UNIVERSIDAD DE GRANADA

Departamento de Ingeniería de Computadores,
Automática y Robótica

Programa de Doctorado en
Tecnologías de la Información y la Comunicación

Tesis Doctoral

Técnicas de códigos de borrado aplicadas a almacenamiento distribuido

Realizada por:

Fabricio Rolando Marcillo Vera

Granada 2023

Editor: Universidad de Granada. Tesis Doctorales
Autor: Fabricio Rolando Marcillo Vera
ISBN: 978-84-1117-896-9
URI: https://hdl.handle.net/10481/82551

UNIVERSIDAD DE GRANADA

Departamento de Ingeniería de Computadores,
Automática y Robótica

Programa de Doctorado en
Tecnologías de la Información y la Comunicación

Tesis Doctoral

Técnicas de códigos de borrado aplicadas a almacenamiento distribuido

Realizada por:
Fabricio Rolando Marcillo Vera

Dirigida por:
Antonio Francisco Díaz García

La memoria titulada **"Técnicas de códigos de borrado aplicadas a almacenamiento distribuido"**, que presenta D. Fabricio Rolando Marcillo Vera para optar al grado de Doctor en Tecnologías de la Información y la Comunicación, ha sido realizada en el Departamento de Ingeniería de Computadores, Automática y Robótica de la Universidad de Granada, bajo la dirección del Doctor D. Antonio Francisco Díaz García.

Granada, 2023

Fabricio Rolando Marcillo Vera

Dr. D. Antonio Francisco Díaz García

-A Samanta Natasha, Alan Joao y Doménica Guadalupe-

Índice general

Índice de figuras

Índice de tablas

Agradecimiento

Dedico mi tesis principalmente a Dios, por darme la fuerza necesaria para culminar esta meta.

A Dña. Cely María Vera Rodríguez, mi madre, por todo su amor y por motivarme a seguir hacia adelante.

A mi compañera de vida, Dra. Guadalupe Vélez, gracias por entender sin preguntar, por apoyarme sin pedírtelo y por quererme como soy.

También a mis hijos Samanta Natasha, Alan Joao y Doménica Guadalupe, por brindarme su apoyo moral y motivación.

A mis hermanos, Líder Alexander, Santiago Fernando, Diego Rubén y Brandon Adolfo, por todo su apoyo incondicional, espero les sirva de ejemplo de que todo se puede lograr.

A mis amigos, Dr. Raúl Hernández, Dr. Wilmar Hernández y Dr. Amilkar Puris Cáceres, por la ayuda técnica, aporte de conocimientos y confiaza brindada.

En especial, quiero agradecer a mi Director de Tesis Dr. Antonio Francisco Díaz García por la confianza que depositó en mí, su constante apoyo, sus indicaciones y orientaciones indispensables en el desarrollo de esta tesis. Quisiera destacar la buena actitud que lo caracteriza.

Resumen

Los sistemas de almacenamiento distribuido tienen sus orígenes en los sistemas de almacenamiento centralizado, que eran utilizados para almacenar grandes cantidades de datos en un solo lugar. Sin embargo, estos sistemas presentaban problemas de escalabilidad y de disponibilidad de datos en caso de fallos, ya que una sola máquina centralizada era responsable de todo el almacenamiento y procesamiento de datos.

Con la llegada de internet y el aumento de la cantidad de datos generados por los usuarios, se hizo evidente la necesidad de sistemas de almacenamiento distribuido que pudieran escalar fácilmente y manejar grandes volúmenes de datos de manera confiable y eficiente.

Uno de los primeros sistemas de almacenamiento distribuido fue el sistema de archivos distribuido Distributed File System (DFS) desarrollado por Sun Microsystems en la década de 1980. Este sistema admitía que los datos se almacenaran en varios servidores y se distribuyeran automáticamente entre ellos, lo que mejoraba la disponibilidad de datos y la escalabilidad del sistema.

Otro sistema de almacenamiento distribuido notable es el sistema de almacenamiento de objetos Object Storage System (OSS), que fue desarrollado en la década de 1990 por empresas como EMC e Hitachi. En este sistema, los datos se dividen en objetos y se almacenan en múltiples servidores, lo que mejora la escalabilidad y la disponibilidad de los datos.

Con la popularidad de la nube y la computación en la nube, los sistemas

de almacenamiento distribuido se han vuelto aún más importantes y se han desarrollado nuevas tecnologías como el sistema de archivos distribuidos de Google (GFS) Google File System, y el sistema de archivos distribuidos de Hadoop (HDFS) Hadoop Distributed File System. Es decir, los sistemas de almacenamiento distribuido surgieron como respuesta a la necesidad de manejar grandes cantidades de datos de manera eficiente y confiable. Desde entonces, han evolucionado y se han vuelto cada vez más importantes en la era de la nube y el Big Data.

El objetivo principal de esta tesis es estudiar, analizar e implementar algoritmos de código de borrado para permitir resiliencia en los sistemas de almacenamiento distribuido. Para conseguir este objetivo inicialmente se ha realizado la revisión del estado de la bibliografía actual de la implementación de códigos de borrado, posterior la puesta en marcha de diferentes técnicas como: el algoritmo Reed-Solomon de la biblioteca Backblaze, el algoritmo Reed-Solomon de la biblioteca zfec y, finalmente, los algoritmos liberasure code rs vand, jerasure code rs vand, jerasure code rs cauchy, flat xor hd 3, flat xor hd 4, isa l rs vand, isa l rs cauchy de la biblioteca PyECLib.

El resultado de esta puesta en marcha nos ha permitido realizar diferentes desarrollos. Primeramente, un plugin nativo de almacenamiento distribuido para Dicoogle PACS, Dicoogle es un sistema de almacenamiento y recuperación de imágenes médicas basado en estándares abiertos. Posteriormente una batería de pruebas para la recuperación de ficheros de imágenes médicas dermatológicas con compresión con pérdida (JPG) y ficheros con compresión sin pérdida (PNG), esta batería es de gran importancia ya que permite documentar los ficheros del estado de la piel del paciente y su evolución a lo largo del tiempo, lo que resulta fundamental para la toma de decisiones clínicas. Por último, una capa de almacenamiento distribuida genérica, es decir soporta cualquier tipo de ficheros, esta implementada en un sistema operativo Open Source, con características de compresión sin pérdida, resiliencia, integridad y perpetuidad de los ficheros.

Capítulo 1

Introducción

En la actualidad, el almacenamiento de datos es uno de los aspectos más importantes de la informática. La cantidad de información que manejamos cada día es enorme y, por tanto, es necesario contar con tecnologías que permitan almacenarla de manera segura y confiable. En este sentido, los códigos de borrado Erasure Coding (EC) y la matriz redundante de discos independientes (RAID) son dos técnicas ampliamente utilizadas para proteger los datos.

Los códigos de borrado permiten recuperar información en caso de pérdida de algunos datos. Estos códigos se basan en la redundancia de información, es decir, se añaden datos extra para poder recuperar la información original en caso de que se produzcan fallos en el almacenamiento. En esta tesis, exploraremos cómo funcionan los códigos de borrado y cómo se aplican en la práctica.

Por otro lado, la matriz redundante de discos independientes son una técnica para almacenar datos en múltiples discos para mejorar la confiabilidad y el rendimiento. Los RAID utilizan diferentes configuraciones para distribuir y replicar datos entre los discos, de manera que, en caso de fallo de uno de ellos, se pueda acceder a los datos desde otros discos sin perder información. También, profundizaré en los diferentes tipos de arreglos

redundantes de matrices RAID y su aplicación en distintos contextos.

Los códigos de borrado son una posibilidad eficiente a RAID, ya que requieren menos espacio de almacenamiento y ofrecen un rendimiento similar. También son más flexibles y escalables, lo que significa que pueden adaptarse mejor a diferentes entornos y requisitos de almacenamiento [71] [169].

Otros sistemas de almacenamiento en red (SAN) Storage Area Network, son una tecnología de almacenamiento de datos que permite que múltiples servidores accedan a un conjunto de dispositivos de almacenamiento compartido a través de una red de alta velocidad. A pesar de que los (SAN) ofrecen muchas ventajas, también presentan algunos desafíos y problemáticas, entre ellos se encuentran: costo, complejidad, fallos de hardware, problemas de rendimiento, seguridad y confiabilidad. Los sistemas SAN son una tecnología valiosa para el almacenamiento y gestión de datos en entornos empresariales, pero también presentan desafíos y problemáticas que deben ser considerados antes de su implementación y durante su gestión [78].

Una alternativa es la codificación del borrado de datos EC (Erasure Coding), que es claramente diferente de otros sistemas basados en hardware. EC es una implementación basada en algoritmos que no está vinculada a ningún hardware específico. Rompe los datos en fragmentos, los expande y los codifica con piezas de información redundantes, y luego distribuye fragmentos codificados entre discos, nodos de almacenamiento o distintas ubicaciones. Con la codificación de borrado, los datos que se vuelven ilegibles en un nodo aún se pueden reconstruir utilizando información sobre los datos almacenados en otro lugar [13].

A diferencia de RAID, la codificación de borrado de datos no requiere un controlador de hardware especializado y proporciona una mejor capacidad de recuperación. Además, proporciona protección durante los procesos de recuperación. Dependiendo del grado de resiliencia, la recuperación

completa es incluso posible cuando solo la mitad de los elementos de datos están disponibles; esa es una gran ventaja sobre el RAID. En comparación con el RAID en espejo, la codificación de borrado de datos también consume menos almacenamiento. El único inconveniente es que la codificación de borrado de datos consume mucha CPU (Central Processing Unit) y puede causar problemas de latencia [166] [170] [171] .

La codificación de borrado de datos se representa con mayor frecuencia utilizando los códigos RS (Reed-Solomon). Para aquellos familiarizados con los códigos RS, dos parámetros de rendimiento importantes son: eficiencia de almacenamiento y tolerancia a fallos. La codificación de borrado de datos implica una compensación entre los dos. La eficiencia de almacenamiento es un indicador de almacenamiento adicional requerido para asegurar la resiliencia, mientras que la tolerancia a fallos es un indicador de la posible recuperación en el caso de fallo de los elementos [181].

Estas métricas son inversamente proporcionales entre sí; más tolerancia a fallos reduce la eficiencia de almacenamiento. Es decir, cuanto más distribuidos y, por lo tanto, geográficamente más generalizados se almacenan los datos, mayor será la latencia, ya que el tiempo requerido para recuperar desde diferentes ubicaciones o sistemas es mayor.

Los centros de datos (datacenter) de hiperescala plantean nuevos desafíos para la resiliencia de los datos en términos de fallos de nodo y lecturas degradadas. Los algoritmos de códigos de borrado de datos modernos han evolucionado para incluir códigos LRC (Local Reconstruction Codes), códigos con disponibilidad, códigos con recuperación secuencial, códigos MSR (Minimum Storage Regenerating), códigos de recuperaciones seleccionables y otros que son altamente personalizados [13] [62].

1.1. Objetivos

El objetivo principal de esta tesis es estudiar, analizar e implementar algoritmos de código de borrado para permitir resiliencia en los sistemas de almacenamiento distribuido. Para ello se considera como base los sistemas actuales que serán estudiados para evaluar una mejora utilizando técnicas basadas en dichos códigos. Los objetivos específicos son:

- Revisar el estado de la bibliografía actual de implementación de códigos de borrado. Dicha revisión se centró en implementaciones que se han desarrollado en los sistemas de almacenamiento existentes de los últimos años.

- Analizar las propuestas y diseñar una capa que facilite el intercambio de datos entre nodos donde sea viable implementar los algoritmos de código de borrado de la literatura para medir las prestaciones de tal manera que sea posible visualizar el rendimiento/penalización que ofrecen estos algoritmos. Para ello, se hizo uso de las bibliotecas de código de borrado de la literatura.

- Implementar algoritmos de código de borrado en un entorno real de almacenamiento, teniendo en cuenta las prestaciones que permitan visualizar el rendimiento/penalización de los algoritmos sobre los sistemas de almacenamiento de estudio.

- Diseñar una batería de pruebas para la validación de los algoritmos de código de borrado implementados en los sistemas de almacenamiento de estudio. Para ello, se determinó medidas de prestaciones para la evaluación global de los algoritmos. De acuerdo con los resultados obtenidos, éstos fueron interpretados y se realizó una comparación con los algoritmos tratados en esta tesis y que se han implementado en otros sistemas u otros entornos de almacenamiento.

Para lograr estos objetivos se ha utilizado la infraestructura del Departamento de Ingeniería de Computadores, Automática y Robótica. En particular, equipos HPC (High Performance Computing), que cuenta con las funcionalidades tanto de hardware y software para poder evaluar las prestaciones globales que se plantean en el trabajo de tesis.

1.2. Estructura de la tesis

Esta memoria está organizada en seis capítulos, los cuales se describen a continuación.

En el Capítulo 2 se presenta los antecedentes del almacenamiento distribuido, con una breve descripción de la codificación de borrado de datos, que opciones tenemos y tipos. Además, se relatan de manera metódica las diferentes implementaciones de RAID. Luego se describen los códigos en la práctica con énfasis en los códigos de recuperación local, códigos de máxima distancia separable y códigos regeneradores.

El Capítulo 3 se describen aspectos relevantes de los desafíos emergentes de resiliencia de datos, se considera la capacidad de la información, sus límites y una descripción del almacenamiento líquido. También, se describe la evolución de los sistemas de almacenamiento distribuido, el almacenamiento en la nube, la arquitectura del almacenamiento en la nube, Big Data en la nube y una perspectiva del almacenamiento distribuido con códigos de borrado.

El Capítulo 4 describe un estudio en el que se evalua el algoritmo de códigos de borrado Reed-Solomon (RS) como una alternativa al almacenamiento de datos redundantes en sistemas Picture Archiving Communication System (PACS). Se utilizó una implementación de códigos de borrado llamada Locally Repairable Codes (LRC), y se comparó su rendimiento con el de la replicación de datos. Los resultados mostraron que los códigos de borrado LRC ofrecían una reducción significativa en

el almacenamiento necesario, con un rendimiento comparable al de la replicación de datos en términos de tiempo de acceso a los datos.

El Capítulo 5 describe un estudio sobre la cuasiperiodicidad de los componentes principales en la descomposición en PCs de imágenes médicas, en particular, para el patrón maculopapular de COVID-19. Se comprimió una imagen médica de diferentes formas y se implementó una batería de pruebas utilizando códigos de borrado Reed-Solomon (RS) para evitar falsos positivos o negativos en el diagnóstico de la enfermedad. Los resultados demostraron que es posible trabajar con reconstrucciones aceptables de imágenes comprimidas en el campo de la dermatología sin perder la calidad y características que permiten llegar a un diagnóstico correcto, y los tiempos de procesamiento son óptimos.

El Capítulo 6 se describe la implementación de una capa de almacenamiento distribuida resiliente, con el objetivo de mostrar el rendimiento de los algoritmos de compresión sin pérdida y las técnicas de codificación y decodificación de diferentes códigos de borrado de la biblioteca PyECLib de Python. Además, se utilizó servicios de AWS, como DynamoDB y Amazon Quantum Ledger Database, junto con Amazon Simple Storage Service S3, para obtener tolerancia a fallos y alta durabilidad de los fragmentos. Los resultados obtenidos muestran tiempos de procesamiento favorables que pueden ser aplicados a cualquier plataforma de sistema de almacenamiento distribuido de código abierto.

En última instancia, se presentan las conclusiones y contribuciones clave de la tesis, así como se mencionan las posibles áreas de desarrollo futuro para mejorar la adaptabilidad de los códigos de borrado en entornos de almacenamiento y recuperación de datos cada vez más complejos y distribuidos.

Capítulo 2

Antecedentes del almacenamiento distribuido

En los sistemas de almacenamiento distribuido, es importante contar con mecanismos eficientes de distribución de datos entre los nodos de almacenamiento para garantizar la disponibilidad de los datos a las aplicaciones rápidas y eficientes. Los sistemas de almacenamiento distribuido cuentan con mecanismos específicos para reconocer los fallos de los discos o de los nodos de almacenamiento y ejecutar las acciones de recuperación especificadas en los protocolos de red, de forma que las aplicaciones puedan ejecutarse de forma eficiente (por ejemplo, un pequeño desequilibrio de carga) [51].

El tema tratado en esta sección es el estudio de los mecanismos en los sistemas que requieren un proceso de replicación de datos. Se trata de un proceso de replicación de datos eficiente que puede penalizar parte del rendimiento global de los sistemas, pero es un mecanismo eficiente para garantizar la disponibilidad de los datos en los sistemas.

La replicación es extremadamente útil para mantener copias de aplicaciones y datos, pero puede ser un problema para sistemas grandes

con muchas peticiones. Este reto se centra en la necesidad de aumentar la replicación manteniendo una alta resistencia.

Por ejemplo, el sistema de archivos HDFS (Hadoop Data File System) por defecto crea datos y metadatos de forma redundante y se crean almacenando tres copias de cada bloque de datos. Sin embargo, este esquema de duplicación proporciona una redundancia mínima en el almacenamiento del sistema de archivos. Además, el alcance es costoso (duplica fraccionadamente los datos y los metadatos) en comparación con soluciones de redundancia alternativas (por ejemplo, el hashing lineal). El punto es que se deben considerar mejores formas de preservar los datos [114] [186].

Sin embargo, en los sistemas RAID, las operaciones de paridad deben ser fiables y suelen imponer una cierta complejidad redundante. Estos sistemas son menos naturales que una replicación completa. El código de borrado (EC) proporciona redundancia sin una replicación estricta [166].

