawbacks of replication and erasure coding were identified with various aspects. The existing researches reducing storage overhead in replication and minimizing resource consumption in erasure coding were analysed carefully. The analysis of literature has helped to address to identify research gaps. The importance of hybrid reliability technique was discussed in chapter 2 and high level architecture to implement hybrid reliability technique using replication and erasure coding was also presented.

Chapter 3 proposed several novel proactive recovery techniques for erasure coded storage to mitigate the issue of excessive bandwidth consumption during data repair. The proposed recovery methods ProDisk, ProMachine, ProHot and ProHot_LazyCold use failure prediction techniques of cloud storage. They suggested to proactively replicating the data chunks from failure predicted device. The most important hardware failures that will compromise data durability/availability are machines and disks. ProDisk method, proactively replicates all data blocks from failure predicted disks. Since disk failures are very critical. It will compromise durability of data. Hence it suggests replicating all data blocks from failure predicted disks when they are predicted for failure. This method does not contribute much to bandwidth savings because disk failure events are very less, compared to temporary node failures which often activate recovery. To reduce the bandwidth usage due to machine failure, ProMachine method has been defined. This method proactively replicates all blocks from failure predicted machines and disks. During observation, it has been evaluated that ProMachine offered exceptional bandwidth savings. However, it also increases temporary storage overhead. To reduce the temporary storage overhead due to proactive replication, we have defined ProHot method. ProHot proactively replicates hot data and it applies normal recovery for cold data. This method reduced temporary storage overhead significantly compared to ProMachine. However, bandwidth

savings are also reduced. Some applications may require high read performance for hot data and they may accept some delay in accessing cold data. For those applications, we have proposed a method called ProHot_LazyCold. This method proactively replicates hot data and applies some delay before recovering cold data. This method substantially improved recovery bandwidth savings with limited temporary storage overhead. All the above methods also reduced durable degraded and available degraded object counts and hence increased reliability. Hence the proposed recovery techniques helps to define cost effective cloud storage for Big Data applications while also ensuring the high velocity data read to enhance the performance of Big Data applications.

To further enhance the efficiency of the existing proactive recovery techniques, an optimization problem was presented on chapter 4. In proactive recovery methods, an extra copy of data blocks should be maintained from the time of prediction till the recovery. An overlap of machine and disk failures may create unnecessary extra copy. Also, activating proactive recovery when the recovery traffic is already will lead to unnecessary throttling. To mitigate those, an optimization problem was formulated with the objective to minimize the number of data blocks replicated due to proactive recovery. Not surprisingly, the optimization of proactive replications further increased the bandwidth and temporary storage savings. The optimization also reduced system's network traffic. To analyse the energy consumption of storage systems for different reliability techniques and for erasure coding with different recovery methods, the storage energy consumptions and recovery energy consumption were estimated. The results showed that the erasure coding provided huge energy savings compared to replication in terms of storage. The analysis of recovery energy consumption in typical erasure and erasure with different proposed proactive recovery methods were conducted. The methods ProDisk, ProHot and ProMachine could not save much energy since the energy savings from recovery is compensated by the energy consumption of temporary storage. However the method ProHot_Lazy cold provided significant energy savings compared to erasure codes.

Even though the proposed proactive recovery methods contributed to reduce degraded read latency in erasure coded storage systems, in several cases, it is better to avoid making changes in Metadata. For the mission critical applications, latency is one of the most

important metrics in client SLA. To avoid the degraded read latency of erasure code, ProactiveCache is proposed in chapter 5. ProactiveCache configures a cache tier with pre-populated objects from failure predicted devices. It performs object evictions when a failure predicted device has come back to life. ProactiveCache can be configured in any object storage as stand alone or with existing traditional cache. ProactiveCache almost eliminates degraded read latency when the prediction accuracy is high. Even though it associates some cost to copy data from underlying storage to cache tier, ProactiveCache provides excellent read performance when there is a failure in underlying system. ProactiveCache enables cloud storage to enhance reliability of Big Data applications in the cost effective manner. This also supports the high-speed data read requirements of Big Data applications.

Finally, in chapter 6 the architecture of "ds-sim_Hybrid" is discussed. The simulator ds-sim is extended to implement "ds-sim_Hybrid". It can be used to analyse the energy consumption of various reliability methods. To implement "ds-sim_Hybrid", several components have been added on ds-sim such that it can perform failure prediction, hot data prediction and proactive replications of various proposed methods. The "ds-sim_Hybrid" accepts one of the proactive recovery methods as input and simulate the given method to calculate several metrics like recovery bandwidth, energy, temporary storage overhead and reliability. It also accepts hot data percentage and prediction percentage to simulate proactive recovery with various prediction accuracy rates.

7.2 Future Research Directions

Although the investigated methodologies in this thesis contribute to improve Big Data reliability, there are still several aspects that need to be explored comprehensively. This section discusses future directions.

7.2.1 Popularity driven recovery

In (n, k) erasure coded storage any failure that jeopardise a single data block to n-k-1 data blocks simply marks the object degraded and they do not affect durability and availability of data. Any read request to the object is served by performing reconstruction

on the fly. Even though repairing the objects that are degraded is important to ensure the durability and availability of data, delaying such a repair particularly when the failure domain is machine (mostly machine failures are transient) will have very minimal impact on durability and availability of data. Degraded read latency can get affected due to the delay in repair. However, a delay in accessing cold data is acceptable for some applications. A software defined recovery system has to be developed such that the system defines an appropriate recovery method by examining the access pattern behaviour of application periodically. Any changes in access pattern behaviour have to reflect on data recovery method of the erasure coded storage.

