Locally Repairable Codes (LRCs)

Shahin Majazi

August 3, 2023

Contents

Abstract

This report explores the world of Locally Repairable Codes (LRCs), an essential class of error-correcting codes with applications in modern data storage and communication systems. The abstract focuses on understanding the fundamental concepts and properties of LRCs, examining the challenges related to their implementation, and investigating the encoding methodologies used to achieve local repairability.

In this study, we delve into the advantages of LRCs over conventional error-correcting codes, emphasizing their ability to achieve fault tolerance with reduced storage overhead. LRCs offer efficient recovery mechanisms, allowing data reconstruction with limited access to neighboring storage units, which enhances the system's reliability and performance.

The practical implications of LRCs in distributed storage environments are also discussed, highlighting their significance in large-scale data centers and cloud-based services. The abstract concludes by underlining the importance of LRCs as a viable and promising solution for safeguarding data integrity and availability in modern data-centric applications.

Keywords: Locally Repairable Codes, Error-Correcting Codes, Fault Tolerance, Data Storage, Distributed Systems.

Chapter 1

Introduction

1.1 Background

Channel coding and Locally Repairable Codes (LRCs) are two crucial concepts in the realm of error correction and data integrity assurance in modern communication and storage systems.

Channel coding, also known as error-correcting coding, plays a fundamental role in mitigating errors that occur during data transmission over noisy channels. When data is transmitted through a communication channel, it is susceptible to noise, interference, and other disturbances, which can lead to data corruption. Channel coding techniques employ sophisticated algorithms to add redundancy to the transmitted data, enabling the receiver to detect and correct errors that may have occurred during transmission. Classic examples of channel coding include Hamming codes, Reed-Solomon codes, and Turbo codes, each offering varying degrees of error-correction capabilities.

On the other hand, Locally Repairable Codes (LRCs) have gained significant attention in the context of distributed storage systems. In the era of massive data centers and cloud storage services, ensuring high data availability and fault tolerance is of paramount importance. LRCs offer an innovative solution to efficiently maintain data redundancy and enable rapid data recovery in the event of node failures. Unlike traditional erasure codes, which require accessing a significant number of nodes during repair operations, LRCs enable local repair, allowing a newcomer node to reconstruct lost data by interacting with only a limited subset of existing nodes. This local repair property significantly reduces the repair bandwidth and enhances the overall performance and reliability of distributed storage systems.

The development and study of Locally Repairable Codes have become an active area of research due to their compelling advantages in terms of fault tolerance, storage overhead, and repair efficiency. Researchers and engineers are continuously exploring new design methodologies and optimization techniques to further improve the performance and practical applicability of LRCs in real-

world scenarios.

In this report, we delve into the intricacies of channel coding and Locally Repairable Codes, examining their underlying principles, properties, and applications. We aim to explore the latest advancements and challenges in both fields and highlight their significance in ensuring reliable and efficient data communication and storage in contemporary data-driven environments. Through a comprehensive analysis, we endeavor to shed light on the potential synergy between these two critical domains and identify avenues for future research and innovation in error correction and data protection technologies.

1.2 Motivation

In recent years, distributed and cloud storage systems have witnessed an unprecedented scale, becoming integral to daily operations rather than rare occurrences. Ensuring high data availability and robustness against multiple node failures has become imperative for such large-scale storage systems. To achieve these objectives, redundancy is introduced among stored data, where erasure coded storage systems emerge as a powerful approach that achieves high reliability without incurring the storage costs associated with data replication.

However, a critical challenge arises in the form of the Repair Problem: maintaining the encoded representation in the face of node failures or erasures. When a storage node departs from the system, a newcomer node must join the array, access existing nodes, and precisely reproduce the lost contents to maintain the same level of redundancy. In this repair process, several metrics can be optimized, such as the total information read from existing disks, the repair bandwidth (total bits communicated in the network), or the total number of disks needed for each repair. Repair bandwidth is the most well-understood metric currently.

Classical codes like Reed-Solomon, although widely used, exhibit significant inefficiencies in distributed environments, especially during single-failure events. This inadequacy has spurred the emergence of a new family of codes called Locally Repairable Codes (LRCs), which offer repair efficiency. LRCs, notably, minimize the number of nodes involved in single-node repairs, leading to improved performance and reliability.