Desde 1960 se ha utilizado códigos de borrado (véase Fig:2.1), los cuales ganaron interés combinando con otras alternativas, los códigos FEC se han utilizado en los últimos años en las criptmonedas, el objetivo del desarrollo de estos códigos son reducir el espacio en bytes utilizado en las unidades de disco y mejorar la redundancia de datos [182] [156].

Siempre que el código tenga corrección de errores y la tasa de codificación no sea demasiado alta (superior al 35 %), la codificación Reed-Solomon es uno de los códigos más antiguos y más utilizados. Los códigos se han implementado para cifrar y autentificar archivos, proteger datos en medios de almacenamiento como disquetes, grabadoras de DVD (Digital Video Disc) y CD-ROM (Compact Disc Read Only Memory). El cifrado y la autentificación siguen el concepto de proteger los datos contra el robo presentando un mensaje que no puede ser entendido sin una clave secreta.

Katina Kralevska aplica en su investigación la construcción e implementación de nuevos códigos de borrado para sistemas de almacenamiento

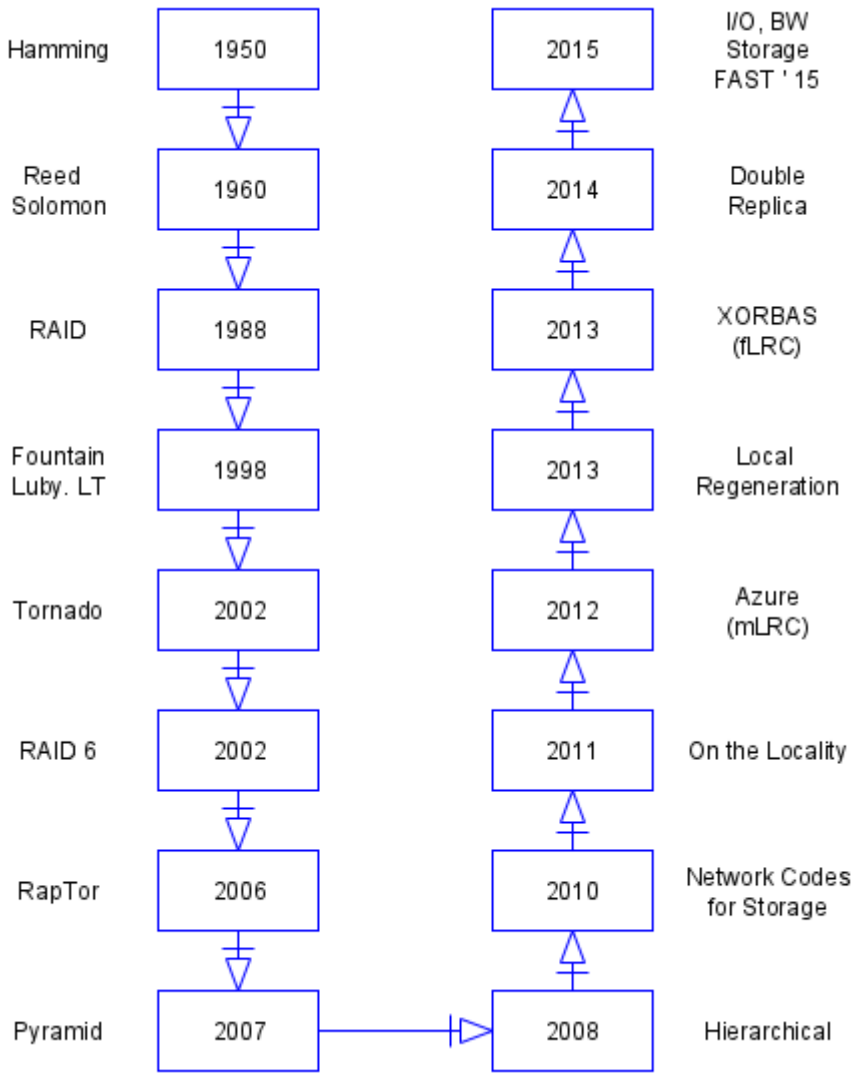


Figura 2.1: Línea de tiempo del código de borrado [156].

distribuido a gran escala que proporcionan ahorros en los recursos de almacenamiento y de red [108].

La idea central de este trabajo fue el estudio y análisis de los algoritmos de codificación de borrado en los sistemas de almacenamiento distribuido, concretamente, este trabajo aborda el estudio de los sistemas de almacenamiento open source.

2.1. Qué es la codificación de borrado de datos, cuáles opciones hay y tipos

La codificación de borrado de datos es una alternativa al RAID ya que puede aumentar la protección de los datos [166].

La codificación de borrado, también conocida como EC [169], es uno de los métodos más populares para la protección de datos. Su eficacia depende de la cantidad de espacio disponible en el disco; por lo tanto, a medida que hay más espacio disponible en el disco, hay más espacio disponible para los datos.

Una estrategia de codificación de borrado puede ser una solución muy flexible para los administradores de almacenamiento. En concreto, puede ayudarles a encontrar el equilibrio adecuado entre el tamaño de la unidad y la inserción simultánea de fallos. En definitiva, pueden decidir entre la rapidez con la que deben recuperarse los datos y el número de fallos que pueden producirse simultáneamente en los sistemas.

Una mezcla de ataques físicos y amenazas internas ha llevado a los líderes del sector a hacer hincapié en una solución que equilibre la seguridad y la asequibilidad. Mientras que las capas de seguridad de datos basadas en hardware proporcionan diversos niveles de protección y estimación del tiempo de recuperación, los costes y las dificultades de escalabilidad hacen que la mayoría de los esfuerzos sean fáciles de eludir, especialmente con las soluciones de almacenamiento de estado sólido PCIe virtual [124] [1].

Las soluciones tradicionales de almacenamiento de paridad RAID dirigidas a las SAN, no logran funcionar adecuadamente en este nuevo

entorno de datos intensivos. Aunque los métodos de partición de datos y colocación de paridad basados en software proporcionan diversos grados de protección, la sobrecarga y el tiempo de recuperación que conlleva el proceso de reparación son generalmente inadecuados [78] [170] .

Más adelante se describe con más detalle los ejemplos, pero, brevemente, estas técnicas se diferencian fundamentalmente de las técnicas RAID, como el striping y el mirroring. Con estas técnicas, se crea una sincronización poco precisa entre dos o más unidades para que parezcan casi una única unidad. Si una unidad falla, hay una copia de seguridad en una de las unidades restantes para mantener los datos. Con la codificación de borrado, un desarrollador hace intencionadamente que los datos sean imperfectos, lo que almacena la redundancia dentro de los datos para que valores específicos o ciertos registros se almacenen en múltiples lugares. Como resultado, si una sola validación falla, ya no es posible hacer una restauración porque toda la teoría de las restauraciones es incorrecta [170].

La codificación de borrado es un método de protección de datos que utiliza matemáticas avanzadas. Los métodos de protección de datos de almacenamiento permiten a los sistemas de archivos regenerar los datos que faltan utilizando fragmentos de datos conocidos llamados bloques de paridad. Los datos adicionales pueden derivarse utilizando esta combinación de datos realizados y de paridad. La combinación de dos bloques puede almacenarse al mismo tiempo que los propios datos, y puede sustituir a una copia en espejo en la mayoría de las circunstancias. A diferencia de la restauración de una copia en espejo, la recuperación con el método de codificación de borrado es exacta . Esta técnica es útil cuando sólo está afectado uno o un pequeño número de archivos de datos específicos, en lugar de todo el sistema de archivos [156].

Hoy en día, la codificación de borrado es un sistema que se explora con frecuencia para mejorar la eficiencia del almacenamiento de datos. La codificación de borrado (EC) rompe los datos en fragmentos y genera piezas

redundantes para cifrar esos fragmentos [1] [171].

Las piezas se distribuyen en varios nodos del almacenamiento. Cuando los datos se rompen, la información sólo se pierde localmente y ningún dato se pierde globalmente. En otras palabras, si se corrompe o compromete de alguna manera un nodo de datos o un dato insalvable dentro de un nodo que se recupera para enviarlo fragmentado a un segundo nodo, todos los demás nodos de datos y los datos son capaces de ser reparados. En una configuración RAID 1 a mirroring, como cada copia reside en un disco individual (frente a RAID 0 striping), los datos son efectivamente una copia de los mismos. Con el acceso de lectura, modificación y escritura a los datos, incluyendo la escritura de nuevos datos, si uno de los discos componentes del par de replicación falla, la réplica puede recuperar los datos. Una disposición RAID 1 es fácil de implementar, pero tiene algunas desventajas, por ejemplo, se necesita espacio para implementarlo. No es un concepto sin fisuras, ya que sólo permite que falle un disco a la vez [97].

Un controlador RAID es un dispositivo utilizado para gestionar los discos duros de una matriz de almacenamiento. Puede utilizarse como nivel de abstracción entre el sistema operativo y los discos físicos, presentando grupos de discos como unidades lógicas. El uso de una controladora RAID puede mejorar el rendimiento y ayudar a proteger los datos en caso de fallo [97].

Una controladora RAID puede estar basada en hardware o en software. En un producto RAID basado en hardware, un controlador físico gestiona toda la matriz. La controladora también puede estar diseñada para admitir formatos de unidad como SATA (Serial Advanced Technology Attachment) y SCSI (Small Computer System Interface). Una controladora RAID física también puede estar integrada en la placa base de un servidor [97].

Con el RAID basado en software, la controladora utiliza los recursos del sistema de hardware, como el procesador central y la memoria. Aunque realiza las mismas funciones que una controladora RAID basada en hardware, es posible que las controladoras RAID basadas en software no

permitan aumentar tanto el rendimiento y pueden afectar al rendimiento de otras aplicaciones del servidor [97].

Si una implementación RAID basada en software no es compatible con el proceso de arranque de un sistema y las controladoras RAID basadas en hardware son demasiado costosas, el firmware, o RAID basado en controladores, es una opción potencial.

Los chips controladores RAID basados en firmware se encuentran en la placa base y todas las operaciones las realiza la unidad central de proceso (CPU), de forma similar al RAID basado en software. Sin embargo, con el firmware, el sistema RAID sólo se implementa al principio del proceso de arranque. Una vez que el Sistema Operativo (OS) se ha cargado, el controlador se hace cargo de la funcionalidad RAID. Un controlador RAID de firmware no es tan caro como una opción de hardware, pero supone una mayor carga para la CPU del ordenador. El RAID basado en firmware también se denomina RAID de software asistido por hardware, RAID de modelo híbrido y RAID simulado [97].

La diferencia clave es que las características de RAID pueden integrarse en cualquier esquema de protección de datos (tipo RAID) y que la codificación de borrado de datos puede lograrse con una programación sencilla. La codificación de borrado de datos también funciona en sistemas con pérdidas. Además de los retos de implementación que conlleva la pérdida, la codificación de borrado de datos también tiene la ventaja de no presentar tanta latencia en comparación con el RAID. Por otro lado, como los códigos de borrado requieren que la redundancia sea decodificada, tienen la sobrecarga inherente de la CPU [108] [147].

RAID 5 es una configuración de varios discos duros que ayuda a mejorar la eficiencia y el rendimiento de lectura. Sin embargo, el RAID 5 es una configuración avanzada que sólo suele utilizar el personal informático. Una configuración estándar de RAID con 5 discos puede ser gestionada y mantenida por el departamento de IT (Information Technology). Y como

el RAID 5 suele utilizar un gran número de discos que hay que sustituir en caso de fallo, sólo se pueden almacenar conjuntos de datos pequeños en una configuración RAID de este tipo [97].

El RAID existe desde hace mucho tiempo. La configuración básica y mas común de protección de datos es el nivel 1 de RAID, también llamado *mirroring*. Como su nombre indica, la duplicación significa que se crean copias alternativas e idénticas. A diferencia de una copia de seguridad convencional, un trabajo de duplicación es rápido y consiste en copiar los datos en un punto alternativo, por lo que se pueden crear más copias muy rápidamente para mantener al menos dos versiones de los archivos informáticos más importantes [166] [97] .

El evaluar y seleccionar la tecnología RAID de reparación autónoma de segmentos (RAID) , realmente no puede hacerlo todo. La tecnología RAID proporciona una sólida protección de datos, pero reduce drásticamente el rendimiento. Por ello, al determinar las configuraciones RAID para la solución de almacenamiento de copias de seguridad de datos, los administradores de IT (Information Technology) deben elegir entre una sólida protección de datos, el rendimiento y una mayor eficiencia del almacenamiento. Las copias de seguridad, los sistemas RAID y las redes de datos son tecnologías arcaicas para todos aquellos que quieren una protección de datos más sólida, esto se debe a que actualmente están siendo sustituidos por códigos de borrado, y estos pueden proporcionar un almacenamiento de datos eficiente miles de veces al igual que en los mencionados anteriormente pero a diferencia de aquellos, que soportan una cantidad moderada de datos pero son más propensos a los fallos, aunque hay corrección de errores incorporada en el sistema de almacenamiento de los dispositivos relacionales o de red, principalmente cuando el dispositivo experimenta un fallo de hardware, los códigos de borrado abordaron una mayor tolerancia a eso [184].

Ante el hecho de que las capacidades de las unidades de disco HDD

(Hard Disk Drive) se han multiplicado por seis en los últimos años, hasta alcanzar más de 6 Terabytes TB, el sistema RAID, actualmente el estándar de facto para la protección de datos, debe ser revisado, ya que el conjunto de unidades SSD (Solid State Drive) altamente transparentes hace que las soluciones de matriz sean cada vez menos atractivas para este caso de uso [14].

Ha surgido una solución al problema del almacenamiento de datos conectados como método de protección contra la pérdida de una unidad. Los sistemas RAID modernos no son suficientes en esta época de discos duros de gran capacidad. Cuanto mayor es la capacidad de un disco, mayores son las probabilidades de que uno de ellos falle [170].

Cuando un disco falla, comienza el proceso de reconstrucción del RAID. Durante este proceso, el conjunto ya no puede estar seguro contra un segundo (o tercer) fallo de disco. Como resultado, los riesgos de un funcionamiento normal mientras se reconstruye han crecido constantemente y desde entonces se ha convertido en un riesgo mucho mayor [166].

Antes, los RAIDs grandes, con cinco discos de 1 Terabyte TB en RAID 5 [97] en un servidor de 2000 dólares americanos , se medían en minutos u horas. Sin embargo, debido a que las velocidades de transferencia no han seguido el ritmo de expansión de la capacidad de los discos, estos grandes RAIDs pueden tardar días o más cuando los datos se copian al nuevo RAID.

En consecuencia, muchos sostienen que ahora se necesitan alternativas al RAID [166], por lo que es importante que nos centremos en la codificación de borrado.

2.2. Casos de uso

La alta carga de la CPU [147] y la latencia de la codificación de borrado la hacen muy adecuada para el archivado en bases de datos, debido a la naturaleza a largo plazo del almacenamiento. Durante largos periodos, cabe

esperar que se degraden más elementos de almacenamiento. Sin embargo, es muy adecuado para personas con grandes conjuntos de datos y un número correspondientemente grande de elementos de almacenamiento.

Además, algunos algoritmos de borrado seguro son más intensivos en términos de uso de CPU que otros. Por ejemplo, el algoritmo de Gutmann, que sobrescribe los datos existentes con patrones complejos de bits, es conocido por ser muy intensivo en términos de uso de CPU y puede tardar mucho tiempo en completarse, especialmente en dispositivos de almacenamiento grandes. Por lo tanto, es importante elegir cuidadosamente el algoritmo de borrado seguro que se va a utilizar, teniendo en cuenta el tamaño y la velocidad del dispositivo de almacenamiento, así como la cantidad de datos que se van a eliminar.

Siguiendo con el tema, PowerScale OneFS Sistema operativo para NAS (Network Attached Storage) de escalamiento horizontal utiliza la redundancia de datos en todo el clúster para evitar la pérdida de datos resultante de fallos de unidades o nodos. La protección está integrada en la estructura del sistema de archivos y puede aplicarse hasta el nivel de archivos individuales [161].

La codificación de borrado también se utiliza en el contexto del almacenamiento de objetos. En este escenario, los operadores de almacenamiento por lo general, proveedores de nube de gran volumen son ahora los usuarios más probables [42].

En contexto, el Sistema Operativo Dell EMC Isilon OneFS se basa en el algoritmo Reed-Solomon, que utiliza la corrección de errores hacia delante (FEC). Mediante FEC, OneFS asigna los datos en fragmentos de 128 KB. Por cada n trozos de datos, OneFS escribe m trozos de protección, o paridad. Cada trozo $n + m$, denominado grupo de protección, se escribe en un disco independiente de un nodo independiente. Este proceso se conoce como *striping* de datos. Al dividir los datos en segmentos en todo el clúster, OneFS puede recuperar archivos en caso de que fallen los discos o los nodos

[161] [137] [182].

En OneFS, los conceptos de política de protección y nivel de protección son diferentes. La política de protección es la configuración de protección que se especifica para los grupos de almacenamiento del clúster. El nivel de protección es la protección real que OneFS consigue para los datos, basándose en la política de protección y en el número real de nodos con capacidad de escritura [161].

MinIO es un servidor de almacenamiento en la nube compatible con Amazon S3, protege los datos con una codificación de borrado en línea por objeto que está escrita en código ensamblador para ofrecer el mayor rendimiento posible. MinIO hace uso de las instrucciones Intel AVX512 para aprovechar al máximo los recursos de la CPU host en múltiples nodos para una rápida codificación de borrado. Una CPU estándar, unidades NVMe rápidas y una red de 100 Gbps permiten escribir objetos codificados por borrado a una velocidad casi de cable [151].