Developing system that defines access pattern defined recovery will automatically avoid unnecessary recovery and also will automatically select the failures which need immediate attention. By selecting appropriate recovery the system will not only regulate storage systems network and energy usage but also reduces access latency of hot data by providing priory to the hot data during recovery.

7.2.2 Efficient Proactive Replica Scheduling

Several proactive replication techniques for erasure codes are proposed in this thesis. However, efficient scheduling of replicas, in cloud storage is not evaluated. Replicas can be scheduled into machine which has smaller I/O queue. In geographically distributed storage, replicas can be allocated to the location where more read requests are received. By scheduling the proactive replicas, into appropriate disks/machines, the energy savings and read performance could be greatly improved. Also, activating reactive or proactive migrations in erasure codes, with respect to device failure predictions, node's read performance and workload evaluation could be other promising research directions.

7.2.3 Failure Prediction

Any disk or machine failure predictions both in replication and erasure coding help to proactively handle those failures. Hence prediction accuracy of the machine or disk failures has to be improved. Most accurate disk failure prediction methods, proposed in literature use SMART attributes. Even though they achieve high accuracy with less false positives,

they could not improve accuracy more than 95% [135, 113, 136, 29]. Failure prediction accuracy has to be improved with machine learning models such as neural networks. Machine availability and unavailability predictions have to be improved with reasonable TIA. Hence the proactive actions can be performed at right time.

7.2.4 Load balancing in Erasure Coding

Replication maintains 3 copies. Any request to read can be redirected to the node which has minimal I/O queue. In erasure coding, if a node has become a hot spot, the tail latencies are increased. Degraded read is activated only when a read time-out occur. Degraded read will select any other k nodes to access the data blocks and it performs reconstruction to obtain an original data. This approach does not only increases tail latency but also increases bandwidth usage due to degraded read. To reduce tail latency due to the poor load balancing of erasure codes, proactive hybrid reliability methods should be developed. Either an extra copy of data has to be proactively maintained or selected objects have to be proactively moved to cache to reduce tail latency in erasure codes.

7.2.5 Reliability of Decentralized Cloud Storage

Decentralized cloud storage is the future for all Big Data storage need. Decentralised storage eliminates central control and hence it improves security and privacy of the storage. In decentralised storage any node can go offline permanently. The decentralized storage Storj [107] have calculated that eight redundancies have to be maintained to improve durability of data. Since bandwidth requirement to maintain eight copies are substantially high, they have decided to use (40, 20) erasure to improve reliability [107]. Using (40, 20) erasure code the durability and availability can be assured with less storage overhead. However, this also confronts several challenges. The most important challenges of using erasure coding in decentralized cloud storage as follows:

- Applying erasure coding on streaming data is more challenging due to the encoding complexity of data. Erasure coding already involves complex encoding. The additional encodings proposed in literature may further increase the encoding com-

plexity. Hence simple and repair efficient encoding techniques must be proposed.

- In decentralized storage, nodes are distributed geographically. Since Reed-Solomon (40, 20) is a popular choice of improving reliability, data are spread into large number of nodes [107]. In case of any failure in this system, data from 20 nodes should be retrieved for repairing lost date. Hence, it provides multiple choices to select a subset of data to perform data reconstruction. To improve degraded read performance in decentralised storage, node selection for data reconstruction should be optimized using participating nodes read performance and physical distance.

7.2.6 Edge Computing and Erasure Coding

Edge computing paradigm enables data storage to be placed closer to clients and it reduce response time [157]. Data access latency of erasure coding is one of the most important issues, which prevent it being more pervasive in cloud storage. ProactiveCache proposed in this thesis suggests configuring cache tier on top of the primary storage with failure predicted data. It significantly reduces degraded read latency. To further amplify degraded read performance, cache tier of Proactive cache can be replaced with edge storage.

Storage capacity of edge tier is very limited and erasure coding could be an optimal mean to improve data reliability of the edge. However, an appropriate erasure code with minimal encoding and decoding complexity should be defined to meet computational limitations of edge.

7.2.7 Big Data and Erasure Coding

One of the important parameter of Big Data is velocity. Velocity refers both read and write efficiency. Erasure coding is the cost efficient solution to improve reliability of Big Data when considering the volume property of Big Data. However, its encoding and decoding complexity will affect the read and write performance. This thesis has addressed several challenges attributed to read performance. In order to improve the write performance of erasure codes, novel techniques should be developed such that it defines high reliability with less encoding complexity.

Bibliography

[1] David Reinsel, John Gantz, and John Rydning. "The digitization of the world: from edge to core". In: *Framingham: International Data Corporation* (2018).

[2] Maheswaran Sathiamoorthy et al. "Xoring elephants: Novel erasure codes for big data". In: *Proceedings of the VLDB Endowment*. Vol. 6. 5. VLDB Endowment. 2013, pp. 325–336.

[3] Rekha Nachiappan et al. "Cloud storage reliability for Big Data applications: A state of the art survey". In: *Journal of Network and Computer Applications* 97 (2017), pp. 35–47.