Challenges in the field of Locally Repairable Codes encompass various aspects. The first major challenge revolves around the Repair Problem itself, seeking to design codes that facilitate exact and low-cost repairs of nodes while maintaining arbitrarily high data rates. Achieving optimal minimum distance and code rate for LRCs is another prominent challenge, as well as ensuring high data rates without compromising repair efficiency.

Additionally, the design and implementation of LRCs present challenges in balancing various trade-offs. For instance, improving repair efficiency might require trade-offs in terms of storage overhead or communication complexity during repairs. Achieving a well-rounded LRC solution necessitates careful consideration of these trade-offs to create a code that optimizes performance across

different scenarios.

This report delves into the motivation and challenges associated with LRCs, exploring the latest advancements in LRC research and the strategies employed to tackle these challenges. By understanding and overcoming these challenges, we can unlock the full potential of Locally Repairable Codes and enhance their applicability in modern data storage and transmission scenarios.

1.3 Necessary Definitions

Before delving into the details of Locally Repairable Codes (LRCs) and their properties, we introduce several essential definitions that form the foundation for our discussions:

M: File Size
The variable M represents the size of the file being stored or transmitted. It serves as a fundamental parameter in LRCs, influencing the design and efficiency of the code.

r: Repair Locality
The repair locality, denoted by r, refers to the number of nodes involved in the repair process when a single-node failure occurs. A high repair locality is desirable in LRCs, as it minimizes the number of nodes participating in repairs, contributing to faster and more localized recovery.

d: Minimum Distance of a Code
The minimum distance, d, of an LRC code is a crucial measure of its error-correcting capability. It represents the minimum number of errors or erasures that the code can detect and correct.

(n, k, r, d, α)-LRC: The (n, k, r, d, α)-LRC notation represents an LRC code with specific parameters:

n: Total number of storage nodes in the system.
k: Number of original data (message) symbols.
r: Repair locality, as defined earlier.
d: Minimum distance of the code, as defined earlier.
α: Entropy of Each Encoded Element

The variable α represents the entropy of each encoded element or the bit size for each encoded symbol.

The entropy value influences the amount of information encoded in each symbol and affects the overall storage efficiency of the LRC.

Understanding these definitions is vital for grasping the subsequent discussions on LRCs and their performance characteristics.

1.4 Relationship between field size and LRCs

Here, we present a table summarizing the relationship between $\mathbb{F}_q$ and known (n, k, r)-LRCs in previous years:

Table 1.1: Relationship between $\mathbb{F}_q$ and LRCs

Construction year	Constraints on the parameters	Field size (q)
2012	$n = \left[\frac{k}{r}\right](r+1)$	$q \geq n$
2014	$(r+1)\|n$	$q \geq 2^{\frac{n \cdot r}{r+1}}$
2013	$(r+1)\|n$	$q \geq n$
2016	$(r+1)\|n$	$q \geq \left(\frac{n \cdot r}{r+1}\right)^{k+1}$

Chapter 2

Locally Repairable Codes (LRCs)

2.1 Challenges and Limitations

In the design and analysis of Locally Repairable Codes (LRCs), it is essential to consider various cost metrics that directly influence the efficiency of the repair process. Three significant cost metrics have been identified:

Repair-Bandwidth: This metric represents the number of bits communicated in the network during the repair process. Minimizing the repair bandwidth is crucial to reduce network congestion and optimize data transfer efficiency.

Disk I/O (Input/Output): Disk I/O measures the number of bits read from existing disks during the repair process. Efficiently managing disk I/O is critical in large-scale distributed storage systems to avoid excessive load on individual disks and optimize data retrieval operations.

Repair Locality: Repair locality refers to the number of nodes involved in the repair process when a single-node failure occurs. Maximizing repair locality enhances fault tolerance and reduces the impact of node failures on the overall system.

Balancing these cost metrics is a challenging task, as optimizing one may come at the expense of another. An effective LRC design should strike the right balance to achieve both high repair efficiency and fault tolerance.

2.2 Explanation

Locally Repairable Codes (LRCs) represent an essential class of error-correcting codes with applications in modern data storage and communication systems. In this chapter, we present the fundamental construction and properties of LRCs, which provide fault tolerance and data reliability in distributed storage environments.