MinIO utiliza el código Reed-Solomon para dividir los objetos en bloques de datos y paridad que pueden configurarse con el nivel de redundancia que se desee. Esto significa que en una configuración de 16 unidades con 8 de paridad, un objeto se divide en 8 bloques de datos y 8 de paridad. Incluso si se pierden hasta 7 ($\frac{n}{2} - 1$) unidades, ya sean de paridad o de datos, se pueden reconstruir los datos de forma fiable a partir de las unidades restantes. La implementación de MinIO garantiza la lectura de objetos o la escritura de nuevos objetos incluso en caso de pérdida o indisponibilidad de varios dispositivos [151] [137].

Además, MinIO divide los objetos en bloques de datos y de paridad en función del tamaño del conjunto de borrado y, a continuación, distribuye de forma aleatoria y uniforme los bloques de datos y de paridad entre las unidades de un conjunto tal que cada unidad no contenga más de un bloque por objeto. Aunque una unidad puede contener bloques de datos y de paridad para varios objetos, un solo objeto no tiene más de un bloque por

unidad, siempre que haya un número suficiente de unidades en el sistema. Para los objetos versionados, MinIO selecciona las mismas unidades para el almacenamiento de datos y paridad, manteniendo un solapamiento cero en cualquier unidad [151].

Backblaze Inc, es una empresa estadounidense de respaldo de datos y almacenamiento en la nube, Backblaze Vaults utiliza una biblioteca de codificación de borrado Reed-Solomon [137] para calcular la paridad y luego utilizarla para reconstruir archivos. Cuando un archivo se almacena en un dispositivo, se divide en 17 partes, todas del mismo tamaño. A continuación, se crean tres fragmentos adicionales que contienen paridad, lo que da un total de 20 fragmentos. El archivo original puede reconstruirse a partir de 17 de los 20 fragmentos [18].

Muchos sistemas de bases de datos comunes que garantizan una alta concurrencia y escalabilidad utilizan una arquitectura de separación de lectura y escritura. Al mismo tiempo, estos sistemas necesitan almacenar una gran cantidad de datos todos los días, lo que requiere diferentes mecanismos de acceso y almacenamiento de datos, como estrategias de acceso a datos fríos y calientes [229].

CBase EC utiliza códigos de borrado para intercambiar características de procesamiento de transacciones y eficiencia de almacenamiento desarrollados por CBase para escenarios financieros y utilizados en el sistema de base de datos bancario. El primer algoritmo utilizado es el reconocimiento de segmento caliente-frío, y el segundo es un algoritmo de conversión dinámica caliente-frío. Se adaptan dos métodos de optimización para mejorar el rendimiento de CBase EC. La sección experimental compara CBase EC con tres réplicas de CBase. Los resultados muestran que, si bien el rendimiento del procesamiento de transacciones no disminuye en más del 6 %, la eficiencia del almacenamiento aumenta en un 18,4 % [229].

Un sistema de almacenamiento por niveles utiliza el método de replicación para proporcionar alta fiabilidad y disponibilidad, que almacena

tres réplicas en diferentes nodos de los clústeres. Los códigos de borrado (EC), como el Reed-Solomon (RS), se utilizan cada vez más para reducir aún más la sobrecarga de almacenamiento, al tiempo que proporcionan un bajo rendimiento de E/S y disponibilidad [93].

WarmCache, es una capa de software para datos en caliente que tiene una copia almacenada mediante codificación de borrado y la otra copia en la capa de datos en memoria. El uso de una copia en la capa de datos de codificación de borrado garantiza la fiabilidad de los datos, mientras que la otra copia en la capa de datos de memoria proporciona un rápido rendimiento de E/S [93].

Los códigos de borrado Reed-Solomon (RS) se utilizan ampliamente en los sistemas de comunicación y almacenamiento de paquetes para recuperarse del borrado. Cuando se implementa un decodificador en una FPGA (Field Programable Gate Array) en una plataforma espacial, sufre una interferencia de evento único (SEU), que puede causar fallas de funcionamiento. Los autores proponen un esquema de detección y localización de errores basado en la recodificación parcial de errores en la memoria de usuario del decodificador [67].

Así, la estabilización de la copia puede ahorrar espacio de almacenamiento en comparación con la duplicación y ofrece niveles más altos y configurables de protección contra fallos de hardware, es adecuada para el almacenamiento a muy gran escala, de archivo y vertical, pero, por el momento, es menos adecuada para las fuentes de datos de producción.

Aunque la codificación de borrado puede ser una medida efectiva para eliminar datos de forma segura y permanente, hay varias barreras que pueden dificultar o incluso impedir el proceso de borrado adecuado. Algunas de las barreras comunes a la codificación de borrado incluyen:

- Velocidad del dispositivo de almacenamiento: El proceso de borrado seguro puede ser muy intensivo en términos de uso de recursos, como la CPU y la memoria. Si el dispositivo de almacenamiento es lento o

tiene poca capacidad, puede llevar mucho tiempo completar el proceso de borrado, lo que puede ser un problema si se necesita borrar los datos de forma rápida y segura.

- Tamaño del dispositivo de almacenamiento: Cuanto más grande sea el dispositivo de almacenamiento, más tiempo y recursos se necesitarán para completar el proceso de borrado seguro. Esto puede ser especialmente problemático si se necesita borrar los datos de varios dispositivos de almacenamiento grandes.

- Acceso físico al dispositivo de almacenamiento: Para realizar un borrado seguro, se debe tener acceso físico al dispositivo de almacenamiento. Esto puede ser un problema si el dispositivo está ubicado en un lugar inaccesible o si no se tiene acceso físico al dispositivo, como en el caso de un servidor remoto.

- Algoritmos de borrado no compatibles: Hay varios algoritmos de borrado seguro disponibles, y algunos dispositivos de almacenamiento pueden requerir algoritmos específicos para un borrado seguro y completo. Si el algoritmo utilizado no es compatible con el dispositivo de almacenamiento, es posible que el proceso de borrado no se complete correctamente.

- Uso de técnicas de recuperación de datos: Aunque el borrado seguro puede ser efectivo para eliminar datos, todavía existen técnicas avanzadas de recuperación de datos que pueden recuperar información después del borrado. Estas técnicas pueden ser especialmente efectivas si el borrado no se realizó de manera adecuada o si se utilizaron algoritmos de borrado inadecuados.

Se puede utilizar matrices RAID con las copias de seguridad para mejorar el rendimiento. Este tipo de copia de seguridad suele combinar varios dispositivos de almacenamiento masivo para crear un único espacio

de almacenamiento de gran tamaño que puedes utilizar para hacer copias de seguridad de tus archivos críticos. Estas matrices RAID aunque muy útiles, no son 100 % a prueba de fallos. La copia de seguridad basada en disco aprovecha intrínsecamente el uso de matrices RAID para acelerar el proceso. Agilizan los procesos de copia de seguridad y restauración, al tiempo que ayudan a salvaguardar los datos [166] [1].

Puede mejorar enormemente el rendimiento de las copias de seguridad utilizando matrices RAID. Las matrices RAID, que combinan varias unidades de disco en una única matriz, se utilizan al realizar copias de seguridad para mejorar el rendimiento. Estas matrices lo consiguen haciendo que los archivos de gran tamaño estén disponibles como un único volumen físico, a la vez que gestionan los posibles fallos de disco para garantizar la protección de sus datos. Combinadas con soluciones de copia de seguridad basadas en discos de clase empresarial, las matrices RAID garantizan que sus datos no sean eliminados por un único fallo [166] [1].

RAID (Redundant Array of Independent Disks) es posible que no se haya oído hablar de el término antes porque no sólo se utiliza con discos duros, sino que a menudo se emplea también con otros dispositivos de almacenamiento. Hay muchas configuraciones RAID diferentes que están diseñadas para dos propósitos: mejorar el rendimiento y la redundancia de datos [166] [71].

2.3. RAID 0

Las matrices RAID 0 incluyen dos o más unidades de disco y proporcionan *striping* entre ellas. El lado positivo es que no es necesario hacer copias de seguridad de los datos, ya que son matrices redundantes, lo que significa que, aunque se almacenen en varios dispositivos, todos los datos se redistribuyen periódicamente entre ellos. El inconveniente de las matrices RAID 0 es que carecen de protección de datos [1] [36] [97].

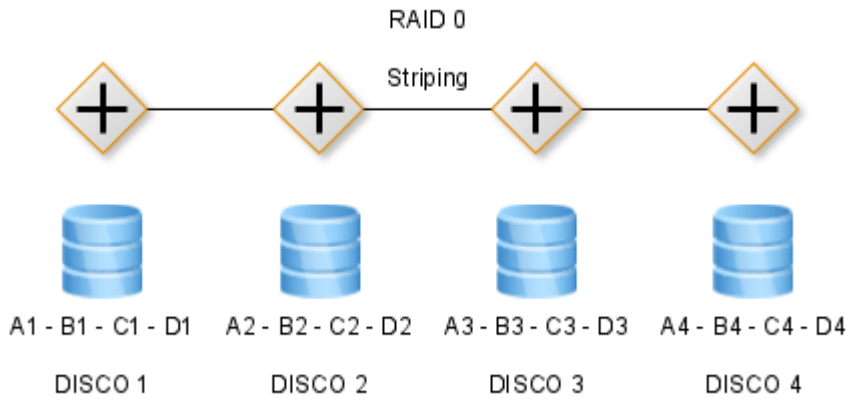


Figura 2.2: Grupo/matriz redundante de discos independientes - Nivel 0 Striping.

La principal ventaja del RAID 0 y de la división en segmentos de discos es la mejora del rendimiento (véase Fig:2.2). Por ejemplo, la separación de datos en tres discos duros triplicaría el ancho de banda de una sola unidad. Si cada unidad funciona a 200 operaciones de entrada/salida por segundo, la división en segmentos de discos permitiría disponer de hasta 600 IOPS para lecturas y escrituras de datos.

RAID 0 es una configuración de almacenamiento que se utiliza para mejorar el rendimiento y la velocidad de acceso a los datos mediante la distribución de los datos en dos o más discos [184].

La desventaja del striping de discos es su baja resiliencia. RAID 0 no utiliza redundancia de datos, por lo que el fallo de cualquier unidad física del conjunto de discos divididos en segmentos provoca la pérdida de los datos de la unidad dividida en segmentos y, en consecuencia, la pérdida de todo el conjunto de datos almacenados en el conjunto de discos duros divididos en segmentos. No debe utilizarse para almacenamiento de misión crítica [184].

2.4. RAID 1

Las matrices RAID 1 constan de dos discos, y se sabe que pueden utilizarse para duplicar la capacidad de almacenamiento de una sola unidad de disco a costa de hacerlo con la mitad de capacidad. Básicamente dobla la capacidad de almacenamiento, pero con una reducción del 50 % entonces realmente vale la pena, esa compensación tiene una justificación lógica. A medida que se han ido mejorando los límites de capacidad, han aumentado los tamaños de los archivos y las cargas de trabajo de las aplicaciones. Eso significa que los borrados de bloques tienen más probabilidades de provocar la pérdida de datos, de ahí que los esfuerzos se desvíen de los esquemas de el aprovisionamiento ligero (Thin Provisioning) hacia las matrices RAID [36] [97].

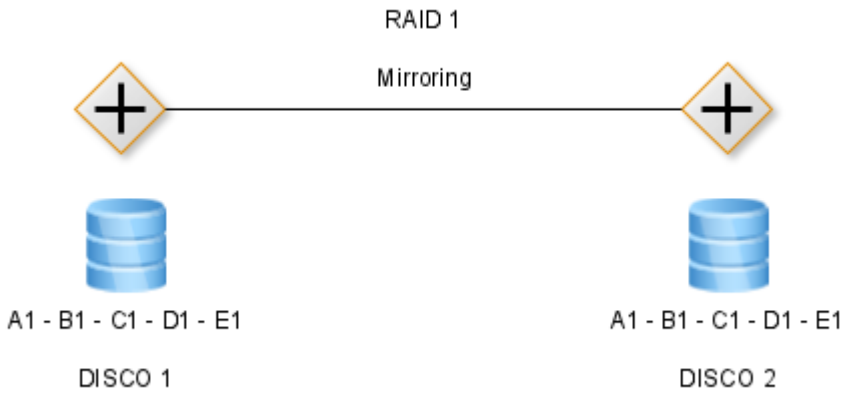


Figura 2.3: Grupo/matriz redundante de discos independientes - Nivel 1 Mirroring.

RAID 1 tiene la ventaja de proporcionar velocidades de lectura mejoradas y protección adicional de los discos duros si el controlador o el software de gestión permiten el acceso simultáneo a más de un medio de almacenamiento (véase Fig:2.3). En el primer caso, esto es posible

mediante el acceso paralelo a diferentes sectores, en el segundo mediante la comparación de los datos.

Su mayor fortaleza la redundancia completa es también su debilidad. Como todos los discos duros de la red deben almacenar los mismos datos, se pierde automáticamente mucha capacidad potencial de almacenamiento. A la inversa, esto significa que el almacenamiento RAID 1 es al menos el doble de costoso (cuando se combinan dos discos duros) que los soportes de datos individuales con la misma capacidad de almacenamiento. Comparado con otros niveles RAID que generan redundancia con la ayuda de la paridad, el factor de alto coste del RAID 1 es una desventaja.

Sin embargo, hay algunos otros métodos que podrían ser mejores, cada uno tiene un proposito.

2.5. RAID 10

Las matrices RAID 1+0 son iguales que las matrices RAID 10, sin embargo, todos los datos y la información de paridad deben repartirse por igual entre todos los dispositivos de la matriz, lo que esencialmente limita todos los dispositivos de la matriz al dispositivo más pequeño. La ventaja del RAID 1+0 es que proporciona tanto rendimiento como protección de datos. Por supuesto, la desventaja es que hay una cantidad muy limitada de datos almacenados en una sola unidad dentro de la matriz, lo que en sí mismo no es un problema, pero cuanto más pequeña es la unidad, menos puede almacenar [36] [97].

RAID 10 almacena todos los datos por duplicado. Mientras uno de los discos de un par espejo siga funcionando, la información almacenada está por tanto a salvo, aunque falle un soporte de datos (véase Fig:2.4). Los datos sólo se pierden si todos los soportes de almacenamiento de un RAID 1 subordinado fallan debido a un defecto u otro motivo. Esta es una de las principales ventajas de un sistema RAID 1+0 en comparación con RAID

0+1, en el que los daños no pueden asignarse a ningún disco específico de una sub-unidad RAID 0. Todo esto significa que reconstruir los datos en un sistema RAID 10 es más sencillo y rápido.

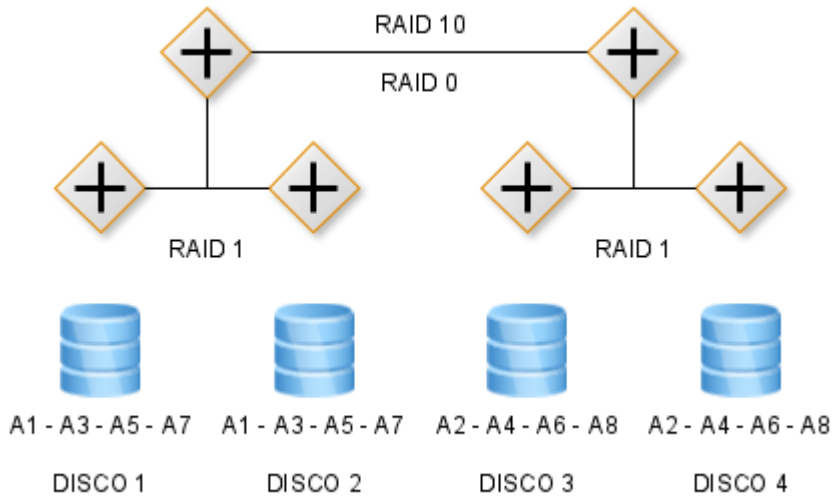


Figura 2.4: Grupo/matriz redundante de discos independientes - Nivel 1+0.

Uno de los grandes puntos fuertes del RAID 10 es la velocidad de salida del sistema. Al dividir los datos en segmentos, los sub-bloques individuales están disponibles en paralelo. Si una aplicación accede al sistema, puede leer de dos o más discos al mismo tiempo y siempre recibe varios datos de un segmento. En comparación con un único disco que no tiene esa opción, la velocidad de lectura en RAID 10 siempre está optimizada [184].

2.6. RAID 5

Una matriz RAID 5 requiere un mínimo de tres discos duros. Esta matriz también utiliza información de paridad y separación de datos. Esta

protección de datos y el rendimiento también promueven la necesidad de esta matriz [36] [97].

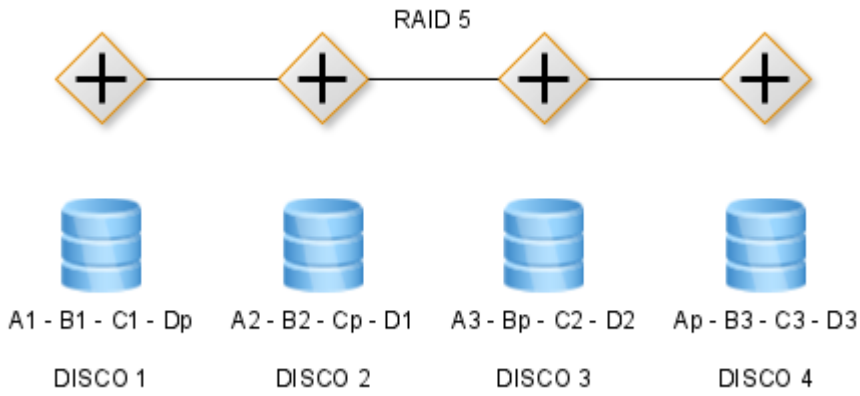


Figura 2.5: Grupo/matriz redundante de discos independientes - Nivel 5.