[4] Mark Silberstein et al. "Lazy means smart: Reducing repair bandwidth costs in erasure-coded distributed storage". In: *Proceedings of the International Conference on Systems and Storage (SYSTOR)*. Haifa, Israel: ACM, 2014, pp. 1–7.

[5] Sungjoon Koh et al. "Understanding system characteristics of online erasure coding on scalable, distributed and large-scale SSD array systems". In: *2017 IEEE International Symposium on Workload Characterization (IISWC)*. IEEE. 2017, pp. 76–86.

[6] John Gantz and David Reinsel. "The digital universe in 2020: Big data, bigger digital shadows, and biggest growth in the far east". In: *IDC iView: IDC Analyze the future* 2007.2012 (2012), pp. 1–16.

[7] Doug Beaver et al. "Finding a Needle in Haystack: Facebook's Photo Storage." In: *OSDI*. Vol. 10. 2010. 2010, pp. 1–8.

[8] Bart Baesens. *Analytics in a big data world: The essential guide to data science and its applications*. John Wiley & Sons, 2014.

[9] Tom Groenfeldt. *Big Data—Big Money Says It Is a Paradigm Buster*. 2012.

[10] CL Philip Chen and Chun-Yang Zhang. "Data-intensive applications, challenges, techniques and technologies: A survey on Big Data". In: *Information Sciences* 275 (2014), pp. 314–347.

[11] Daniel Ford et al. "Availability in Globally Distributed Storage Systems." In: *Osdi*. Vol. 10. 2010, pp. 1–7.

[12] Haryadi S Gunawi et al. "Towards Automatically Checking Thousands of Failures with Micro-specifications." In: *HotDep*. 2010.

[13] Peipei Wang, Daniel J Dean, and Xiaohui Gu. "Understanding Real World Data Corruptions in Cloud Systems". In: *Cloud Engineering (IC2E), 2015 IEEE International Conference on*. IEEE. 2015, pp. 116–125.

[14] AWS. *Summary of the Amazon DynamoDB Service Disruption and Related Impacts in the USEastRegion*. 2016. URL: `https://aws.amazon.com/message/5467D2/` (visited on 2016).

[15] James S Plank. "Erasure codes for storage systems: A brief primer". In: *The Usenix Magazine* 38.6 (2013), pp. 44–50.

[16] Cheng Huang et al. "Erasure Coding in Windows Azure Storage." In: *Usenix annual technical conference*. Boston, MA. 2012, pp. 15–26.

[17] Subramanian Muralidhar et al. "f4: Facebook's warm blob storage system". In: *Proceedings of the 11th USENIX conference on Operating Systems Design and Implementation*. USENIX Association. 2014, pp. 383–398.

[18] KV Rashmi et al. "A Solution to the Network Challenges of Data Recovery in Erasure-coded Distributed Storage Systems: A Study on the Facebook Warehouse Cluster." In: *HotStorage*. 2013.

[19] Danny Harnik, Dalit Naor, and Itai Segall. "Low power mode in cloud storage systems". In: *Parallel & Distributed Processing, 2009. IPDPS 2009. IEEE International Symposium on*. IEEE. 2009, pp. 1–8.

[20] Akshay Kumar, Ravi Tandon, and T Charles Clancy. "On the latency and energy efficiency of erasure-coded cloud storage systems". In: *arXiv preprint arXiv:1405.2833* (2014).

[21] Wenying Zeng et al. "Research on cloud storage architecture and key technologies". In: *Proceedings of the 2nd International Conference on Interaction Sciences: Information Technology, Culture and Human*. ACM. 2009, pp. 1044–1048.

[22] Mike Mesnier, Gregory R Ganger, and Erik Riedel. "Object-based storage". In: *IEEE Communications Magazine* 41.8 (2003), pp. 84–90.

[23] Raghavendra Kune et al. "The anatomy of big data computing". In: *Software: Practice and Experience* 46.1 (2016), pp. 79–105.

[24] James O'Reilly. *Network Storage: Tools and Technologies for Storing Your Company's Data*. Morgan Kaufmann, 2016.

[25] R Vijayakumari, R Kirankumar, and K Gangadhara Rao. "Comparative analysis of google file system and hadoop distributed file system". In: (2014).

[26] Yogesh Sharma et al. "Reliability and energy efficiency in cloud computing systems: Survey and taxonomy". In: *Journal of Network and Computer Applications* 74 (2016), pp. 66–85.

[27] Rajasekharan. *Data Reliability in Highly Fault-Tolerant Cloud Systems*. 2014. URL: `https://pdfs.semanticscholar.org/abe7/%207e70864a0d914365ed755cac5ce1abc3b8b0.pdf/` (visited on).

[28] Eric Brewer et al. "Disks for data centers". In: *White paper for FAST* 1.1 (2016), p4.

[29] Gordon F Hughes et al. "Improved disk-drive failure warnings". In: *IEEE Transactions on Reliability* 51.3 (2002), pp. 350–357.

[30] Kashi Venkatesh Vishwanath and Nachiappan Nagappan. "Characterizing cloud computing hardware reliability". In: *Proceedings of the 1st ACM symposium on Cloud computing*. ACM. 2010, pp. 193–204.