An (n, k, r)-LRC is a code of length n, designed to take k information symbols as input, such that any of its n output coded symbols can be recovered by accessing and processing at most r other symbols (i.e., an LRC has all-symbol locality). Codes with all-symbol locality that meet the above distance bound are termed optimal LRCs and are known to exist when $(r+1)$ divides n. Some works extend the designs and theoretic bounds to the case where repair bandwidth and locality are jointly optimized during multiple local failures. Additionally, designing LRCs with optimal distance for all code parameters n, k, r that are suitable for deployment in distributed storage clusters is currently an active research area.

We start with an $(m = n \cdot \frac{r}{r+1}, k)$-Reed-Solomon code, and then we proceed with a re-encoding process as follows: the m Reed-Solomon coded symbols are divided into $\frac{m}{r}$ groups, each containing r symbols without overlapping. Each group is then re-encoded using a specific $(r+1, r)$ Maximum-Distance-Separable (MDS) code. The n outputs of the $\frac{m}{r}$ local codes form the encoded symbols of the LRC.

Notably, the locality r is effortlessly achieved by the local codes. When a symbol is missing, the remaining r coded symbols in its group can be utilized to reconstruct it, which is a property of the MDS local code. Although the code, as presented in this figure, is not initially in systematic form, it can be readily transformed into one through a linear transformation applied to the generator matrix.

The subsequent sections of this chapter will explore the advantages of LRCs over conventional error-correcting codes, examine the challenges related to their implementation, and discuss the encoding methodologies used to achieve local repairability.

2.3 Theorems without proving

Theorem 1: An $C(n, M, r, d, \alpha)$ code will satisfy:

$$d \leq n - \left[\tfrac{M}{\alpha}\right] - \left[\tfrac{M}{r\alpha}\right] + 2$$

Special case:

$$\text{If } M = k \text{ and } \alpha = 1, \text{ then } d \leq n - k - \left[\tfrac{k}{r}\right] + 2.$$

Theorem 2: Achievability of the distance bound when $(r + 1)|n$:

Let $(r + 1)|n$ and $r \leq n - d$, then exist (n, M, r, d, α)-LRC with minimum code distance.

$$d = n - [\tfrac{M}{\alpha}] - [\tfrac{M}{r\alpha}] + 2$$

Special case:

$$\text{If } M = k \text{ and } \alpha = 1 \text{ , then } d = n - k - [\tfrac{k}{r}] + 2$$

2.4 Code Construction

Below we state the circular placement of elements in nodes of the first $(r+1)$-group

Table 2.1: Cicular placement of elements in nodes of the first $(r+1)$-group

Blocks	**node 1**	**node 2**	...	**node** r	**node** $r+1$
blocks of $\mathbf{y^{(1)}}$	$y_1^{(1)}$	$y_2^{(1)}$	...	$y_r^{(1)}$	$y_{r+1}^{(1)}$
blocks of $\mathbf{y^{(2)}}$	$y_2^{(2)}$	$y_3^{(2)}$	...	$y_{r+1}^{(2)}$	$y_1^{(2)}$
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
blocks of $\mathbf{y^{(r)}}$	$y_r^{(r)}$	$y_{r+1}^{(r)}$	...	$y_{r-2}^{(r)}$	$y_{r-1}^{(r)}$
blocks of $\mathbf{s}$	s_{r+1}	s_1	...	s_{r-1}	s_r

Methodology:

In this section we present a simple construction of an optimal (n,k,r)-LRC, assuming that $(r+1)|n$: Let $m = n \cdot \frac{r}{r+1}$ and assume that $1 < r < k$. In this construction method, we take the output of an (m,k)-Reed-Solomon (RS) code, divide it into $\frac{m}{r}$ non-overlapping groups, each consisting of r coded symbols of each group into $r+1$ new symbols using specific $(r+1,r)$ MDS code. We refer to this $(r+1,r)$ MDS code as a local code. The aforementioned construction will be shown to have i) the desired locality r and ii) otimal minimum distance. In 2.1, we give a sketch of the construction.