RAID 5 se caracteriza por una buena relación costo-beneficio gracias a su eficiente generación de redundancia. A diferencia de otros sistemas, los archivos no se guardan en varias versiones, sino sólo de forma redundante gracias a los bloques de paridad (véase Fig:2.5). En comparación con el uso de unidades individuales, la capacidad de almacenamiento disminuye, pero las unidades RAID 5 conservan una parte relativamente grande de la capacidad original. Por otro lado, RAID 5 es una solución rentable para aumentar la velocidad de lectura. La división en segmentos de los datos permite el acceso en paralelo a varias partes de un bloque de datos relacionado, de modo que los dispositivos de consulta pueden completar el proceso de lectura mucho más rápido.

Otro punto fuerte del RAID 5 es su mayor fiabilidad. Si un disco falla porque está defectuoso, o si los datos de un disco se pierden por otros motivos, las operaciones pueden seguir funcionando. Dado que el resto de medios de almacenamiento se utilizan más durante el proceso

de recuperación, el riesgo de fallo aumenta significativamente durante este periodo [184].

En el lado negativo, el inconveniente de la disposición de almacenamiento RAID 5 es que cuando falla la unidad más pequeña, el conjunto de discos pierde todos los datos, lo que puede tardar mucho tiempo en recuperarse.

Sin embargo, RAID 5 no está exento de desventajas. Cada proceso de escritura en el grupo de discos duros está conectado con un paso de lectura adicional para comprobar y recalcular la información de paridad disponible. Se necesita otro paso para distribuir los datos de paridad para los datos de usuario recién almacenados en los discos. En comparación con los discos individuales y otros niveles RAID como RAID 0, la velocidad de escritura de los soportes de datos en un sistema RAID 5 es mucho menor [184].

2.7. RAID 50

RAID 50, o 5+0, es un nivel RAID anidado. El RAID anidado combina dos técnicas básicas de RAID para aprovechar las ventajas de ambas. Todos los niveles RAID anidados incluyen RAID 0, o separación de discos. La división en segmentos reparte los datos entre varias unidades, lo que permite que las lecturas y escrituras se realicen con mayor rapidez. Por sí solo, el RAID 0 no tiene paridad [36] [97].

Al combinar la división en segmentos del RAID 0 con la paridad distribuida uniformemente del RAID 5, el RAID 50 proporciona al RAID 0 la paridad que necesita para tener redundancia y tolerancia a fallos. Distribuye los datos en al menos dos matrices de discos RAID 5 y requiere un mínimo de seis discos (véase Fig:2.6) [97].

RAID 50 es una configuración de almacenamiento que combina características de RAID 0 y RAID 5 para proporcionar un equilibrio entre rendimiento y seguridad de datos. RAID 50 ofrece un mayor rendimiento de escritura y una mejor protección de datos que RAID 5 en caso de fallo de

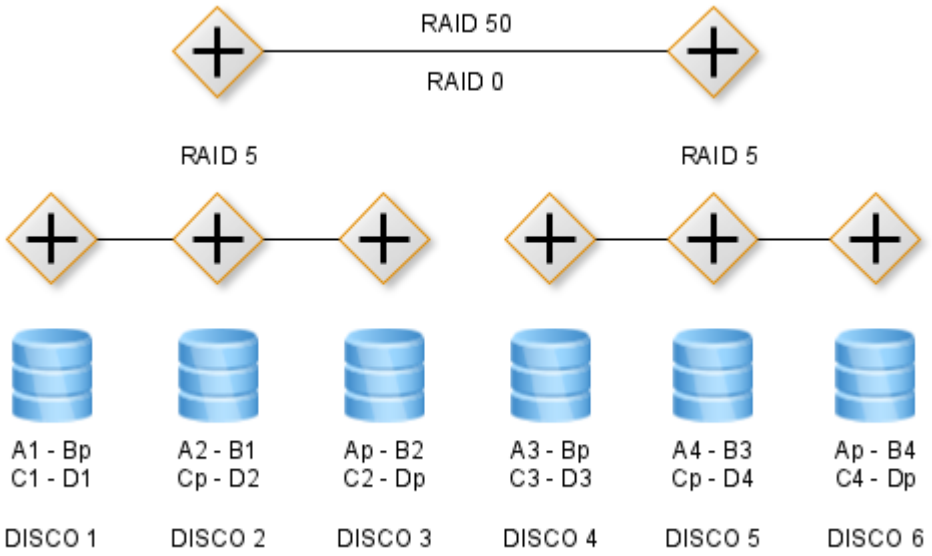


Figura 2.6: Grupo/matriz redundante de discos independientes - Nivel 5+0.

un disco. RAID 50 es capaz de realizar reconstrucciones más rápidas, una necesidad en un momento en el que el tiempo de inactividad se considera inaceptable [97].

Aunque el rendimiento se degrada inevitablemente en caso de fallo de un disco, no lo hace tanto como en una matriz RAID 5, ya que un único fallo sólo afecta a una de las matrices, dejando la otra totalmente funcional. De hecho, RAID 50 puede soportar hasta cuatro fallos de disco si cada disco que falla está en una matriz RAID 5 diferente [97].

RAID 50 se utiliza mejor para aplicaciones que necesitan una alta fiabilidad y que necesitan manejar altas tasas de peticiones y alta transferencia de datos con un menor coste de discos que una matriz RAID 10 (1+0, en espejo y en segmentos). Sin embargo, dado que se necesita un mínimo de seis discos para configurar una matriz RAID 50, el coste no se

elimina completamente como factor [97].

Una de las desventajas del RAID 50 es que, al igual que el RAID 5, necesita un controlador sofisticado. Para obtener el máximo rendimiento, una matriz RAID 50 debe tener discos sincronizados. Lamentablemente, esto limita las opciones de disco, porque no todos los discos pueden sincronizarse [184].

RAID 50 tiene menos espacio utilizable en disco que RAID 5, debido a que asigna un disco por matriz para la paridad. Sin embargo, sigue teniendo más espacio utilizable que otros niveles RAID, sobre todo que los que utilizan la duplicación [97].

Con un requisito mínimo de seis discos, RAID 50 puede ser una opción costosa, pero ese espacio adicional en disco justifica el gasto al proteger sus datos. No es infalible, y múltiples fallos de disco en las matrices RAID 5 subyacentes en la configuración son malas noticias. Una de las ventajas del RAID 5, sin embargo, es que una matriz RAID 5 puede soportar un solo fallo, por lo que las probabilidades de que ambas matrices dentro de la configuración 50 queden fuera de servicio al mismo tiempo son improbables [184].

2.8. RAID 6

RAID 6 es una configuración de almacenamiento que se utiliza para proporcionar una mayor protección de datos y redundancia en comparación con otras configuraciones como RAID 5 y RAID 0, RAID 6 utiliza menos almacenamiento que, por ejemplo, una matriz RAID 10, que sólo puede almacenar la mitad de su capacidad total de almacenamiento en datos, ya que la otra mitad la utiliza la duplicación [36] [97] [170].

Si una matriz RAID 6 contiene el número mínimo de discos (cuatro), también sólo puede almacenar la mitad de la capacidad total en datos, ya que RAID 6 reserva la capacidad de dos discos para la paridad. La diferencia

viene a medida que se añaden discos [170].

El porcentaje de capacidad utilizable aumenta a medida que se añaden discos a una matriz RAID 6. Si se utilizan ocho discos en RAID 6, por ejemplo, la paridad, una técnica que comprueba si los datos se han perdido o se han sobrescrito cuando se mueven de un lugar a otro del almacenamiento o cuando se transmiten entre ordenadores, sólo consume el 25 % de la capacidad del disco (véase Fig:2.7) [170].

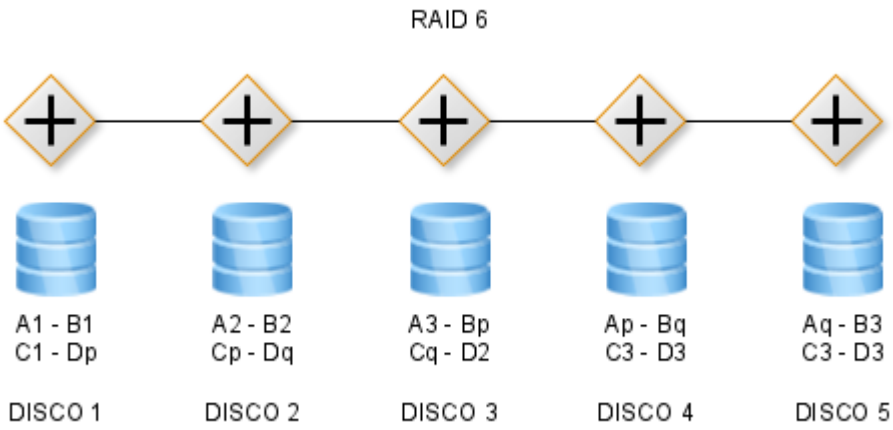


Figura 2.7: Grupo/matriz redundante de discos independientes - Nivel 6.

Cada conjunto de paridades debe calcularse por separado utilizando RAID 6. Esto ralentiza el rendimiento de escritura. RAID 6 también es más caro debido a los dos discos adicionales necesarios para la paridad. A menudo se emplean coprocesadores de controladores RAID para gestionar los cálculos de paridad y mejorar la velocidad de escritura en RAID 6 [170].

El principal inconveniente del RAID 6 es que tarda mucho tiempo en reconstruir la matriz después de un fallo de disco debido a los lentos tiempos de escritura de RAID 6. Incluso con una matriz de tamaño moderado, los tiempos de reconstrucción pueden llegar a las 24 horas [170].

RAID 6 requiere un hardware especial; es importante utilizar un controlador específicamente diseñado para soportarlo.

2.9. RAID 60

RAID 60 (también conocido como RAID 6+0) es una configuración RAID anidada o "híbrida"que proporciona la doble paridad distribuida de RAID 6 con el striping directo a nivel de bloque de RAID 0. Como una matriz RAID 0 dividida en elementos RAID 6, la configuración RAID 60 mínima requiere ocho unidades (véase Fig:2.8) [9] [36] [97].

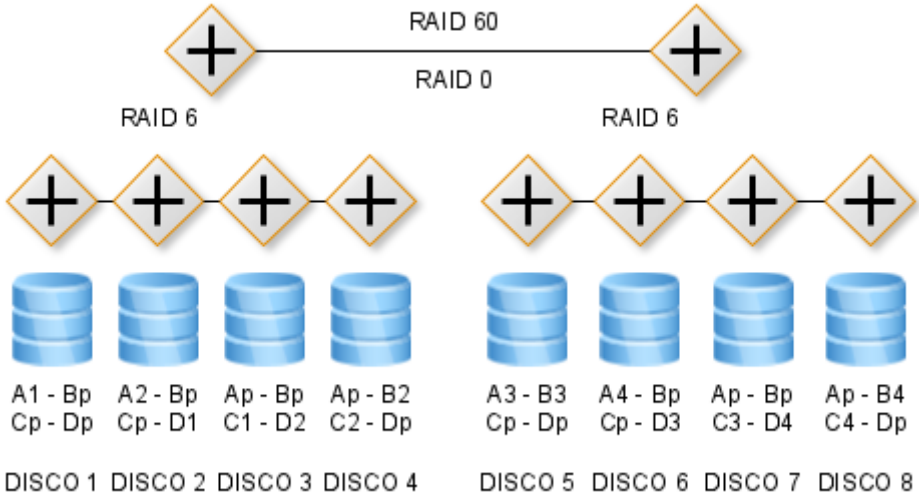


Figura 2.8: Grupo/matriz redundante de discos independientes - Nivel 6+0.

Al igual que RAID 50, RAID 60 (6+0) es un conjunto de discos multinivel. Se trata básicamente de conjuntos RAID 6 que se agregan a un nivel superior en un conjunto RAID 0. Un conjunto RAID ofrece redundancia y puede soportar la pérdida de hasta dos discos en cada

conjunto de paridad [9].

Desde una perspectiva de fiabilidad pura, una matriz RAID 60 es más fiable que las matrices RAID 50 debido en gran parte al disco de paridad adicional empleado en RAID 60. Sin embargo, en cuanto se pierden más de dos discos en un único conjunto de paridad, el conjunto RAID 0 se rompe y la única opción real es la recuperación de datos [9].

2.10. Perspectiva del código de borrado

RAID es alternativa para almacenar datos. Requiere algunos pasos adicionales, pero funcionan bien para garantizar la coherencia en caso de que se rompa una unidad o se corrompan los datos. Durante una operación RAID, un controlador RAID trabaja con la combinación de tres tecnologías: (striping, mirroring and parity), normalmente en alguna combinación de ellas [36].

La duplicación y la copia en espejo distribuyen los datos uniformemente entre varias unidades para equilibrar el número de peticiones a cada unidad, lo que mejora el rendimiento. Para una segunda opinión, el uso de una copia espejo o de replicación puede garantizar la integridad de los datos. Cuando un archivo activo/paridad está dañado, esta comprobación garantizará que los datos no se sobrescriban [97].

Los niveles RAID se denominan según su especificación. RAID 0 es una forma ineficaz de protección de datos sin duplicación ni paridad. RAID 1 refleja los datos, pero no almacena la paridad. RAID 5 divide los datos en varias unidades y proporciona protección de almacenamiento RAID 5 y paridad. RAID 10 adopta RAID 5 y RAID 1 tanto para el rendimiento como para la protección de datos, y es el nivel RAID más común [97] .

Los conjuntos RAID de discos duros permiten lograr transferencias de datos y recuperación de procesos mucho más rápidas; mientras que la adición de unidades puede aumentar el rendimiento del servidor, la resistencia, el

almacenamiento y modificar también la disponibilidad aleatoria [14].

Los controladores RAID no proporcionan una protección adecuada cuando se produce un fallo de disco, y eso plantea un problema de seguridad importante. Cuando falla un disco de una matriz, RAID protege los datos y éstos quedan en un estado inseguro hasta que se pueda sustituir el disco. Además, en la era media, los controladores RAID necesitan varios discos debido a su acceso compartido, lo que amplifica enormemente la vulnerabilidad del host y limita su capacidad de almacenamiento. Además, hoy en día no se puede predecir el volumen de datos almacenados ni siquiera en un único centro de datos. Con los sistemas de localización críticos para las finanzas por ejemplo [1] [184].

Hay razones por las que la codificación de borrado se está convirtiendo en varias alternativas comunes para RAID, y una de ellas es que lleva menos tiempo y esfuerzo que RAID recrear los datos cuando se pierden o se dañan. La codificación de borrado divide los datos en segmentos que se pueden añadir, codifica estos segmentos y, a continuación, los combina. Podría compararse con RAID porque codifica los datos [9].

Como la utilización de la memoria es un aspecto importante a considerar, y dado que los archivos grandes exceden la capacidad de la mayoría de las memorias de los computadores, nuestro codificador emplea dos almacenamientos intermedios de tamaño fijo, un búfer de datos dividido en k bloques y un búfer de codificación dividido en m bloques. El codificador lee toda la información del búfer de datos del fichero grande, la codifica en el búfer de codificación y luego escribe el contenido de ambos búferes en $k + m$ ficheros separados. Repite este proceso hasta que el archivo está totalmente codificado, registrando tanto el tiempo total como el tiempo de codificación [166].

Las empresas de host y de nube que experimentaron las limitaciones de tamaño y rendimiento asociadas a los tamaños de banda de las matrices que han buscado formas diferentes de almacenar los datos. Entre

estas innovaciones se encuentran la informática a hiperescala, el uso de servidores redundantes para proteger los datos, los que utilizan funciones de autocorrección como la nivelación del desgaste y el código de corrección de errores, y los proveedores de SSD que añaden funciones de protección de datos primarios y replicación a las soluciones basadas en software [166].

2.11. Descripción del código de borrado

La codificación de borrado es un método en el que los datos se dividen en trozos o fragmentos particulares y estos fragmentos se codifican con un número configurable de piezas redundantes. Por ejemplo, si la cantidad de datos es de 5 GB, pueden codificarse en 5 fragmentos, y pueden mantenerse en 5 ubicaciones espaciadas en diferentes discos y/o nodos de almacenamiento [110].

La codificación de borrado es un sistema que permite reconstruir los datos si se corrompen. Este tipo de codificación de datos tiene como objetivo permitir que la información que se corrompe pueda reconstruirse utilizando información sobre los datos que está almacenada en otra parte del conjunto, o incluso en otra ubicación [109].

Con los métodos de comprobación de redundancia cíclica, que no utilizan los códigos Reed-Solomon, la matemática utilizada para crear un conjunto de números, requeriría una manipulación extra. Esto implica que los datos perdidos podrían no recuperarse si se pierde un miembro. Sin embargo, los métodos Reed-Solomon se basan en la misma matemática que la de las imágenes [108] [137].

Desarrollada originalmente en 1960, la teoría de corrección de errores permite tratar eficazmente los defectos de los datos de los CD, DVD e incluso de las sondas espaciales. Aunque no se utiliza a menudo, las agencias espaciales la emplean para retransmitir las señales de naves espaciales lejanas, como la *Voyager* [156].

2.11.1. Codificación para borrado

El mercado del almacenamiento se encuentra en una posición única, ya que existe demanda tanto de los mercados tradicionales de HPC (High Performance Computing) y almacenamiento empresarial, como de medios de comunicación y entretenimiento, junto con los nuevos mercados en crecimiento de IA (Artificial Intelligence) y aprendizaje automático. Esto ha creado un enorme potencial para aumentar la cuota de mercado de los proveedores de almacenamiento, siempre que puedan ofrecer el rendimiento necesario que requieren los usuarios de HPC e IA [30] [62].

Mientras los volúmenes de almacenamiento siguen aumentando drásticamente, los proveedores de almacenamiento intentan satisfacer la demanda incrementando el rendimiento del almacenamiento, al tiempo que introducen métodos más eficientes de gestión de datos en grandes plataformas de almacenamiento de varios petabytes [200].