[31] Eduardo Pinheiro, Wolf-Dietrich Weber, and Luiz André Barroso. "Failure Trends in a Large Disk Drive Population." In: *FAST*. Vol. 7. 1. 2007, pp. 17–23.

[32] Ao Ma et al. "RAIDShield: characterizing, monitoring, and proactively protecting against disk failures". In: *ACM Transactions on Storage (TOS)* 11.4 (2015), p. 17.

[33] Hussam Abu-Libdeh, Lonnie Princehouse, and Hakim Weatherspoon. "RACS: a case for cloud storage diversity". In: *Proceedings of the 1st ACM symposium on Cloud computing*. ACM. 2010, pp. 229–240.

[34] Matei Zaharia et al. "Spark: Cluster computing with working sets." In: *HotCloud* 10.10-10 (2010), p. 95.

[35] Wenhao Li, Yun Yang, and Dong Yuan. "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. 2011, pp. 496–502.

[36] V Anto Vins, S Umamageswari, and P Saranya. "A survey on regenerating codes". In: (2014).

[37] Chao Jin et al. "P-Code: A new RAID-6 code with optimal properties". In: *Proceedings of the 23rd international conference on Supercomputing*. ACM. 2009, pp. 360–369.

[38] Liping Xiang et al. "Optimal recovery of single disk failure in RDP code storage systems". In: *ACM SIGMETRICS Performance Evaluation Review* 38.1 (2010), pp. 119–130.

[39] Mario Blaum et al. "EVENODD: An efficient scheme for tolerating double disk failures in RAID architectures". In: *IEEE Transactions on computers* 44.2 (1995), pp. 192–202.

[40] Yang Yixian. "Peiod Distribution for Blaum-Roth Code". In: *JOURNAL OF BEIJING UNIVERSITY OF POSTS AND TELECOMMUNICATIONS* 3 (1994).

[41] James S Plank. "The raid-6 liber8tion code". In: *The International Journal of High Performance Computing Applications* 23.3 (2009), pp. 242–251.

[42] Cheng Huang and Lihao Xu. "STAR: An efficient coding scheme for correcting triple storage node failures". In: *IEEE Transactions on Computers* 57.7 (2008), pp. 889–901.

[43] Cheng Huang, Minghua Chen, and Jin Li. "Pyramid codes: Flexible schemes to trade space for access efficiency in reliable data storage systems". In: *ACM Transactions on Storage (TOS)* 9.1 (2013), p. 3.

[44] Quan-Qing Xu et al. "CRL: Efficient Concurrent Regeneration Codes with Local Reconstruction in Geo-Distributed Storage Systems". In: *Journal of Computer Science and Technology* 33.6 (2018), pp. 1140–1151.

[45] James S Plank, Mario Blaum, and James L Hafner. "SD codes: erasure codes designed for how storage systems really fail." In: *FAST*. 2013, pp. 95–104.

[46] Mehrtash Mehrabi et al. "Improving the Update Complexity of Locally Repairable Codes". In: *IEEE Transactions on Communications* 66.9 (2018), pp. 3711–3720.

[47] Jun Li and Baochun Li. "Parallelism-Aware Locally Repairable Code for Distributed Storage Systems". In: *2018 IEEE 38th International Conference on Distributed Computing Systems (ICDCS)*. IEEE. 2018, pp. 87–98.

[48] Alexandros G Dimakis et al. "Network coding for distributed storage systems". In: *IEEE Transactions on Information Theory* 56.9 (2010), pp. 4539–4551.

[49] Korlakai Vinayak Rashmi, Nihar B Shah, and P Vijay Kumar. "Optimal exact-regenerating codes for distributed storage at the MSR and MBR points via a product-matrix construction". In: *IEEE Transactions on Information Theory* 57.8 (2011), pp. 5227–5239.

[50] Xiaoqiang Pei et al. "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. 2015, p. 14.

[51] Osama Khan et al. "Rethinking erasure codes for cloud file systems: minimizing I/O for recovery and degraded reads." In: *FAST*. 2012, p. 20.

[52] KV Rashmi et al. "A hitchhiker's guide to fast and efficient data reconstruction in erasure-coded data centers". In: *ACM SIGCOMM Computer Communication Review*. Vol. 44. 4. ACM. 2014, pp. 331–342.

[53] Subedi.P. et al. "CoARC: co-operative, aggressive recovery and caching for failures in erasure coded hadoop." In: *In Parallel Processing (ICPP), 2016 45th International Conference*. IEEE. 2016, pp. 288–293.

[54] Changho Suh and Kannan Ramchandran. "Exact-repair MDS code construction using interference alignment". In: *IEEE Transactions on Information Theory* 57.3 (2011), pp. 1425–1442.

[55] Viveck R Cadambe, Syed A Jafar, and Hamed Maleki. "Distributed data storage with minimum storage regenerating codes-exact and functional repair are asymptotically equally efficient". In: *arXiv preprint arXiv:1004.4299* (2010).

[56] Dimitris S Papailiopoulos and Alexandros G Dimakis. "Distributed storage codes through Hadamard designs". In: *Information Theory Proceedings (ISIT), 2011 IEEE International Symposium on*. IEEE. 2011, pp. 1230–1234.

[57] Changho Suh and Kannan Ramchandran. "On the existence of optimal exact-repair MDS codes for distributed storage". In: *arXiv preprint arXiv:1004.4663* (2010).