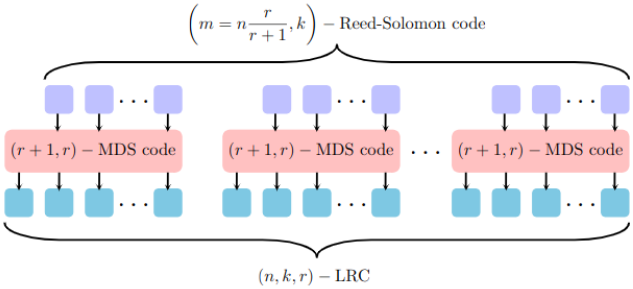

Figure 2.1: A sketch of (n,k,r)-LRC construction

Fig. 2.1:

More formally, let $\mathbb{F}_p$ be a field of size $p > m$, where p is a power of some prime. Consider a file that is cut into k symbols

$$x = [x_1, \ldots, x_k]$$

, where each symbol is an element of the field $\mathbb{F}_{p^{k+1}}$. These k symbols are encoded into n coded symbols

$$y = [y_1, \ldots, y_n]$$

$$y = x \cdot G$$

where G is the $k \times n$ generator matrix, with elements over the field $\mathbb{F}_{p^{k+1}}$. The construction of G follows.

Construction: Let $\alpha_1, \ldots, \alpha_m$ be $m = \frac{n \cdot r}{r+1}$ distinct elements of the field $\mathbb{F}_p$, with $p > m$, and ω be a primative element of the field $\mathbb{F}_{p^{k+1}}$. Also let V be a $k \times m$ Vandermonde matrix with its i-th column being equal to

$$\overline{\alpha_i} = (1, \alpha_i, \ldots, \alpha_i^{k-1})^T$$

. Then, the generator matrix of the code is

$$G = V \cdot (I_{m/r} \otimes A)$$

, where I_s is the identity matrix of size s and $A = (a_{i,j})$ is an $r \times (r+1)$ matrix defined as follows: it has 1_s on the main diagonal, and ωs on the diagonal whose elements $\alpha_{i,j}$ satisfy $j - i = 1$. An example of an $(r = 3, r+1 = 4)$ A matrix is given below

$$A = \begin{pmatrix} 1 & \omega & 0 & 0 \\ 0 & 1 & \omega & 0 \\ 0 & 0 & 1 & \omega \end{pmatrix}$$

Encoding on top of existing Reed-Solomon stripes:

Construction above has a very important property: the coded symbols can be generated using Reed-Solomon coded symbols. This design flexibility of the presented codes comes in sharp contrast to other known schemes, which requires decoding the entire already stored information; a process that can be cost-inefficient, when it comes to large-scale storage applications. In our case, when RS-symbols are already stored in a system, the only coding overhead is in re-encoding groups of r RS-coded symbols, in $r+1$ coded symbols, which can be done with complexity $O(n)$, since A has at most 2 elements per row and column.

Recall that the first step encoding of construction above uses RS encoded symbols over $\mathbb{F}_{p^{k+1}}$, where the evaluation points are taken from the sub-field $\mathbb{F}_p$: In other words, we evaluate a polynomial of $\mathbb{F}_{p^{k+1}}[x]$ at the points of the

field $\mathbb{F}_p$. The following lemma shows that grouping together $k+1$ RS encoded symbols over $\mathbb{F}_p$ can be viewed as a single encoded RS symbol over $\mathbb{F}_{p^{k+1}}$. Hence, already encoded RS symbols over $\mathbb{F}_p$ can be used in the first encoding step, by a simple grouping, and the entire construction reduces to performing only the second step of the local encoding.

Lemma:

Each coded symbol of an (m,k)-RS code over $\mathbb{F}_{p^{k+1}}$ evaluated at points of the field $\mathbb{F}_p$ has an equivalent representation as a vector of $k+1$ coded symbols of an (m,k)-RS code over $\mathbb{F}_p$.

- ***Proof:***

Let $\beta = (\beta_0, \ldots, \beta_{k-1})$ be the information symbols to be encoded, where each $\beta_i \in \mathbb{F}_{p^{k+1}}$ and ω be a primitive element of $\mathbb{F}_{p^{k+1}}$, then each symbol β_i can be written as a polynomial in ω of degree at most k and coefficients from $\mathbb{F}_p$, namely

$$\beta_i = \sum_{j=0}^{k} \beta_{i,j}\omega^j, \text{ and } \beta_{i,j} \in \mathbb{F}_p$$