La arquitectura de Qumulo Core se basa en la tecnología SBS (Scalable Block Store) [196]. Este producto proporciona un enfoque de protección de datos eficiente y funciona como la tecnología base que permite una protección de datos eficiente basada en bloques con codificación de borrado [183].

La codificación de borrado es completamente diferente del RAID y resuelve las deficiencias de éste. Es más eficaz, más configurable y más eficiente en términos de espacio que el striping de datos y almacenamiento o el mirroring RAID. Permite un escalado ilimitado de los datos, a la vez que protege todos los datos y mantiene una protección total de los mismos [36] .

Entre otras cosas, utiliza matemáticas avanzadas (es decir, la fórmula Reed-Solomon) para permitir la reconstrucción de los datos que faltan en los bloques de datos (bloques de paridad). El método de duplicación de RAID se realiza utilizando dos copias completas y ofrece una mayor eficiencia en comparación con la codificación de borrado. Sin embargo, la codificación de

borrado requiere un bloque de paridad por cada tres bloques de datos. Por lo tanto, sería más eficiente utilizar este método en su lugar [181].

Por ejemplo para cifrar con codificación $n = 3$; $k = 2$ se necesitan tres datos con un código de borrado triple, tienen que ocurrir tres cosas. Los lugares donde se van a almacenar los datos (los lugares), los datos que hay que proteger (el elemento) y la forma en que se va a realizar la protección (la tecnología). Se accede a tres bloques desde un tercer bloque (el bloque de paridad) mientras se utiliza el algoritmo de codificación. El contenido del bloque de paridad se calcula en función de los bloques principales [1] [13].

Dado que cada bloque se ha escrito en una unidad formateada, cada bloque de medios diferentes se graba en una unidad de almacenamiento diferente. Por ejemplo, los bloques 1 y 2 se almacenan en unidades distintas. Si una de las tres unidades se estropeara, la información de los bloques 1 y 2 aún podría reconstruirse desde las otras dos unidades [13].

2.11.2. Como funciona la codificación

Los administradores de IT que diseñan sistemas de almacenamiento deben planificar con antelación para que los datos de misión crítica no se pierdan cuando se produzca cualquier tipo de fallo. Los sistemas de almacenamiento vienen en diferentes formas pero tienen una cosa en común que es la posibilidad de fallar y perder datos. La utilización de la codificación de borrado evita la pérdida de datos debido a un fallo del sistema o a un desastre (véase Fig:2.9) [171].

Es decir, se lee el primer conjunto de datos (puntos negros sólidos), mientras se trata el nivel de redundancia. Sin embargo, si falta el bloque de datos 1, se lee el segundo conjunto de datos (puntos negros punteados), junto con el bloque de paridad. Esto permite reconstruir el primer conjunto de datos [13].

Si el bloque de datos 2 está en el disco que ha fallado, el sistema lee el bloque de datos 1 y el bloque de paridad. El SBS [196] descubre que el

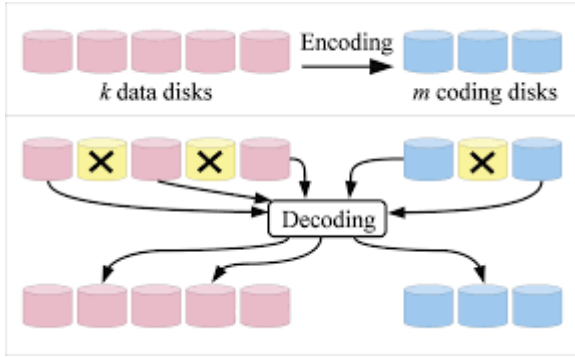


Figura 2.9: Cómo funciona la codificación de borrado. [171]

bloque de datos 2 y el bloque de paridad están en el mismo grupo de bloques y están en ejes diferentes. Así, el sistema puede leer simultáneamente de ambos bloques. Un cifrado de $n = 3$; $k = 2$ tiene una eficacia de $(\frac{2}{3})$ $(\frac{k}{n})$ o del 67 %. Aunque es más seguro que la duplicación, que tiene una eficiencia del 50 %, un cifrado $n = 3$; $k = 2$ sólo protegería contra un único fallo de disco $m = 1$.

2.12. Los códigos en la práctica

Sistemas distribuidos como Hadoop, el sistema de archivos de Google y Windows Azure han evolucionado para incluir soporte para códigos de borrado dentro de sus sistemas, con el fin de aprovechar las ventajas de una mayor eficiencia de almacenamiento en comparación con la triple replicación. Sin embargo, el uso de códigos de borrado tradicionales genera un tráfico de reparación adicional que se traduce en mayores tiempos de reparación. Esto dio lugar a varias construcciones teóricas de códigos para la reparación eficiente de nodos . Uno de los mayores éxitos es, sin duda, la adopción de códigos LR en el clúster de producción de Windows Azure [13] [111].

2.12.1. Códigos de recuperación local (LR)

Los códigos LR. En [89], los autores comparan los resultados de la evaluación del rendimiento de un código LR ($n = 16$, $k = 12$, $r = 6$) con los de un código RS [$n = 16$, $k = 12$] en el clúster de producción Azure y demuestran el ahorro en reparaciones que ofrece el código LR. Posteriormente, los autores implementaron un código LR ($n = 18$, $k = 14$, $r = 7$) en el almacenamiento de Windows Azure y demostraron que este código tiene un grado de reparación comparable al de un código RS [$n = 9$, $k = 6$], pero tiene una sobrecarga de almacenamiento de 1.29 frente a 1.5 en el caso del código RS. Este código ha supuesto un ahorro de millones de dólares para Microsoft [52]. También los autores [193] implementaron HDFS-Xorbas, que utiliza códigos LR en lugar de códigos RS en HDFS-RAID. El código LR de Xorbas se construye sobre un código RS añadiendo partes XOR locales adicionales. La evaluación experimental de Xorbas se llevó a cabo en Amazon EC2 y en un clúster de Facebook, en los que se comparó el rendimiento de reparación del código LR ($n = 16$, $k = 10$, $r = 5$) frente a un código RS [$n = 14$, $k = 10$]. Un segundo sistema de almacenamiento distribuido que cuenta con un complemento de código LR es CephFS (Ceph File System) [27].

2.12.2. Códigos de máxima distancia separable (MDS)

Códigos MDS con ahorro de ancho de banda. El sistema de códigos de borrado Hitchhiker presentado en [178] es una implementación práctica del marco piggybacking introducido en [177]. Los autores implementaron Hitchhiker en HDFS y evaluaron su rendimiento en un clúster de almacenamiento de datos de Facebook.

Hitchhiker se ha incorporado ahora a Apache Hadoop. En [109], se discute la implementación en HDFS de una clase de códigos de array MDS llamados códigos HashTag. El marco teórico de los códigos HashTag se presentó en [110]. Estos códigos permiten bajos niveles de sub-packetización

a expensas de un mayor ancho de banda de reparación y están diseñados para reparar eficientemente nodos sistemáticos.

2.12.3. Códigos Regeneradores (RG)

Códigos RG. NCCloud [88] es uno de los primeros trabajos que se ocupó de la evaluación práctica del rendimiento de los códigos RG. El sistema de almacenamiento NCCloud está construido sobre un código MSR funcional de 2 paridades. En [111] se estudia el rendimiento del código pentágono (que es un código RBT MBR) y de un código heptagon-local (que es un código LRG) en un entorno Hadoop. Estos dos códigos poseen una doble replicación inherente de símbolos de código, tienen una sobrecarga de almacenamiento ligeramente superior a 2 y su rendimiento se compara con la replicación doble y triple.

En [176], los autores presentan una versión de acceso óptimo del código PM MSR, al que denominan código PM-RBT. Se presentan los resultados de una evaluación experimental de un código PM-RBT de tasa $(\frac{1}{2})$ en instancias de Amazon EC2. En [126], los autores introducen códigos de borrado denominados Beehive que se construyen sobre códigos MSR. Estos códigos reparan múltiples fallos simultáneamente y se implementan utilizando el PM MSR en Lenguaje C++ usando la librería de aceleración de almacenamiento de Intel ISAL (Intelligent Storage Library) [94].

En [163], los autores presentan la evaluación de un código MSR de alta tasa conocido como código mariposa tanto en Ceph como en HDFS. Este código es una versión simplificada de los códigos MSR presentados en [65] correspondientes a la presencia de dos nodos de paridad. Este código posee la propiedad de acceso óptimo excepto en el caso de la reparación de un único nodo de paridad, y tiene un nivel de sub-paquetización $\alpha = 2^{k-1}$.

Más recientemente, en [218], los autores presentan el código Clay que corresponde a los códigos de [243] [192] [127]. El código Clay se implementa sobre Ceph basándose en la perspectiva de capas acopladas de [192] y se

evalúa sobre un cluster de Amazon AWS. El código Clay es simultáneamente óptimo en términos de sobrecarga de almacenamiento, ancho de banda de reparación, acceso óptimo y nivel de sub-paquetización. Como parte de este trabajo, se ha añadido soporte de código vectorial a Ceph y se está considerando que el código Clay forme parte de la base de código maestro de Ceph.

Capítulo 3

Desafíos emergentes de resiliencia de datos

La creciente cantidad de datos y la incapacidad de las bases de datos tradicionales para manejar esas cantidades de datos ha dado lugar a un nuevo tipo de almacén de datos denominado Sistemas de Almacenamiento Distribuido. El código de borrado se perfila como la mejor alternativa para garantizar la redundancia de los datos almacenados en la nube. Hasta la fecha se han llevado a cabo numerosas investigaciones fundamentales para que el código de borrado resulte más atractivo para los proveedores de servicios en la nube y más eficaz para los usuarios. Su adopción se está produciendo gradualmente, ya que algunos de los principales proveedores de servicios en la nube (Amazon, Gooogle, Facebook, Microsoft Azure e IBM) están integrando el código de borrado en sus sistemas. Cuestiones como la seguridad, la actualización en directo y el control de la capa de aplicación se han identificado como obstáculos para la adopción del código de borrado por parte de los DSS "Sistema de Almacenamiento Distribuido" [244].

3.1. Capacidad de información

3.1.1. Límites

Por ejemplo los autores en [142] introducen un modelo genérico de sistema de almacenamiento distribuido y se presentan los límites fundamentales. Se presenta la noción de capacidad de información de un sistema distribuido. Sea m el tamaño de los datos de origen en bits. Considérese un sistema de almacenamiento distribuido con N nodos, cada uno de los cuales almacena s bits de datos.

Si Δ denota el tiempo medio entre fallos de un nodo, la tasa de borrado ϵ puede definirse como $\epsilon = \frac{s}{\Delta}$. Cuando se produce un fallo en un nodo, un reparador lleva a cabo la reparación del nodo de forma que se garantice que los datos de origen pueden recuperarse a partir de los datos de los nodos que sobreviven en cualquier momento. El tiempo medio hasta la pérdida de datos (MTTDL) es la cantidad media de tiempo durante la cual pueden recuperarse los datos de origen. Sea γ la tasa de reparación, que es la tasa a la que el reparador lee y escribe datos. Denotemos $\sigma = \frac{\gamma}{\epsilon}$ la relación entre la tasa de reparación y la tasa de borrado. La capacidad de información de un sistema de almacenamiento distribuido se define entonces como la mayor cantidad de datos fuente m para la que es posible un gran MTTDL. Así se demuestra que la capacidad de información se aproxima a $(1 - \frac{1}{2\sigma})\ N\ s$ bits a medida que σ y N aumentan [142].

3.1.2. Almacenamiento líquido

La idea del almacenamiento líquido en la nube descrita en [143], proporciona un almacenamiento de objetos duradero basado en la difusión de datos generados de forma redundante a través de una red de cientos a miles de nodos de almacenamiento potencialmente poco fiables. Un sistema líquido utiliza una combinación de un gran código, reparación retardada y organización de almacenamiento de flujo, en el que se utilizan códigos

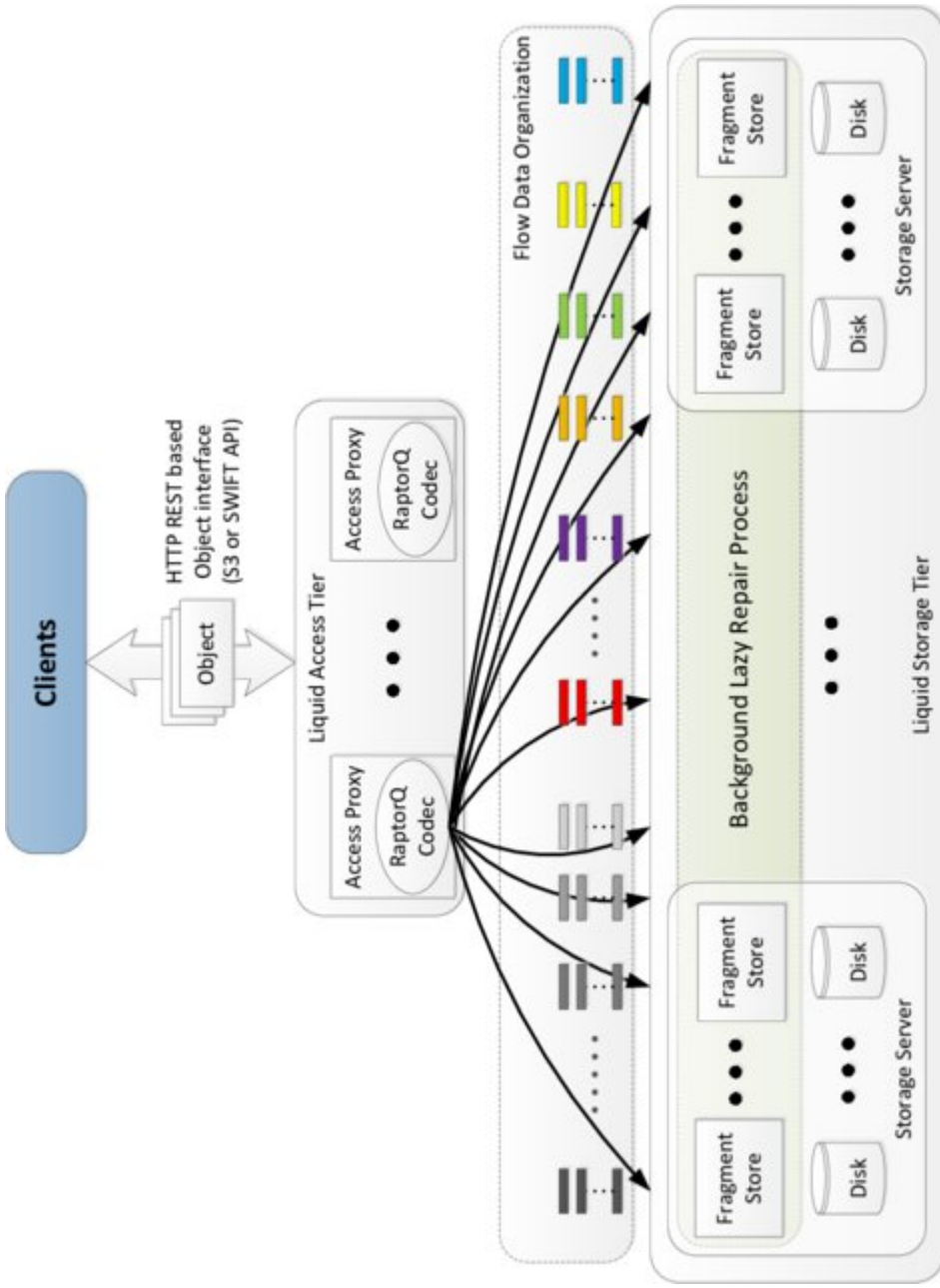


Figura 3.1: Arquitectura de un sistema líquido. [143]

de gran longitud de bloque (por ejemplo, los autores utilizan un código de longitud de bloque 3010 en una de sus simulaciones) para distribuir los datos almacenados pertenecientes a cada objeto entre un gran número de nodos (véase Fig:3.1).

El almacenamiento líquido emplea una estrategia de reparación pasiva en la que la reparación se ejecuta lentamente en segundo plano. Los autores presentan resultados de simulación que muestran que el almacenamiento líquido ofrece un mejor rendimiento MTTDL en comparación con los sistemas basados en códigos de longitud de bloque pequeña. Se demuestra que el rendimiento de los sistemas de almacenamiento líquido se aproxima a los límites de capacidad para el almacenamiento distribuido [142].

3.2. Sistemas de Almacenamiento Distribuido (DSS)

Últimamente el acrónimo DSS "Sistema de Almacenamiento Distribuido" se emplea con menos frecuencia y poco a poco ha sido sustituido por el término "Almacenamiento en la Nube" para hacer referencia a los sistemas de almacenamiento para grandes cantidades de datos [87].

3.2.1. La evolución del DSS

Los sistemas de almacenamiento distribuido han evolucionado mucho debido al auge de la computación en la nube en los últimos años. Los sistemas de archivos distribuidos heredan muchos componentes de los centralizados y los utilizan de forma distribuida. Hay dos formas de aumentar la capacidad de almacenamiento: mediante el escalado o mediante el aumento del número de dispositivos de almacenamiento en un sistema de almacenamiento. El crecimiento de los dispositivos de almacenamiento impone muchos retos relacionados con los protocolos y topologías de

interconexión, la gestión de errores, la coherencia de los datos, la seguridad, etcétera [59].

La continua evolución de las tecnologías de almacenamiento y redes ha favorecido el rápido crecimiento del campo del almacenamiento distribuido en los últimos años, pero la demanda generalizada de más y mejor almacenamiento es una importante fuerza impulsora de este crecimiento [87].