[58] Viveck R Cadambe et al. "Optimal repair of MDS codes in distributed storage via subspace interference alignment". In: *arXiv preprint arXiv:1106.1250* (2011).

[59] Itzhak Tamo, Zhiying Wang, and Jehoshua Bruck. "MDS array codes with optimal rebuilding". In: *Information Theory Proceedings (ISIT), 2011 IEEE International Symposium on*. IEEE. 2011, pp. 1240–1244.

[60] Yunnan Wu. "A construction of systematic MDS codes with minimum repair bandwidth". In: *IEEE Transactions on Information Theory* 57.6 (2011), pp. 3738–3741.

[61] Jy–yong Sohn, Beongjun Choi, and Jaekyun Moon. "A class of MSR codes for clustered distributed storage". In: *2018 IEEE International Symposium on Information Theory (ISIT)*. IEEE. 2018, pp. 2366–2370.

[62] KV Rashmi et al. "Explicit construction of optimal exact regenerating codes for distributed storage". In: *Communication, Control, and Computing, 2009. Allerton 2009. 47th Annual Allerton Conference on*. IEEE. 2009, pp. 1243–1249.

[63] Yuchong Hu, Patrick PC Lee, and Kenneth W Shum. "Analysis and construction of functional regenerating codes with uncoded repair for distributed storage systems". In: *INFOCOM, 2013 Proceedings IEEE*. IEEE. 2013, pp. 2355–2363.

[64] Kenneth W Shum and Yuchong Hu. "Functional-repair-by-transfer regenerating codes". In: *Information Theory Proceedings (ISIT), 2012 IEEE International Symposium on*. IEEE. 2012, pp. 1192–1196.

[65] Nihar B Shah et al. "Distributed storage codes with repair-by-transfer and nonachievability of interior points on the storage-bandwidth tradeoff". In: *IEEE Transactions on Information Theory* 58.3 (2012), pp. 1837–1852.

[66] Sian-Jheng Lin and Wei-Ho Chung. "Novel repair-by-transfer codes and systematic exact-MBR codes with lower complexities and smaller field sizes". In: *IEEE Transactions on Parallel and Distributed Systems* 25.12 (2014), pp. 3232–3241.

[67] Yubin Chen and Yan Wang. "On the Non-Existence of Minimum Storage Regenerating Codes With Repair-by-Transfer Property". In: *IEEE Communications Letters* 19.12 (2015), pp. 2070–2073.

[68] Yuchong Hu et al. "Cooperative recovery of distributed storage systems from multiple losses with network coding". In: *IEEE Journal on Selected Areas in Communications* 28.2 (2010).

[69] Anne-Marie Kermarrec, Nicolas Le Scouarnec, and Gilles Straub. "Repairing multiple failures with coordinated and adaptive regenerating codes". In: *Network Coding (NetCod), 2011 International Symposium on*. IEEE. 2011, pp. 1–6.

[70] Kenneth W Shum. "Cooperative regenerating codes for distributed storage systems". In: *Communications (ICC), 2011 IEEE International Conference on*. IEEE. 2011, pp. 1–5.

[71] Kenneth W Shum and Yuchong Hu. "Exact minimum-repair-bandwidth cooperative regenerating codes for distributed storage systems". In: *Information Theory Proceedings (ISIT), 2011 IEEE International Symposium on*. IEEE. 2011, pp. 1442–1446.

[72] Anyu Wang and Zhifang Zhang. "Exact cooperative regenerating codes with minimum-repair-bandwidth for distributed storage". In: *INFOCOM, 2013 Proceedings IEEE*. IEEE. 2013, pp. 400–404.

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

[74] Hanxu Hou et al. "Rack-aware regenerating codes for data centers". In: *IEEE Transactions on Information Theory* (2019).

[75] Shan Qu et al. "Multi-Rack Regenerating Codes for Hierarchical Distributed Storage Systems". In: *2018 IEEE International Conference on Communications (ICC)*. IEEE. 2018, pp. 1–6.

[76] P. Li et al. "ProCode: A Proactive Erasure Coding Scheme for Cloud Storage Systems". In: *Proceedings of the 2016 IEEE 35th Symposium on Reliable Distributed Systems (SRDS)*. Budapest, Hungary: IEEE, 2016, pp. 219–228.

[77] W. Li, Y. Yang, and D. Yuan. "Ensuring Cloud Data Reliability with Minimum Replication by Proactive Replica Checking". In: *IEEE Transactions on Computers* 65.5 (2016), pp. 1494–1506.

[78] Nicolas Bonvin, Thanasis G Papaioannou, and Karl Aberer. "Dynamic cost-efficient replication in data clouds". In: *Proceedings of the 1st Workshop on Automated Control for Datacenters and Clouds*. ACM. 2009, pp. 49–56.

[79] Asaf Cidon et al. "Copysets: Reducing the Frequency of Data Loss in Cloud Storage." In: *Usenix Annual Technical Conference*. 2013, pp. 37–48.

[80] Osama Al-Haj Hassan et al. "Replication in overlay networks: A multi-objective optimization approach". In: *International Conference on Collaborative Computing: Networking, Applications and Worksharing*. Springer. 2008, pp. 512–528.

[81] Jinwei Liu and Haiying Shen. "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. 2016, pp. 242–251.