The RS symbol over $\mathbb{F}_{p^{k+1}}$ is simply the evaluation of the polynomial $f_\beta(x)$ defined as

$$f_\beta(x) = \sum_{i=0}^{k-1} \beta_i x^i,$$

at m distinct elements of the field $\mathbb{F}_p$. However

$$f_\beta(x) = \sum_{i=0}^{k-1} \beta_i x^i = \sum_{i=0}^{k-1}\sum_{j=0}^{k} \beta_{i,j}\omega^j x^i = \sum_{j=0}^{k} \omega^j g_j(x)$$

where $g_j(x)$ is a polynomial of degree at most $k-1$ over $\mathbb{F}_p$. Specifically for $\alpha \in \mathbb{F}_p$

$$f_\beta(\alpha) = \sum_{i=0}^{k} \omega^j g_j(\alpha)$$

In other words, the evaluation of $f_\beta(x)$ at the point $\alpha \in \mathbb{F}_p$ equals to the summation of $k+1$ summands and each summand is a product of some power of ω with an evaluation of a polynomial of degree less than k over $\mathbb{F}_p$. There for, if $g_j(\alpha)$ for $j = [0, \ldots, k]$ are $k+1$ RS encoded symbols derived by evaluating $k+1$ polynomials $g_j(x)$ over $\mathbb{F}_p$ at the point $\alpha \in \mathbb{F}_p$, then can be viewed as a one encoded RS symbol over $\mathbb{F}p^{k+1}$ evaluated at $\alpha \in \mathbb{F}_p$. This symbol can be represented as a vector of length $k+1$ over $\mathbb{F}_p$: this is done by a simple concatenation of the $k+1$ symbols into the vector $(g_0(\alpha), \ldots, g_k(\alpha))$.

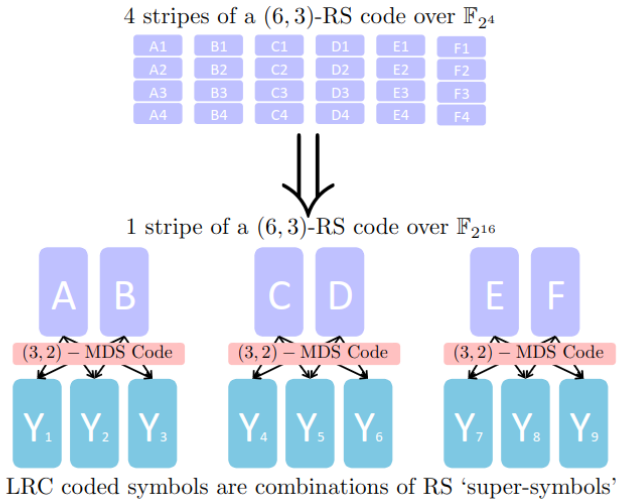

Figure 2.2: Example of a $(n = 9, k = 3, r = 2)$-LRC using construction.

Fig. 2.2:

Assume that $k + 1 = 4$ stripes of RS-encoded data over $\mathbb{F}_{2^4}$ are stored in 6 nodes. Each node can be viewed as storing a single RS symbol over $\mathbb{F}_{(2^4)^4}$. The second step encoding partitions the symbols into pairs, and generates from any pair, 3 symbols over $\mathbb{F}_{2^{16}}$, to provide the locality property. in Fig. 2.2, we give an example of a $(9, 3, 2)$-LRC. In the first step of the encoding, we evaluate the polynomial at $m = \frac{n \cdot r}{r+1} = 6$ distinct points of the field $\mathbb{F}_p$, hence $p > 6$. Assuming we would like to operate over binary characteristics, we pick $p = 2^4$ (note that also $p = 2^3$ would suffice).

Table 2.1:

There are three key properties that are obeyed by our placement:

i) Each node contains r coded blocks coming from different $\mathbf{y}^{(1)}$ coded vectors and 1 additional parity element.

ii) The blocks in the $r + 1$ nodes of the i-th $(r + 1)$-group, have indices that appear only in that specific repair group, $i \in [\frac{n}{r+1}]$.

ii) The blocks of each "row" have indices that obey a circular pattern, i.e., the first row of elements has indices $\{1, 2, \ldots, r+1\}$ the second $\{2, 3, \ldots, r+1, 1\}$, and so on.