La demanda surge de un aparente mandato económico y cultural de facto para almacenar y archivar tantos bits como sea posible. Esta tendencia plantea nuevos e interesantes retos, ya que a los sistemas de almacenamiento se les pide que almacenen no sólo bits, sino también su contenido semántico [103].

A continuación, se describen los problemas a los que se enfrentan los sistemas de almacenamiento distribuido:

- Acceso a contenidos compartidos.
- Disponibilidad.
- Capacidad de supervivencia.
- Interoperabilidad.
- Búsqueda.
- Almacenamiento en caché.
- Equilibrio de carga.
- Escalabilidad.

Los autores en [244] analizaron dos conjuntos de sistemas de almacenamiento distribuido. El primero (Past, Intermemory and Farsite) archivaba los datos mediante la replicación. El segundo conjunto (Napster, Gnutella, Mojo Nation and Freenet) de sistemas de almacenamiento distribuido con características de sistemas P2P (Peer to Peer).

3.2.2. Sinopsis de los DSS

Iterar en el diseño e implementación de sistemas de almacenamiento distribuido con énfasis en las características requeridas del DSS, transparencia local, almacenamiento permanente, consistencia, disponibilidad, rendimiento y seguridad. Este artículo [87] cubre el sistema de archivos distribuido (DFS), el sistema de archivos de Red Sun (NFS), el sistema de archivos Andrew (AFS), el sistema de archivos Coda (CFS) y el sistema de archivos de Google (GFS). Entre estos, encontramos que solo GFS se está implementando comercialmente actualmente.

Desarrollado a fines de la década de 1990, el sistema de archivos GFS de Google utiliza miles de sistemas de almacenamiento creados a partir de bloques baratos para proporcionar almacenamiento a escala de Petabytes (PB) para una gran comunidad de usuarios con diversas necesidades. Por lo tanto, no sorprende que una de las principales preocupaciones de los diseñadores de GFS fuera la confiabilidad del sistema, sujeto a fallos de hardware, fallos de software del sistema, fallos de aplicaciones y, por último, pero no menos importante, el error humano [148].

El sistema ha sido desarrollado tras un exhaustivo análisis de las características de los archivos y patrones de acceso. Algunos de los aspectos clave de este análisis reflejados en el diseño del GFS son:

- La escalabilidad y la fiabilidad son características críticas del sistema; deben tenerse en cuenta desde el principio, y no en fases posteriores del diseño.

- El tamaño de la gran mayoría de los archivos oscila entre unos pocos Gigabytes (GB) y cientos de Terabytes (TB).

- La operación más común es añadir a un archivo existente; las operaciones de escritura aleatoria en un archivo son extremadamente infrecuentes.

- Las operaciones de lectura secuencial son la norma.
- Los usuarios procesan los datos en masa y se preocupan menos por el tiempo de respuesta.
- Para simplificar la implementación del sistema, el modelo de coherencia debe relajarse sin imponer una carga adicional a los desarrolladores de aplicaciones.

Como resultado de este análisis se tomaron varias decisiones de diseño:

- Segmentar un archivo en fragmentos grandes.
- Implementar una operación atómica de anexión de archivos que permita a varias aplicaciones que operan simultáneamente anexar datos al mismo archivo.
- Construir el clúster entorno a una red de interconexión de gran ancho de banda en lugar de baja latencia.
- Separa el flujo de control del flujo de datos; programa el tráfico de datos de gran ancho de banda enrutando las transferencias de datos a través de conexiones TCP [86] para reducir el tiempo de respuesta.
- Aprovechar la topología de la red enviando los datos al nodo más cercano de la red.
- Eliminar el almacenamiento en caché del lado del cliente; El almacenamiento en caché aumenta la coherencia entre múltiples réplicas del lado del cliente y es poco probable que mejore el rendimiento.
- Garantizar la coherencia canalizando las operaciones de archivo críticas a través de un maestro que controle todo el sistema.

- Minimizar la participación del maestro en las operaciones de acceso a archivos para evitar la contención de puntos de acceso y garantizar la escalabilidad.
- Soportar mecanismos eficientes de comprobación y recuperación rápida.
- Soportar mecanismos eficientes de recolección de información no utilizada.

Arquitectura de clúster Google GFS (véase Fig:3.2). El maestro controla un gran número de servidores de fragmentos; mantiene metadatos como los nombres de los archivos, la información de control de acceso, la ubicación de todas las réplicas para cada fragmento de cada archivo y el estado de los servidores de fragmentos individuales. Algunos de los metadatos se guardan en almacenamiento persistente, por ejemplo, el registro de operaciones registra el espacio de nombres de los archivos, así como el mapeo de archivo a fragmento.

Los servidores de fragmentos se ejecutan en Linux y se comunican directamente con las aplicaciones utilizando metadatos proporcionados por el servidor maestro. El flujo de datos está separado del flujo de control. Las rutas de datos y control se muestran por separado con líneas gruesas para rutas de datos y líneas finas para rutas de control. Las flechas muestran el flujo de control entre la aplicación, el servidor maestro y los servidores de fragmentos (véase Fig:3.2).

3.2.3. Estrategias utilizadas en la computación en nube

En los últimos años, la computación en la nube se ha vuelto cada vez más importante. En los últimos años, empresas como Google, Amazon y Microsoft han construido grandes centros de datos. Estos centros de datos abarcan dominios geográficos y administrativos y, a menudo, están

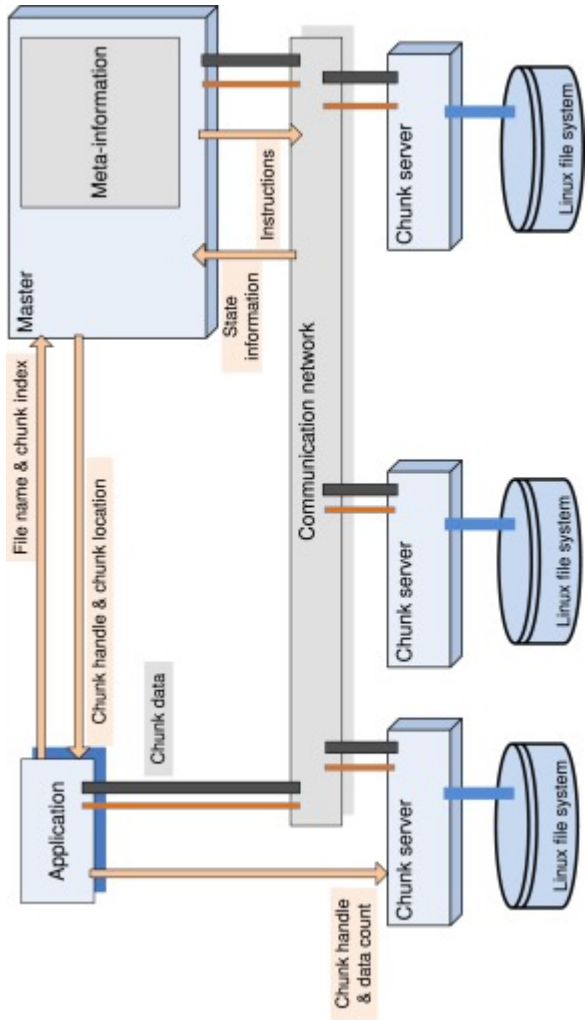


Figura 3.2: Arquitectura de un clúster GFS. [148]

formados por computadoras de escritorio centrales, y la cantidad total de computadoras administradas por estas empresas asciende a millones. Además, el uso de la virtualización permite que un solo nodo físico se represente como un grupo de nodos virtuales, lo que da como resultado un grupo aparentemente inagotable de recursos informáticos. Con economías de escala, estos centros de datos pueden ofrecer CPU, red y almacenamiento a precios significativamente más bajos, lo que a su vez respalda la decisión de muchas instituciones de alojar sus servicios en la nube. En este estudio [59] se examinaron los sistemas xFS, Amazon S3, Dynamo, GFS, Bigtable y Azure. Compararon estos sistemas en términos de patrones de falla, replicación, acceso a datos, integridad, garantías de consistencia, metadatos, ubicación de datos y seguridad. Los problemas identificados fueron errores en el uso de componentes básicos, ubicación de datos y replicación [148].

3.2.4. Almacenamiento en la nube

El almacenamiento en la nube es un modelo de servicio en el que los datos se transfieren y almacenan en un sistema de almacenamiento remoto donde se mantienen, administran, respaldan y ponen a disposición de los usuarios a través de una red (generalmente Internet). Los usuarios suelen pagar por su almacenamiento de datos en la nube con una tarifa de uso mensual [148].

El almacenamiento en la nube se basa en una infraestructura de almacenamiento virtualizada con interfaces accesibles, flexibilidad y escalabilidad casi instantáneas, múltiples arrendamientos y recursos escalables. Los datos basados en la nube se almacenan en grupos lógicos o en servidores de almacenamiento discretos básicos ubicados en las instalaciones o en centros de datos operados por proveedores de servicios en la nube de terceros [103].

Los proveedores de servicios en nube gestionan y mantienen los datos transferidos a la nube. Los servicios de almacenamiento se proporcionan bajo demanda en la nube, con capacidad que aumenta y disminuye según sea

necesario. Las organizaciones que optan por el almacenamiento en la nube eliminan la necesidad de comprar, gestionar y mantener una infraestructura de almacenamiento interna [148].

El almacenamiento en la nube reduce en gran medida el costo por Gigabyte de almacenamiento, pero los proveedores de almacenamiento en la nube aumentan los gastos operativos, lo que puede hacer que la tecnología sea significativamente más costosa, dependiendo de cómo se use [149].

Una de las primeras investigaciones de [103] en 2011 identificó el almacenamiento en la nube como un nuevo concepto que surge al mismo tiempo que la computación en la nube, que puede dividirse en nube pública, nube privada y nube híbrida. El autor presenta brevemente el almacenamiento en la nube. Abarcan las principales tecnologías en la nube y el almacenamiento en la nube [148].

El sistema de almacenamiento masivo de datos Google GFS [87] y el popular HDFS de Hadoop de código abierto [39], han sido utilizados en importantes áreas tecnológicas y el almacenamiento en la nube. Es decir, se está convirtiendo en una investigación en apogeo tanto para el mundo académico como para la industria [103].

3.2.5. Arquitectura y almacenamiento en la nube

En el auge de datos actual, no sorprende que el almacenamiento en la nube también esté creciendo en popularidad. Los datos de más rápido crecimiento son los datos de archivo, que son adecuados para el almacenamiento en la nube debido a varios factores, como el costo, la frecuencia de acceso, la protección y la disponibilidad. Pero no todo el almacenamiento en la nube es igual. Un proveedor puede estar preocupado principalmente por el costo, mientras que otro puede estar preocupado principalmente por la disponibilidad o el rendimiento. Ninguna arquitectura tiene un enfoque único, pero el grado en que una arquitectura implementa cierta funcionalidad define su mercado y los patrones de uso

correspondientes [148].

La computación en la nube es una plataforma de computación y servicios floreciente que ha arrasado con la computación empresarial. A través de los servicios de red, las plataformas de computación en la nube facilitan el acceso a la infraestructura de almacenamiento y la computación de alto rendimiento de una organización. La computación en la nube es también un nuevo modelo de negocio. La computación en la nube ofrece escalabilidad masiva, alto rendimiento y confiabilidad a un costo extremadamente bajo en comparación con los sistemas de almacenamiento dedicados [135] [148].

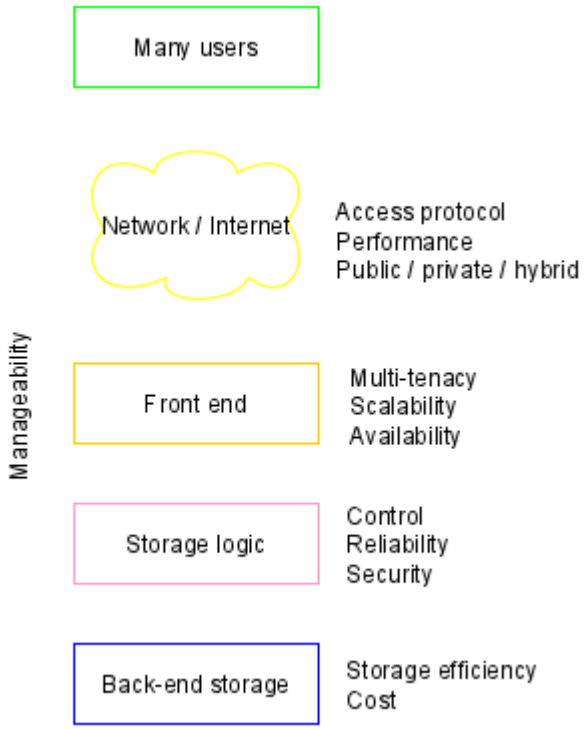


Figura 3.3: Arquitectura genérica de almacenamiento en la nube . [102]

Las arquitecturas de almacenamiento en nube consisten principalmente

en la entrega de almacenamiento bajo demanda de forma altamente escalable y multiusuario. En términos generales (véase Fig:3.3), las arquitecturas de almacenamiento en nube constan de un front-end que exporta una API para acceder al almacenamiento. En los sistemas de almacenamiento tradicionales, esta API es el protocolo SCSI; pero en la nube, estos protocolos están evolucionando. Podemos encontrar front-ends de servicios Web, front-ends basados en archivos e incluso front-ends más tradicionales (como Internet SCSI, o iSCSI). Detrás del front-end hay una capa lógica de almacenamiento, esta capa implementa una serie de funciones, como la replicación y la reducción de datos, sobre los algoritmos tradicionales de colocación de datos (teniendo en cuenta la colocación geográfica). Por último, el back-end implementa el almacenamiento físico de los datos. Puede tratarse de un protocolo interno que implemente funciones específicas o de un back-end tradicional para los discos físicos [102].

3.2.6. Big Data en la nube

Big Data es una combinación de datos estructurados, semiestructurados y no estructurados recopilados por organizaciones que pueden extraerse para obtener información y utilizarse en proyectos de aprendizaje automático, modelado predictivo y otras aplicaciones de análisis avanzado [75].

Los sistemas que procesan y almacenan “Big Data” y las herramientas que respaldan el uso analítico de big data se han convertido en partes comunes de la arquitectura de gestión de datos en las organizaciones. Los grandes datos [75] suelen tener las siguientes caracteristicas:

- El gran volumen de datos en muchos entornos.
- La gran variedad de tipos de datos que se almacenan con frecuencia en los sistemas de “Big Data”.
- La velocidad a la que se generan, recopilan y procesan gran parte de los datos.

- Veracidad
- Valor
- Variabilidad

En la actualidad, los datos que deben explorar y explotar los sistemas informáticos que aumentan a un ritmo exponencial. La cantidad masiva de datos o los llamados "Big Data" ejercen presión sobre las tecnologías existentes para proporcionar un soporte escalable, rápido y eficiente. Las aplicaciones recientes y el actual apoyo al usuario de la informática multidominio han contribuido a la migración de la informática centrada en los datos a la centrada en el conocimiento. Sin embargo, sigue siendo un reto almacenar y colocar o migrar de forma óptima estos enormes conjuntos de datos entre "Centro de Datos" (DC). En particular, debido al frecuente cambio de comportamiento de las aplicaciones y los DC (es decir, recursos o latencias), también es necesario analizar los patrones de acceso o uso de los datos. En primer lugar, el objetivo principal es encontrar una ubicación de almacenamiento de datos más adecuada que mejore el coste global de colocación de los datos, así como el rendimiento de la aplicación [148] [75].

En [149] los autores, ofrecen una visión general del estado del arte de la ubicación de Big Data centrada en la nube junto con las metodologías de almacenamiento de datos. Es un intento de poner de relieve la correlación real entre estos dos en términos de un mejor apoyo a la gestión de Big Data. Nos centramos en los aspectos de gestión que se ven bajo el prisma de las propiedades no funcionales. Al final, se puede apreciar el profundo análisis de las respectivas tecnologías relacionadas con la gestión de Big Data y ser guiados hacia su selección en el contexto de la satisfacción de sus requisitos de aplicación no funcionales. Además, se presentan retos que ponen de relieve las lagunas actuales en la gestión de Big Data y marcan el camino por el que debe evolucionar en un futuro próximo [149] [75].

Si bien Big Data no es lo mismo que una cantidad específica de datos,

las implementaciones de Big Data a menudo incluyen Terabytes, Petabytes o incluso Exabytes de datos creados y recopilados a lo largo del tiempo.

3.2.7. Visión, problemas y retos

Desde electrodomésticos hasta empresas industriales, la industria de las tecnologías de la información y la comunicación (ICT) está cambiando el mundo. Participamos en nuevas tecnologías (como computación en la nube, computación en la niebla, Internet de las cosas (IoT), inteligencia artificial (AI) y atención médica y electrodomésticos y generamos cantidades masivas de datos [148] [75].

Para 2025, se espera que 75 mil millones de dispositivos procesen más de 175 Zettabytes (ZB) de datos por año. La tecnología 5G, o tecnología celular, ha aumentado considerablemente las velocidades de la red, lo que permite a los usuarios cargar videos de ultra alta definición en tiempo real, lo que generará flujos masivos de grandes datos "Big Data". Además, los dispositivos inteligentes equipados con inteligencia artificial y que actúan como humanos en la red (como los coches autónomos, etc.) también generarán big data. Estos cambios repentinos y la generación masiva de datos plantean un serio desafío para el almacenamiento y la gestión de datos tan grandes y heterogéneos [74].

La investigación más reciente [70] proporciona información de vanguardia sobre los problemas y desafíos del almacenamiento heterogéneo de grandes datos, sus contramedidas (es decir, desde una perspectiva de seguridad y gobernanza) y futuras oportunidades de almacenamiento en la nube. Estos temas se revisan en detalle y se revelan nuevos impulsos para los investigadores de almacenamiento en la nube [123].