[82] Sai-Qin Long, Yue-Long Zhao, and Wei Chen. "MORM: A Multi-objective Optimized Replication Management strategy for cloud storage cluster". In: *Journal of Systems Architecture* 60.2 (2014), pp. 234–244.

[83] Da-Wei Sun et al. "Modeling a dynamic data replication strategy to increase system availability in cloud computing environments". In: *Journal of computer science and technology* 27.2 (2012), pp. 256–272.

[84] Yanzhen Qu and Naixue Xiong. "RFH: A resilient, fault-tolerant and high-efficient replication algorithm for distributed cloud storage". In: *2012 41st International Conference on Parallel Processing*. IEEE. 2012, pp. 520–529.

[85] Mohamed-K Hussein and Mohamed-H Mousa. "A light-weight data replication for cloud data centers environment". In: *International Journal of Engineering and Innovative Technology* 1.6 (2012), pp. 169–175.

[86] Dejene Boru et al. "Energy-efficient data replication in cloud computing datacenters". In: *Cluster computing* 18.1 (2015), pp. 385–402.

[87] Zeng Zeng and Bharadwaj Veeravalli. "Optimal metadata replications and request balancing strategy on cloud data centers". In: *Journal of Parallel and Distributed Computing* 74.10 (2014), pp. 2934–2940.

[88] Jinwei Liu et al. "Popularity-aware Multi-failure Resilient and Cost-effective Replication for High Data Durability in Cloud Storage". In: *IEEE Transactions on Parallel and Distributed Systems* (2018).

[89] John D Cook, Robert Primme, and Ab de Kwant. "Compare cost and performance of replication and erasure coding". In: *hitachi Review* 63 (2014), p. 304.

[90] Jun Li and Baochun Li. "Erasure coding for cloud storage systems: a survey". In: *Tsinghua Science and Technology* 18.3 (2013), pp. 259–272.

[91] Hakim Weatherspoon, John Kubiatowicz, et al. "Erasure Coding Vs. Replication: A Quantitative Comparison." In: *IPTPS*. Vol. 1. Springer. 2002, pp. 328–338.

[92] WK Lin, Dah Ming Chiu, and YB Lee. "Erasure code replication revisited". In: *Peer-to-Peer Computing, 2004. Proceedings. Proceedings. Fourth International Conference on*. IEEE. 2004, pp. 90–97.

[93] Rodrigo Rodrigues and Barbara Liskov. "High availability in DHTs: Erasure coding Vs. replication". In: *Peer to Peer Systems IV* (2005), pp. 226–239.

[94] Daniel Ford et al. "Availability in globally distributed storage systems". In: (2010).

[95] Julio Araujo, Frédéric Giroire, and Julian Monteiro. "Hybrid approaches for distributed storage systems". In: *Data Management in Grid and Peer-to-Peer Systems* (2011), pp. 1–12.

[96] Yadi Ma et al. "An ensemble of replication and erasure codes for cloud file systems". In: *INFOCOM, 2013 Proceedings IEEE*. IEEE. 2013, pp. 1276–1284.

[97] Peng Li et al. "ProCode: A Proactive Erasure Coding Scheme for Cloud Storage Systems". In: *Reliable Distributed Systems (SRDS), 2016 IEEE 35th Symposium on*. IEEE. 2016, pp. 219–228.

[98] Zouheir Daher and Hassan Hajjdiab. "Cloud Storage Comparative Analysis Amazon Simple Storage vs. Microsoft Azure Blob Storage". In: *International Journal of Machine Learning and Computing* 8.1 (2018).

[99] Yaser Mansouri, Adel Nadjaran Toosi, and Rajkumar Buyya. "Cost optimization for dynamic replication and migration of data in cloud data centers". In: *IEEE Transactions on Cloud Computing* (2017).

[100] AWS. *Amazon S3 Storage Classes*. 2019. URL: `https://aws.amazon.com/s3/storage-classes/` (visited on 2019).

[101] Ali R Butt et al. "Exploring Trade-Offs between Energy Savings and Reliability in Storage Systems". In: *The Green Computing Book: Tackling Energy Efficiency at Large Scale* (2014), p. 149.

[102] Eduardo Pinheiro, Ricardo Bianchini, and Cezary Dubnicki. "Exploiting redundancy to conserve energy in storage systems". In: *ACM SIGMETRICS Performance Evaluation Review*. Vol. 34. 1. ACM. 2006, pp. 15–26.

[103] Kevin M Greenan et al. "A Spin-Up Saved Is Energy Earned: Achieving Power-Efficient, Erasure-Coded Storage." In: *HotDep*. 2008.

[104] Daniel Ford et al. "Availability in Globally Distributed Storage Systems". In: *Proceedings of the 9th USENIX Symposium on Operating Systems Design and Implementation (OSDI)*. Vancouver, Canada: USENIX, 2010.

[105] J. S. Plank. "T1: erasure codes for storage applications". In: *Proceedings of the 4th USENIX Conference on File and Storage Technologies (FAST)*. San Francisco, USA: USENIX, 2005, pp. 1–74.

[106] Yaser Mansouri and Rajkumar Buyya. "Dynamic replication and migration of data objects with hot-spot and cold-spot statuses across storage data centers". In: *Journal of Parallel and Distributed Computing* 126 (2019), pp. 121–133.