Fig. 2.3:

We follow the same placement for all $\frac{n}{r+1}$ groups. In 2.3, we show an LRC of the above construction with $n = 6$ and $k = 4$ and $r = 2$. Observe that we spent

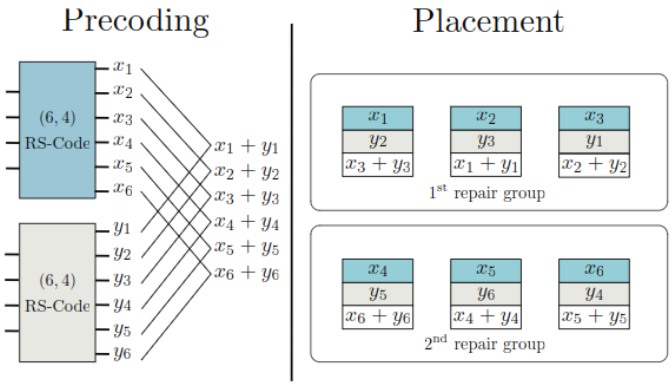

Figure 2.3: A $(6, 4)$-LRC with $r = 2$

3 blocks per coded element, instead of $\frac{M}{k} = 2$. This allows us to have easy and repairability with locality 2. The distance of this code is guaranteed through the 2 $(6, 4)$-MDS pre-codes.

2.5 Repairing Lost Nodes

We will concentrate on the repair of lost nodes in the first repair group of r nodes. This is sufficient since the block placement is identical across repair groups. The key observation is that each node within a repair group stores $r+1$ blocks of distinct indices: the $r + 1$ blocks of a particular index are stored in $r + 1$ distinct nodes within the repair group. Hence, when for example the first node fails, then element $\mathbf{y_1^{(1)}}$ is regenerated by downloading $\mathbf{s_1}$ from the second node in the group, $\mathbf{y_1^{(r+1)}}$ from the third, ..., and $\mathbf{y_1^{(2)}}$ from the last node in the group. A simple XOR of the above blocks suffices to reconstruct the lost block. Hence, for every lost block of a failed node, the r remaining blocks of the same index that are stored in the r remaining nodes of the repair group have to be XORed to regenerate what was lost.

Fig. 2.4:

The repair locality is 2 since 2 remaining nodes are involved in reconstructing the lost information of the first node, or the first coded element. Observe that we repair by only transferring blocks: no block combinations are need to be performed at the sender nodes.

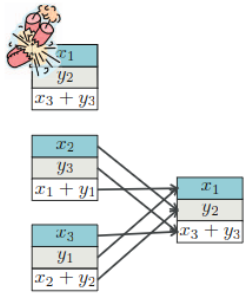

Figure 2.4: The repair of a node failure.

2.6 Rate and Code Distance

The effective coding rate of the codes and distance of them will be

$$d = n - k + 1$$

$$R = \frac{\text{size of useful nformation}}{\text{total storage spent}} = \frac{M}{n \cdot \alpha} = \frac{r \cdot k}{(r+1) \cdot n}$$

That is, the rate of the code is a fraction $\frac{r}{r+1}$ of the coding rate of an (n, k) MDS code, hence is always upper bounded by $\frac{r}{r+1}$. This is due to the extra storage overhead required to store the parity blocks s_i, $i \in [n]$.

Chapter 3

Graphs

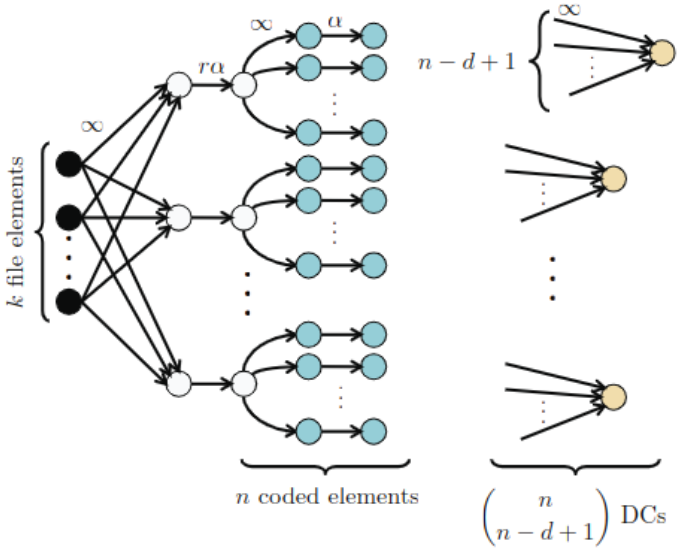

Figure 3.1: General flow graph.