3.3. DSS con códigos de borrado

En los sistemas de almacenamiento con código de borrado, un objeto de datos se divide en m bloques y se codifican para generar los n bloques redundantes ($m \leq n$). Utilizando las propiedades de los códigos de borrado, el sistema es capaz de reconstruir los datos originales recogiendo m de los n bloques codificados. El factor de redundancia de los datos se define como ($\frac{n}{m}$), y como m es siempre menor que n, los datos pueden reconstruirse siempre que el número de fallos de los nodos de almacenamiento no supere (m - n). Algunos ejemplos de códigos de borrado son:

- Códigos Reed-Solomon
- Códigos Jerárquico
- Códigos Regenerativos
- Códigos Autorreparables

Los sistemas P2P están en posición de aprovechar las ganancias en ancho de banda de red, capacidad de almacenamiento y recursos computacionales para proporcionar infraestructuras de almacenamiento duraderas a largo plazo. En [226], realizan una comparación cuantitativamente la creación de una infraestructura de almacenamiento distribuido autorreparable y resistente a los fallos mediante un sistema replicado o un sistema resistente al borrado. Se demostró que los sistemas que emplean códigos de borrado tienen un tiempo medio hasta fallos varios órdenes de magnitud superior al de los sistemas replicados con requisitos similares de almacenamiento y ancho de banda. Y lo que es más importante, los sistemas resistentes al borrado utilizan un orden de magnitud menos de ancho de banda y almacenamiento para proporcionar una durabilidad del sistema similar a la de los sistemas replicados [226].

Los códigos de borrado suponen un ahorro de capacidad de almacenamiento de entre el 60 y el 70 % en comparación con la replicación, con la misma cantidad de redundancia. El problema de los códigos de borrado es que, junto con las ventajas de reducir el espacio de almacenamiento necesario para la misma cantidad de redundancia, también conllevan requisitos adicionales de ancho de banda, sobrecargas de almacenamiento y cargas informáticas al crear los bloques y repararlos. Encontrar el código de borrado óptimo en términos de menor procesamiento y ancho de banda, sigue siendo un problema abierto.

Los códigos de borrado pueden clasificarse según [13] las categorías especificadas:

- Códigos de distancia máxima separable (véase Fig:3.4) (MDS).
  - Códigos Reed-Solomon
  - Códigos Regeneradores
  - Códigos Autorreparables
- Códigos de regeneración de ancho de banda mínimo (MBR)
  - Códigos Reparables Localmente
  - Códigos de Reconstrucción Local
  - Códigos de Regeneración Local
- Reducción de sobrecarga de almacenamiento (MSR)
  - Reparación Exacta Regeneración Mínima de Almacenamiento
  - Códigos de Regeneración Funcional
  - Almacenamiento Mínimo Funcional Código Regenerador - Protección de Integridad de Datos (FMSR-DPI)

3.3.1. Códigos Reed-Solomon (RS)

El código Reed-Solomon [181] [169] [171] se implementa en dos partes: el codificador y el decodificador. El codificador toma un bloque de datos digitales y añade bits redundantes adicionales, que se calculan mediante el código Reed-Solomon. El decodificador procesa cada bloque e intenta corregir los errores y recuperar los datos originales [13]. Si se produce un error durante la transmisión o el almacenamiento, el error puede detectarse y corregirse en función de las características del Reed-Solomon.

Durante el proceso de codificación, los datos se dividen en conjuntos de k símbolos de datos de tamaño fijo s y se genera una cierta cantidad de símbolos de paridad y se añaden para obtener una codificación de tamaño n. Los símbolos de paridad derivados serán (n - k). También significa que se pueden corregir hasta (n - k) símbolos si se borran los (n - k) símbolos de datos.

3.3.2. Códigos RG

Los Códigos Regeneradores [50] abordan el problema de la reconstrucción (también llamada reparación) de fragmentos codificados perdidos contactando sólo con un subconjunto de fragmentos, lo que reduce los requisitos de ancho de banda.

En los códigos RS tradicionales [50], cada fragmento se considera un único símbolo que se codifica junto con otros fragmentos y se almacena en distintos nodos. El código regenerativo asume que cada fragmento consta de α símbolos. Es decir, un nodo no sólo almacenará un fragmento codificado, sino varios almacenados como uno solo [13]. Durante el proceso de reparación (un nodo fallido), se contacta con un conjunto aleatorio de d nodos residuales (conocidos como nodos ayudantes) y se descargan $\beta \leq \alpha$ símbolos de cada nodo. El ancho de banda de reparación, la cantidad total de datos descargados asciende a $\gamma = \mathrm{d}\beta$.

Los códigos regeneradores se caracterizan por [n, k, d], donde d especifica

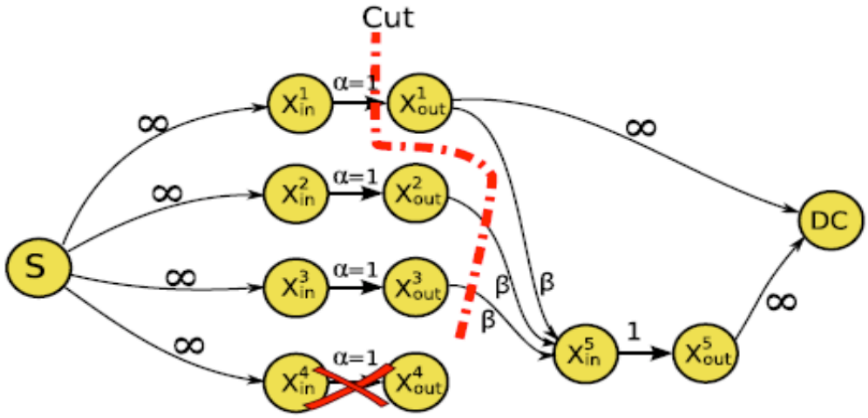


Figura 3.4: Inspirado en los códigos de red [156].

la cantidad de nodos que deben contactarse para la reparación. Con estas nuevas características, surgió el problema de encontrar el equilibrio entre almacenamiento y ancho de banda de reparación (véase Fig:3.4).

3.3.3. Códigos LR

En los Códigos de Reconstrucción Local (LRC) [89], la colocación de los bloques y el reconocimiento de los bastidores son de vital importancia. El LRC disminuye la cantidad de tráfico de red que es necesario transmitir al leer bloques para la reconstrucción del objeto. Al colocar los fragmentos más cerca unos de otros se reduce el tiempo de transmisión, por lo que también se minimiza el tiempo de reconstrucción y reparación.

Otra novedad de LRC es el uso de la paridad local. Un grupo de bloques se codifica para generar una paridad local y, a continuación, otros grupos realizan la misma rutina. Por último, se crean paridades globales codificando todos los bloques del objeto asociado. Esto se hace manteniendo el mismo nivel de tolerancia a fallos (véase Fig:3.5). Lo más destacado de este método está orientado a la reparación de un único fallo [13].

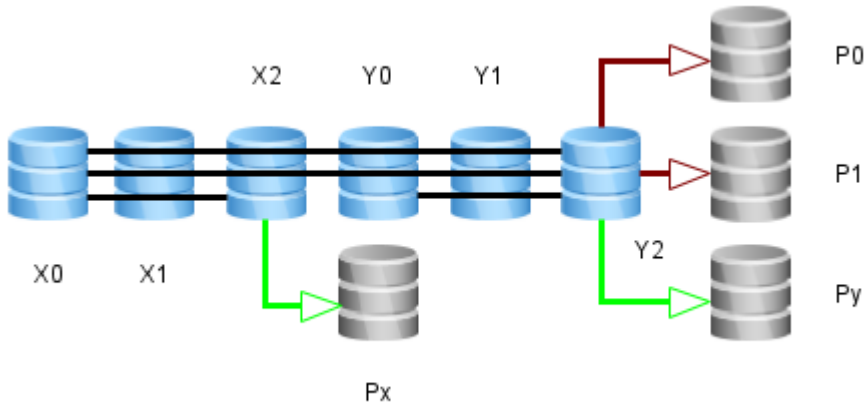


Figura 3.5: LRC $k = 6$ fragmentos de datos, $l = 2$ paridades locales y $r = 2$ paridades globales [89].

Los LRC son significativos para despliegues en los que no sólo importa el ancho de banda de reparación, sino también el número de nodos con los que hay que estar en comunicación durante la reparación. El número de nodos con los que es necesario ponerse en contacto durante la reparación suele denominarse localidad de reparación [62].

3.3.4. Códigos Jerárquicos

La redundancia es la técnica básica para proporcionar fiabilidad en sistemas de almacenamiento formados por múltiples componentes. Un esquema de redundancia define cómo se producen y mantienen los datos redundantes [156].

El esquema de redundancia más sencillo es la replicación, que sin embargo adolece de ineficacia de almacenamiento. Otro enfoque es la codificación de borrado, que proporciona el mismo nivel de fiabilidad que la replicación utilizando una cantidad de almacenamiento significativamente

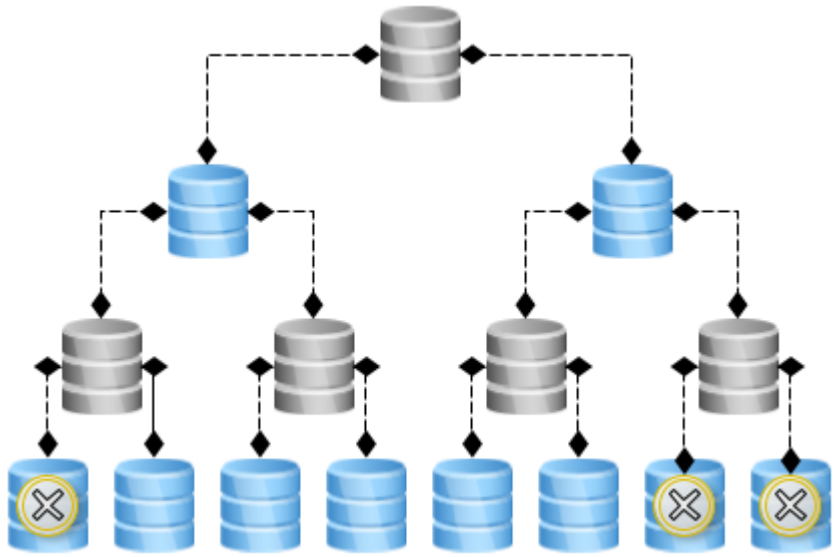


Figura 3.6: Códigos Jerárquicos y Piramidales [156].

menor [13].

Cuando se pierden datos redundantes, hay que sustituirlos. Mientras que la sustitución de los datos replicados consiste en una simple copia, con los códigos de borrado se convierte en una operación compleja: los nuevos datos se producen realizando una codificación sobre otros datos disponibles. La cantidad de datos que hay que leer y codificar es d veces mayor que la cantidad de datos producidos, donde d, denominado grado de reparación, es mayor que 1 y depende de la estructura del código. Esto implica que la codificación tiene un mayor coste computacional y de E/S, lo que, para los sistemas de almacenamiento distribuido, se traduce en un aumento del tráfico de red [53].

La mayoría de los códigos de borrado, como los códigos Reed-Solomon [169] [171] o los códigos MBR [89] están pensados para resolver los problemas

computacionales de los construcción de bloques. Sin embargo, no abordan el problema del ancho de banda limitado. Dos códigos de borrado que pretenden encontrar una solución flexible que permita reducir el tráfico de red manteniendo las ventajas de los códigos de borrado tradicionales son los códigos jerárquicos [53] y los códigos autorreparables [158]. Sin embargo, ambas soluciones conllevan un coste de almacenamiento, lo que no es un problema importante en los sistemas P2P, ya que el almacenamiento está ampliamente disponible (véase Fig:3.6).

3.3.5. Códigos Autorreparables

Los códigos de borrado constituyen una alternativa eficaz a la redundancia basada en la replicación en los sistemas de almacenamiento en red[13]. Sin embargo, conllevan una elevada sobrecarga de comunicación para su mantenimiento, cuando se pierden algunos de los fragmentos codificados y es necesario reponerlos. Esta sobrecarga se debe a la necesidad fundamental de mantener por separado primero una copia del objeto completo antes de poder generar y reponer cualquier fragmento codificado individualmente. Existe un gran interés por explorar alternativas, entre las que destacan los códigos regeneradores (RGC) y los códigos jerárquicos (HC) [158].

Los códigos autorreparables tienen dos objetivos principales:

- Minimizar la cantidad absoluta de transferencia de datos necesaria para recrear los datos perdidos de un nodo.
- Minimizar el número de nodos que hay que contactar para reparar el fallo de un nodo.

Como tal, en los códigos autorreparables, un objeto también se dividido en bloques y se crean algunos bloques redundantes para aumentar la disponibilidad en caso de reparación [156].

La diferencia fundamental entre los códigos autorreparables (*Self Repairing*) y códigos jerárquicos está en su construcción, y en satisfacer

algunas propiedades fundamentales. Estas propiedades permiten no sólo una baja sobrecarga de comunicación para recrear un fragmento perdido, sino también la reconstrucción independiente de diferentes fragmentos perdidos en paralelo, posiblemente en diferentes partes de la red. La diferencia fundamental entre los códigos autorreparables y los códigos jerárquicos es que los distintos fragmentos codificados en los códigos jerárquicos no tienen funciones simétricas [158].

3.4. Cómo funciona la codificación de borrado

La codificación de borrado funciona dividiendo una unidad de datos, como un archivo u objeto, en varios fragmentos (bloques de datos) y luego creando fragmentos adicionales (bloques de paridad) que se pueden utilizar para la recuperación de datos. Para cada fragmento de paridad, el algoritmo EC calcula el valor de la paridad basándose en los fragmentos de datos originales (véase Fig:3.7). Los datos y los fragmentos de paridad se almacenan en varias unidades para protegerlos contra la pérdida de datos en caso de que una unidad falle o los datos se corrompan en una de las unidades. Si ocurre tal evento, los fragmentos de paridad se pueden usar para reconstruir la unidad de datos sin experimentar pérdida de datos [156].

3.5. EC vs RAID

RAID [36] está diseñado para proteger contra fallos de unidades, pero muchos niveles de RAID tradicionales están diseñados para manejar solo una o dos fallos de unidades en un grupo. Con las unidades de mayor capacidad de la actualidad, si una unidad falla, la recuperación de datos y el proceso de recuperación pueden demorar días. La capacidad de tolerar solo una o dos fallos no es suficiente para proporcionar redundancia para la protección de datos. La codificación de borrado se basa en una alta disponibilidad de

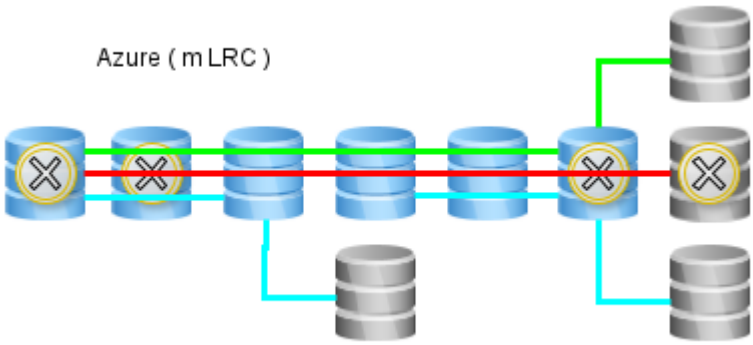


Figura 3.7: Azure (mLRC) $k = 6$, $l = 2$, $r = 2$. [156]

datos [108]. En resumen, divide los datos en fragmentos configurables y los distribuye a través de un conjunto de diferentes sistemas de almacenamiento.

En pocas palabras, la codificación de borrado (EC) es un método de protección de datos que divide los datos en bloques. Luego se expanden y codifican con datos redundantes y se almacenan en varios medios. La codificación de borrado aumenta la redundancia del sistema para tolerar fallos.

La codificación de borrado y el RAID a veces se confunden, pero son muy diferentes [133]. El RAID permite almacenar datos en distintas ubicaciones y protege contra los fallos de las unidades. En la codificación de borrado, los datos se dividen en partes, luego se expanden y se codifican. Después, los bloques de datos se guardan en varias ubicaciones. Aunque el RAID facilita la protección de datos, la codificación de borrado consume menos almacenamiento y el RAID es más eficiente en términos de tiempo. Tanto el RAID como la codificación de borrado son deseables en función de las situaciones.

Las empresas que requieren un entorno de almacenamiento seguro deben utilizar tecnología de codificación de borrado. Aquí hay algunas situaciones

en las que la codificación de borrado puede ser muy útil:

- Sistemas de matrices de discos
- Redes de datos
- Aplicaciones de almacenamiento distribuido
- Almacenamiento de objetos
- Almacenamiento de archivos

Un caso de uso habitual de la codificación de borrado es el almacenamiento en la nube basado en objetos. Como la codificación de borrado requiere una alta utilización de la CPU e incurre en latencia, es adecuada para aplicaciones de archivado. La codificación de borrado es menos adecuada para cargas de trabajo primarias, ya que no puede proteger contra amenazas a la integridad de los datos [154].

Un caso de uso común para la codificación de borrado es el almacenamiento en la nube basado en objetos. Dado que la codificación de borrado requiere una alta utilización de la CPU e introduce latencia, es adecuada para aplicaciones de archivo. El paradigma actual es: la codificación de borrado es menos adecuada para cargas de trabajo más grandes porque no puede prevenir las amenazas a la integridad de los datos [147].