[107] JT Olio. *Storj*. Nov 6, 2018(accessed 2019). URL: `https://storj.io/blog/2018/11/replication-is-bad-for-decentralized-storage-part-1-erasure-codes-for-fun-and-profit/`.

[108] Alexandros G Dimakis et al. "Network coding for distributed storage systems". In: *IEEE transactions on information theory* 56.9 (2010), pp. 4539–4551.

[109] M. Sathiamoorthy et al. "Xoring elephants: Novel erasure codes for big data". In: *Proceedings of the VLDB Endowment* 6.5 (2013), pp. 325–336.

[110] Cheng Huang et al. "Erasure Coding in Windows Azure Storage". In: *Presented as part of the 2012 USENIX Annual Technical Conference (USENIX ATC 12)*. Boston, USA: USENIX, 2012, pp. 15–26.

[111] KV Rashmi et al. "A hitchhiker's guide to fast and efficient data reconstruction in erasure-coded data centers". In: *ACM SIGCOMM Computer Communication Review* 44.4 (2015), pp. 331–342.

[112] Jing Li et al. "Hard drive failure prediction using Decision Trees". In: *Reliability Engineering & System Safety* 164 (2017), pp. 55–65.

[113] Jing Li et al. "Hard drive failure prediction using classification and regression trees". In: *Proceedings of the 44th Annual IEEE/IFIP International Conference onDependable Systems and Networks (DSN)*. Atlanta, USA: IEEE, 2014, pp. 383–394.

[114] Bahman Javadi et al. "The Failure Trace Archive: Enabling the comparison of failure measurements and models of distributed systems". In: *Journal of Parallel and Distributed Computing* 73.8 (2013), pp. 1208–1223.

[115] Tomasz Agrawal Bikash and Wiktorski and Chunming Rong. "Analyzing and Predicting Failure in Hadoop Clusters Using Distributed Hidden Markov Model". In: *Cloud Computing and Big Data*. Cham: Springer International Publishing, 2015, pp. 232–246.

[116] Suzhen Wu, Hong Jiang, and Bo Mao. "Proactive data migration for improved storage availability in large-scale data centers". In: *IEEE Transactions on Computers* 64.9 (2015), pp. 2637–2651.

[117] Suzhen Wu et al. "Improving availability of raid-structured storage systems by workload outsourcing". In: *IEEE Transactions on Computers* 60.1 (2011), pp. 64–79.

[118] Dong Dai et al. "Provenance-based object storage prediction scheme for scientific big data applications". In: *Big Data (Big Data), 2014 IEEE International Conference on*. IEEE. 2014, pp. 271–280.

[119] James S Plank et al. "A Performance Evaluation and Examination of Open-Source Erasure Coding Libraries for Storage." In: *Fast*. Vol. 9. 2009, pp. 253–265.

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

[121] Kumar.A, Tandon. R, and Clancy. T. C. "On the latency and energy efficiency of distributed storage systems." In: *IEEE Transactions on Cloud Computing* 5.2 (2017), pp. 221–233.

[122] Atefeh Khosravi, Saurabh Kumar Garg, and Rajkumar Buyya. "Energy and carbon-efficient placement of virtual machines in distributed cloud data centers". In: *European Conference on Parallel Processing*. Springer. 2013, pp. 317–328.

[123] Alexandros G Dimakis et al. "Network coding for distributed storage systems". In: *IEEE Transactions on Information Theory* 14.1 (2010), pp. 4539–4551.

[124] Nachiappan.R et al. "Cloud storage reliability for Big Data applications: A state of the art survey." In: *Journal of Network and Computer Applications* 97 (2017), pp. 35–47.

[125] Rashmi. K. V, Shah. N. B, and P. V. Kumar. "Optimal exact-regenerating codes for distributed storage at the MSR and MBR points via a product-matrix construction." In: *IEEE Transactions on Information Theory* 57.9 (2011), pp. 5227–5239.

[126] Cheng Huang et al. "Erasure Coding in Windows Azure Storage". In: *Usenix annual technical conference*. 2012, pp. 15–26.

[127] Maheswaran Sathiamoorthy et al. "Xoring elephants: Novel erasure codes for big data". In: *Proceedings of the VLDB Endowment*. Vol. 6. 5. VLDB Endowment. 2013, pp. 325–336.

[128] Rashmi.K. V et al. "Network coding for distributed storage systems". In: *ACM SIGCOMM Computer Communication Review* 44.4 (2015), pp. 331–342.

[129] Silberstein.M et al. "Lazy means smart: Reducing repair bandwidth costs in erasure-coded distributed storage". In: *In Proceedings of International Conference on Systems and Storage*. ACM. 2014, pp. 1–7.

[130] Peng Li et al. "Procode: A proactive erasure coding scheme for cloud storage systems". In: *2016 IEEE 35th Symposium on Reliable Distributed Systems (SRDS)*. IEEE. 2016, pp. 219–228.

[131] John Wilkes et al. "The HP AutoRAID hierarchical storage system". In: *ACM Transactions on Computer Systems (TOCS)* 14.1 (1996), pp. 108–136.

[132] Julio Araujo, Frédéric Giroire, and Julian Monteiro. "Hybrid approaches for distributed storage systems". In: *International Conference on Data Management in Grid and P2P Systems*. Springer. 2011, pp. 1–12.