Fig. 3.1:

We sketch the graph where the black vertices correspond to the k sources, each of which has entropy $\frac{M}{k}$. The $\frac{n}{r+1}$ white vertices represent nodes that bottleneck the in-flow within a repair group. The blue vertices correspond to the n nodes, or coded elements, and the yellow vertices are the DCs (sinks) of the network.

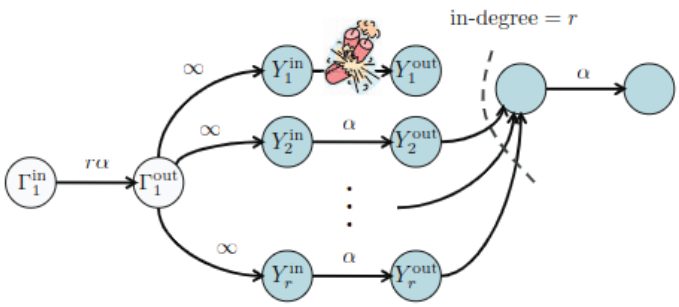

Figure 3.2: Flow bottlenecks.

Fig. 3.2:

When an element is lost, a new node can join the system and download everything from the remaining nodes in its group. In Fig. 2, we show the part of the flow graph that enforces the "bottleneck" induced by a repair group, where we consider the first group, without loss of generality. For a group $\Gamma(i)$, $i \in [\frac{n}{r+1}]$, we add node Γ_i^{in} that receives. Flow from the sources and is connected with an edge of capacity $r\alpha$ to a new node Γ_i^{out}. The latter connects to the $r+1$ elements of the i-th group. We should note that when we are considering a specific group, it is implied that any block within that group can be repaired from the remaining r elements. When a block is lost, the functional dependence among the elements of that group allows a newcomer block to compute a function on the remaining r elements and reconstruct what was lost. Linear combinations of the file elements travel along the edges of this graph towards the sinks, which we call Data Collectors (DCs). A DC needs to connect to as many coded elements as such that it can reconstruct the file. This is equivalent to requiring source-to-sink $(s-t)$ cuts between the file elements and the DCs that are at least equal to M, i.e., the file size. An $s-t$ cut in $G(n,k,r,d,\alpha)$ determines the amount of flow, or entropy, that can travel from the source elements to the destinations. then the minimum of all the cuts is at least as much as the file size M.

Chapter 4

Conclusion

In conclusion, Locally Repairable Codes (LRCs) offer significant advantages over older encoding methods, making them a compelling choice in modern data storage and transmission systems.

One key advantage of LRCs is their enhanced fault tolerance. Unlike traditional coding schemes, LRCs enable the repair of a single failed storage unit without requiring access to the entire code. This local repair property reduces the strain on resources and substantially improves the system's reliability and availability.

Furthermore, LRCs provide improved data recovery capabilities. When data loss occurs due to disk failures or other issues, LRCs allow for efficient recovery by accessing only a limited number of other storage units. This reduction in retrieval complexity minimizes the time and computational resources needed for data reconstruction, leading to faster recovery processes.

Another remarkable benefit of LRCs is their efficiency in terms of storage overhead. Compared to traditional encoding methods, LRCs achieve high fault tolerance with a lower redundancy level. This means that a smaller portion of additional storage space is required to provide the same level of error correction capability, resulting in more efficient utilization of resources.

Moreover, LRCs exhibit excellent performance in distributed storage systems. As data centers and cloud-based services become increasingly prevalent, the ability to locally repair codes is crucial for large-scale distributed storage environments. LRCs reduce the need for extensive data transfers during repairs, minimizing network congestion and enhancing overall system performance.

In summary, the advantages of Locally Repairable Codes over older encoding methods lie in their enhanced fault tolerance, efficient data recovery, reduced storage overhead, and superior performance in distributed storage systems. These benefits position LRCs as a highly attractive solution for modern data-centric applications, where data integrity, availability, and cost-effectiveness are paramount considerations.

References

Bibliography

[1] IEEE 2012: Locally Repairable Codes

[2] IEEE 2014: Locally Repairable Codes

[3] arXiv 2022: Good locally repairable codes via propagation rules

[4] IEEE 2016: Optimal Locally Repairable Codes and Connections to Matroid Theory

[5] IEEE 2021: Construction of Binary Locally Repairable Codes With Optimal Distance and Code Rate

[6] arXiv 2022: Bounds and Constructions of Singleton-Optimal Locally Repairable Codes with Small Localiti