[133] Xu.W and Y. Lu. "Energy Analysis of Hadoop Cluster Failure Recovery." In: *In Parallel and Distributed Computing, Applications and Technologies*. IEEE. 2013, pp. 141–146.

[134] Huang.J, Zhang.Fa nd Qin.X, and Xie. C. "Exploiting redundancies and deferred writes to conserve energy in erasure-coded storage clusters." In: *ACM Transactions on Storage* 9.2 (2013), p. 4.

[135] Li.J et al. "Hard drive failure prediction using classification and regression trees." In: *44th Annual IEEE/IFIP International Conference*. IEEE. 2014, pp. 383–394.

[136] Li. J et al. "Hard drive failure prediction using Decision Trees". In: *ACM SIGCOMM Computer Communication Review* 164 (2017), pp. 55–65.

[137] Javadi.B et al. "The Failure Trace Archive: Enabling the comparison of failure measurements and models of distributed systems." In: *Journal of Parallel and Distributed Computing* 73.8 (2013), pp. 1208–1223.

[138] Bikash Agrawal, Tomasz Wiktorski, and Chunming Rong. "Analyzing and predicting failure in hadoop clusters using distributed hidden Markov model". In: *Second International Conference on Cloud Computing and Big Data in Asia*. Springer. 2015, pp. 232–246.

[139] Jinwei Liu and Haiying Shen. "A popularity-aware cost-effective replication scheme for high data durability in cloud storage". In: *2016 IEEE International Conference on Big Data (Big Data)*. IEEE. 2016, pp. 384–389.

[140] J. Liu and H. Shen. "A popularity-aware cost-effective replication scheme for high data durability in cloud storage". In: *Proceedings of the 2016 IEEE International Conference on Big Data (Big Data)*. Washington, DC, USA: IEEE, 2016, pp. 384–389.

[141] Nachiappan. R et al. "Adaptive Bandwidth-Efficient Recovery Techniques in Erasure-Coded Cloud Storage". In: *In European Conference on Parallel Processing*. Springer. 2018, pp. 325–338.

[142] Ray Quattromini. "Green Data Storage". In: *Fortuna Power Systems Ltd*. 2017.

[143] Vishwanath.A et al. "Estimating the energy consumption for packet processing, storage and switching in optical-IP routers." In: *In Optical Fiber Communication Conference*. Optical Society of America. 2013, p. 6.

[144] Yaochen Hu and Di Niu. "Reducing access latency in erasure coded cloud storage with local block migration". In: *IEEE INFOCOM 2016-The 35th Annual IEEE International Conference on Computer Communications*. IEEE. 2016, pp. 1–9.

[145] Peng Li et al. "Parallelizing degraded read for erasure coded cloud storage systems using collective communications". In: *2016 IEEE Trustcom/BigDataSE/ISPA*. IEEE. 2016, pp. 1272–1279.

[146] Lee Patrick PC Zhu Yunfeng Lin Jian and XuYinlong. "Boosting degraded reads in heterogeneous erasure-coded storage systems". In: *IEEE Transactions on Computers* 64.8 (2014), pp. 2145–2157.

[147] Peng Li. "Enabling Low Degraded Read Latency and Fast Recovery for Erasure Coded Cloud Storage Systems". In: *2017 47th Annual IEEE/IFIP International Conference on Dependable Systems and Networks Workshops (DSN-W)*. IEEE. 2017, pp. 164–167.

[148] Zhirong Shen et al. "Cross-rack-aware single failure recovery for clustered file systems". In: *IEEE Transactions on Dependable and Secure Computing* (2017).

[149] Jing Zhang et al. "Aggrecode: Constructing route intersection for data reconstruction in erasure coded storage". In: *IEEE INFOCOM 2014-IEEE Conference on Computer Communications*. IEEE. 2014, pp. 2139–2147.

[150] Ping Xie et al. "SmartRec: fast recovery from single failures in heterogeneous RAID-coded storage systems". In: *The Computer Journal* 61.6 (2017), pp. 896–911.

[151] Xingjun Zhang et al. "NADE: nodes performance awareness and accurate distance evaluation for degraded read in heterogeneous distributed erasure code-based storage". In: *The Journal of Supercomputing* (2019), pp. 1–30.

[152] Raluca Halalai et al. "Agar: A caching system for erasure-coded data". In: *2017 IEEE 37th International Conference on Distributed Computing Systems (ICDCS)*. IEEE. 2017, pp. 23–33.

[153] Ceph. *Ceph documentation*. 2016(accessed 2019). URL: `https://docs.ceph.com/docs/master/`.

[154] Horst Rinne. *The Weibull distribution: a handbook*. Chapman and Hall/CRC, 2008.

[155] Jack PC Kleijnen. "Validation of models: statistical techniques and data availability". In: *Proceedings of the 31st conference on Winter simulation: Simulation—a bridge to the future-Volume 1*. 1999, pp. 647–654.

[156] Lars St, Svante Wold, et al. "Analysis of variance (ANOVA)". In: *Chemometrics and intelligent laboratory systems* 6.4 (1989), pp. 259–272.

[157] Amir Vahid Dastjerdi and Rajkumar Buyya. "Fog computing: Helping the Internet of Things realize its potential". In: *Computer* 49.8 (2016), pp. 112–116.