3.6. Beneficios de la codificación de borrado

A diferencia de los modelos de hardware, la codificación de borrado (EC) es una opción que está ganando popularidad rápidamente. EC se basa en algoritmos y, por lo tanto, es independiente del hardware específico. No requiere un controlador de hardware dedicado y proporciona una mayor robustez. Aún mejor, también brinda protección durante la recuperación. Según el grado de persistencia, se puede lograr una recuperación completa

cuando solo la mitad de los elementos de datos (cada elemento) están disponibles. En este sentido, tiene grandes ventajas sobre RAID. Además, EC también utiliza menos espacio de almacenamiento en comparación con la duplicación [35]. Las principales ventajas se describen a continuación:

- Utilización del espacio de almacenamiento.
- Mayor fiabilidad.
- Idoneidad.
- Sólo un subconjunto.
- Flexibilidad.

3.7. Utilidad de la codificación de borrado

Primero, considere las capacidades del sistema. No todos los sistemas hiperconvergentes admiten la codificación de borrado. Aunque la codificación de borrado se está volviendo más común, aún no es universalmente compatible. Suponiendo que su sistema hiperconvergente admita RAID y codificación de borrado, el siguiente paso es determinar qué opción es la mejor para su plataforma [108].

RAID es una tecnología madura que existe desde hace décadas. Sin embargo, el problema con el uso de RAID es que, dependiendo de su configuración, puede tener una sobrecarga muy alta o brindar una protección limitada. Por ejemplo, una matriz RAID 5 puede proteger un sistema hiperconvergente de la falla de una sola unidad. Sin embargo, si falla una unidad, es posible que lleve demasiado tiempo reconstruir la matriz después de reemplazar la unidad fallida. Durante este tiempo, el sistema es vulnerable a más fallos en el disco [170].

Por el contrario, una matriz RAID 10 está protegida contra múltiples fallos simultáneas, pero con un costo adicional del 50 %. Por ejemplo, si una

matriz de este tipo consta de 10 discos, 5 de ellos se utilizarán solo para la duplicación de datos. Eso es mucha capacidad desperdiciada [9].

Mientras que, la codificación de borrado le permite definir el nivel de protección de datos. No está limitado a estrictos esquemas de protección de datos como RAID 5 o RAID 10. La codificación de borrado también tiende a ser más flexible. Además, la codificación de borrado tiende a hacer un mejor uso del espacio disponible que las soluciones RAID espejo [203].

Capítulo 4

Evaluación de códigos de borrado en Dicoogle PACS

Este capítulo pretende maximizar el rendimiento de codificación y decodificación de imágenes médicas a través del paralelismo computacional, los ficheros utilizados son de tipo DICOM (Digital Imaging and Communication in Medicine), ya que, es un estándar para la transmisión de imágenes y datos en hardware para fines médicos y se utiliza comúnmente para ver, almacenar, imprimir y transmitir imágenes. Como parte de la forma en que DICOM transmite archivos, la plataforma PACS (Picture Archiving and Communication System), Dicoogle, se ha convertido en una de las plataformas de procesamiento y visualización de imágenes más demandadas. La arquitectura Dicoogle PACS no garantiza la recuperación de información de imagen en caso de pérdida de información. Por lo tanto, esta investigación propone una solución de recuperación de archivos en la arquitectura Dicoogle. Los resultados muestran que la propuesta es óptima en términos de tiempo de procesamiento de imágenes para el sistema de almacenamiento Dicoogle PACS.

4.1. Introducción

En el campo de la medicina, es necesario establecer normas o estándares para los formatos de imagenes a compartir entre disposivos. Específicamente, estas normas abordan aspectos de la mensajería y la comunicación entre equipos de imágenes tales como: equipos de rayos X fijos y portátiles, Ultrasonido (US), Tomografía computarizada sencilla (CT) y multicapa (MSCT), Resonancia magnética (MRI), Tomografía por positrones (PET), Tomografía por emisión de fotones simples (SPECT) y Angiografía Coronaria. En este sentido, la Asociación Nacional de Fabricantes Eléctricos (NEMA) de Estados Unidos conjuntamente con el Colegio Americano de Radiología (ACR) han contribuido en el desarrollo de normas y estándares para el procesamiento de imágenes médicas [167].

La transmisión entre dispositivos de imágenes médicas requiere de estandares de comunicación y almacenamiento fiables. Para ello, el DICOM (Digital Imaging and Communication On Medicine) es un protocolo estándar de comunicación entre sistemas de información. Asímismo, DICOM es un formato de almacenamiento de imágenes médicas que aparece como solución a los problemas de interoperabilidad entre diversos dispositivos, incluyendo el hardware de propósito médico [8].

El estandar DICOM integra el sistema de comunicación PACS (Picture Archiving and Communication System), el cual se ha convertido en el principal enfoque de almacenamiento de imágenes médicas tales como: radiología, imagénes de ultrasonido y medicina nuclear [240]. En este sentido, surge el término de sistema de información médica, el cual consiste en la comunicación y almacenamiento de imágenes médicas como solución para el sistema de gestión de tecnología de procesamiento de imágenes de un entorno hospitalario. Una de las ventajas importantes de PACS, es que permite reducir de manera significativa las dosis de radiación en los pacientes y los costos operativos del departamento de radiología [240].

El mercado de los PACS es heterogéneo, con diversos proveedores de

PACS que han desplegado instalaciones en centros sanitarios. Las interfaces de consulta y recuperación de DICOM proporcionadas por los PACS tienen múltiples variaciones, relacionadas con las clases SOP (Service Object Pair) implementadas, las sintaxis de transferencia, las negociaciones extendidas, los atributos y tipos de correspondencia. Estas variaciones pueden hacer que la integración de un nuevo consumidor DICOM con un PACS sea compleja y lleve mucho tiempo.

Aunque la mayoría de los productos PACS proporcionan una declaración de conformidad con DICOM como una descripción de sus capacidades de consulta/recuperación, no existe un análisis colectivo que describa las diversas capacidades y variaciones de consulta y recuperación de los sistemas de tratamiento de imágenes [21]. También, otro aspecto importante es el referente a la seguridad del sistema. Con respecto a esto, desde un punto de vista práctico, las medidas de seguridad específicas de los PACS deben considerarse junto con las medidas aplicables a la infraestructura de Tecnología de Información (IT) en su conjunto, con el fin de fortalecer la seguridad de los sistemas PACS que son accesibles desde Internet [54].

En la actualidad, las modalidades de imágenes médicas generan nuevos datos y su desarrollo ha sido exponencial. PACS gestiona el almacenamiento y distribución de la información, sin embargo, la alta disponibilidad es un reto para las tecnologías actuales. La cantidad de datos que se debe almacenar y distribuir requiere una gran infraestructura local para manejar la carga y tolerancia a fallos, lo que genera mayor consumo económico [120].

En este punto, vale la pena mencionar que en los últimos años la creciente necesidad de durabilidad y eficiencia en el almacenamiento de ficheros hacen que los códigos de borrado (EC) sean un nuevo objetivo de investigación. Por ejemplo, el algoritmo de código de borrado Reed-Solomon (RS) $(n; k)$ [181]. El algoritmo RS realiza los procesos de codificación dados n bloques de datos originales y decodificación para recuperar datos perdidos desde k bloques redundantes. De acuerdo a lo anterior, es importante mencionar que,

los sistemas de almacenamiento distribuidos basados en códigos de borrado son cada vez más utilizados por los proveedores de almacenamiento para Big Data. Estos ofrecen la misma fiabilidad que la replicación de datos, con una disminución significativa de la cantidad de almacenamiento para mantener fiabilidad en los datos. Sin embargo, surge el problema cuando los nodos del sistema de almacenamiento se encuentran distribuidos en una zona geográficamente extensa [153].

Los sistemas de almacenamiento con código de borrado son rentables y se despliegan en grandes centros de datos para lograr una alta fiabilidad con un bajo coste de almacenamiento [89], [214]. El código Reed-Solomon [181] es parte de una popular familia de códigos de borrado utilizados en ColossusFS de Google [42], Hadoop Distributed File System (HDFS) [214], [193], y otros sistemas de almacenamiento [194], [178].

A manera de resaltar la importancia de usar algoritmos de código de borrado, a continuación se mencionan implementaciones de estos algoritmos en sistemas de ficheros distribuidos.

El sistema An Erasure-Coded Data Archival System for Hadoop Clusters (aHDFS) aprovecha los procesos de codificación paralelos y canalizados. Los procesos de codificación permiten acelerar el rendimiento del archivo de datos en el sistema de archivos distribuido Hadoop (HDFS) en clusters Hadoop. En particular, aHDFS aprovecha las réplicas de datos de tres vías y los modelos de programación MapReduce en el clúster Hadoop para aumentar el rendimiento [38].

También, el sistema HDFS-RAID de Facebook implementa la codificación y decodificación de Reed-Solomon [181] en el formato de un sistema de archivos RAID (Redundant Array Of Independent Disks) distribuido que se ejecuta sobre HDFS [193]. Es decir, las versiones anteriores de HDFS lograban la tolerancia a fallos mediante la replicación de múltiples copias de datos similar a RAID 1 en las matrices de almacenamiento tradicionales.

Además, el sistema HDFS-EC (sistema de almacenamiento que imple-

menta código de borrado) desarrollado e implementado por Cloudera reduce significativamente la sobrecarga de almacenamiento al tiempo que logra una tolerancia a fallos similar o mejor mediante el uso de celdas de paridad similar a RAID 5 [39].

Expuestas las implementaciones anteriores y antes de la introducción a los algoritmos de código de borrado, es importance mencionar que HDFS utilizaba exclusivamente la replicación *3x* para la tolerancia a fallos, lo que significa que un archivo de 1 GB utiliza 3 GB de espacio de almacenamiento. Con EC, el mismo nivel de tolerancia a fallos para un archivo de 1 Gigabyte (GB) se puede conseguir utilizando sólo 1,5 GB de espacio de almacenamiento. Como resultado, se espera que esta característica cambie de forma significativa el coste total de propiedad para el uso de Hadoop , empleando la codificación de borrado para reducir el coste de almacenamiento en comparación con la replicación, manteniendo una alta tolerancia a fallos [155] [228].

Además de lo mencionado previamente, hay que añadir que existen tecnologías que pueden ser explotadas para mejorar el rendimiento y optimización de los recursos en un entorno de procesamiento de imágenes médicas . Por ejemplo, el plugin para el almacenamiento tolerante a fallos usando la biblioteca Backblaze de código abierto Reed-Solomon, y los ficheros de imagen DICOM a través del sistema Dicoogle PACS que se presenta en esta investigación [241] [174] [225].

Teniendo en cuenta todo lo dicho anteriormente, el objetivo fundamental de la presente investigación es proponer una solución para maximizar el rendimiento de codificación y decodificación de las imágenes médicas basado en el paralelismo computacional, lo cual se hace para garantizar la recuperación de las imágenes en el sistema de configuración global. En esta investigación, se hace una fusión de las tecnologías antes descritas y, para el proceso de codificación y decodificación, se utiliza el algoritmo Reed-Solomon.

4.2. Dicoogle PACS

Dicoogle es un sistema de comunicación y almacenamiento de imágenes PACS de código abierto; dicho código es de libre acceso y se puede obtener en [49], este fue desarrollado por UA. PT BIOINFORMATICS [213] y BMD Software [212]. Esta herramienta se utiliza en tres tipos de casos: producción, investigación y docencia. La arquitectura modular permite el rápido desarrollo de nuevas funcionalidades debido a la disponibilidad de un kit de desarrollo de software (SDK) [117].

El proyecto Dicoogle permite realizar consultas sobre un conjunto de repositorios distribuidos que se indexan lógicamente como una sola unidad ágil. En [223], los autores describieron un servicio de transmisión basado en la nube [231] que actúa como un enlace de comunicación entre diferentes instituciones, lo que permite a la comunidad acceder, compartir y descubrir registros de imágenes [223].

En [207] autores propusieron una arquitectura de archivos PACS basados en la nube que proporciona privacidad, integridad y disponibilidad de datos. La alternativa propuesta fue independiente del proveedor de la nube y los módulos core [231]. Las métricas operativas para varias modalidades de imágenes médicas se tabularon y compararon para Google Storage, Amazon S3 y LAN PACS [207].

En [188] se describió el proceso de desarrollo y validación del sistema Dicoogle en dos centros de salud. Este sistema es un nuevo enfoque capaz de recopilar e indexar información de diferentes archivos PACS, desarrollado sobre Dicoogle, que permite la construcción de múltiples vistas sobre repositorios de datos según el estándar DICOM de forma flexible y rápida. La metodología desarrollada en [188] aporta a la mejora del uso de metadatos DICOM almacenados en los PACS dispersos de los Departamentos de Radiología.

Otra implementación basada en Dicoogle es dónde se propuso la arquitectura e implementación de un sistema de recuperación de imágenes

basado en contenido perfilado basado en consultas (CBIR) [219]. La alternativa propuesta por los autores fue que los perfiles CBIR permiten la especificación de funciones de distancia como del conjunto de características que deben estar presentes para que esa funciones se ejecute.

Actuales investigaciones mencionan a Dicoogle como un marco de software que permite a desarrolladores e investigadores crear prototipos e implementar ágilmente nuevas funciones, optimizando los servicios integrados de comunicaciones e imágenes digitales en la medicina DICOM [220]. La implementación mencionada completa un archivo PACS el cual es altamente extensible debido a su arquitectura basada en complementos y su funcionalidad lo que permite la exploración de grandes conjuntos de datos DICOM y metadatos asociados [189].

En [168] se presenta la arquitectura de software de Dicoogle basada en diferentes complementos, que incluye un complemento de almacenamiento con dos enfoques, uno dirigido al almacenamiento local y otro al almacenamiento en la nube [231]. Además, la arquitectura del software Dicoogle es compatible con un sistema de indexación basado en documentos y protocolos P2P (Peer to Peer). Además, en [73] se propone una arquitectura de un PACS de patología web totalmente compatible con los formatos de comunicación y datos estándar DICOM. La alternativa incluye un archivo PACS responsable de almacenar datos de imágenes de diapositivas completas en formato DICOM para imágenes de diapositivas completas (WSI) y proporciona una interfaz de comunicación basada en los últimos servicios web DICOM. El otro componente es un visor de huella cero que se ejecuta en cualquier navegador web. Consume datos utilizando servicios web de archivo PACS estándar [73].

En [119], el mecanismo de registro seguro para repositorios de imágenes médicas fue diseñado e instanciado como una extensión de un conocido archivo de código abierto. Se implementó una nueva capa de servicios web para brindar una solución a los protocolos DICOM Web para almacenar,

buscar y recuperar datos de imágenes médicas.

En [118], se describe el entorno Dicoogle, con especial énfasis en su paquete de aprendizaje, los recursos disponibles y el impacto de la plataforma en la investigación y la academia. En [118], los autores comenzaron presentando una descripción general de su concepto arquitectónico, la investigación más reciente respaldada por Dicoogle, algunas observaciones obtenidas de su uso en la enseñanza y estadísticas sobre el uso del software en todo el mundo. Además, en [118] se presenta una comparación entre la plataforma Dicoogle y los PACS de código abierto más populares del mercado. Además, se discuten las soluciones actuales al problema del almacenamiento y la gestión de imágenes médicas. Además, se presenta una solución basada en una red privada de nodos para superar problemas como privacidad, redundancia y alta disponibilidad.

4.3. Dicoogle SDK

Dicoogle SDK (Software Development Kit) fue diseñado para simplificar el desarrollo de nuevas funcionalidades de módulos de uso específico que son desarrollado por otros investigadores. El objetivo de dichos investigadores es garantizar la compatibilidad de nuevos desarrollos SDK. Para desarrollar un nuevo módulo funcional, los desarrolladores deben mover el paquete que constituye al directorio de complementos de Dicoogle. Luego, Dicoogle genera automáticamente los nuevos módulos al inicio. El SDK de Dicoogle analiza que todas las operaciones basadas en almacenamiento, las consultas y la indexación estén disponibles a través de su interfaz de programación de aplicaciones interna (API) [117].

La arquitectura general de Dicoogle (véase Fig. 4.1) reemplaza la base de datos relacional tradicional con un proceso ágil en la indexación y recuperación. Dicoogle fue diseñado para extraer, indexar y almacenar todos los metadatos presentados en archivos DICOM de estudios de pacientes,

incluidas etiquetas privadas [117].

4.4. Mercado PACS

El mercado de PACS es heterogéneo, con varios proveedores de PACS que han implementado instalaciones en centros de atención médica. Las interfaces de consulta y recuperación DICOM proporcionadas por PACS tienen múltiples variaciones relacionadas con las clases SOP (par de objetos de servicio) implementadas, sintaxis de transferencia, negociaciones extendidas, atributos y tipos de coincidencia. Estas variaciones pueden integrar un nuevo consumidor DICOM con un complejo PACS y hacer que consuma mucho tiempo.

Aunque la mayoría de los productos PACS proporcionan una declaración de conformidad con DICOM como una descripción de sus capacidades de consulta y recuperación, no existe un análisis colectivo que describa las diversas capacidades de consulta, recuperación y variaciones de los sistemas de imágenes [21]. Además, otro aspecto importante se refiere a la seguridad del sistema. Con respecto a esto, desde una perspectiva práctica, Las medidas de seguridad específicas de PACS deben considerarse junto con las medidas aplicables a la infraestructura de tecnología de la información (TI). El propósito de esto es fortalecer la seguridad de los sistemas PACS a los que se puede acceder desde Internet [54].

4.5. Erasure Code Reed-Solomon

Reed-Solomon (RS) es un algoritmo de código de borrado con propiedades necesarias para el almacenamiento de archivos y la recuperación confiable de los mismos [98] [56] [137]. Es simple de implementar a la vez que es una técnica fehaciente y que garantiza recuperar un elemento de datos completo, incluso cuando parte o partes del archivo de datos almacenado

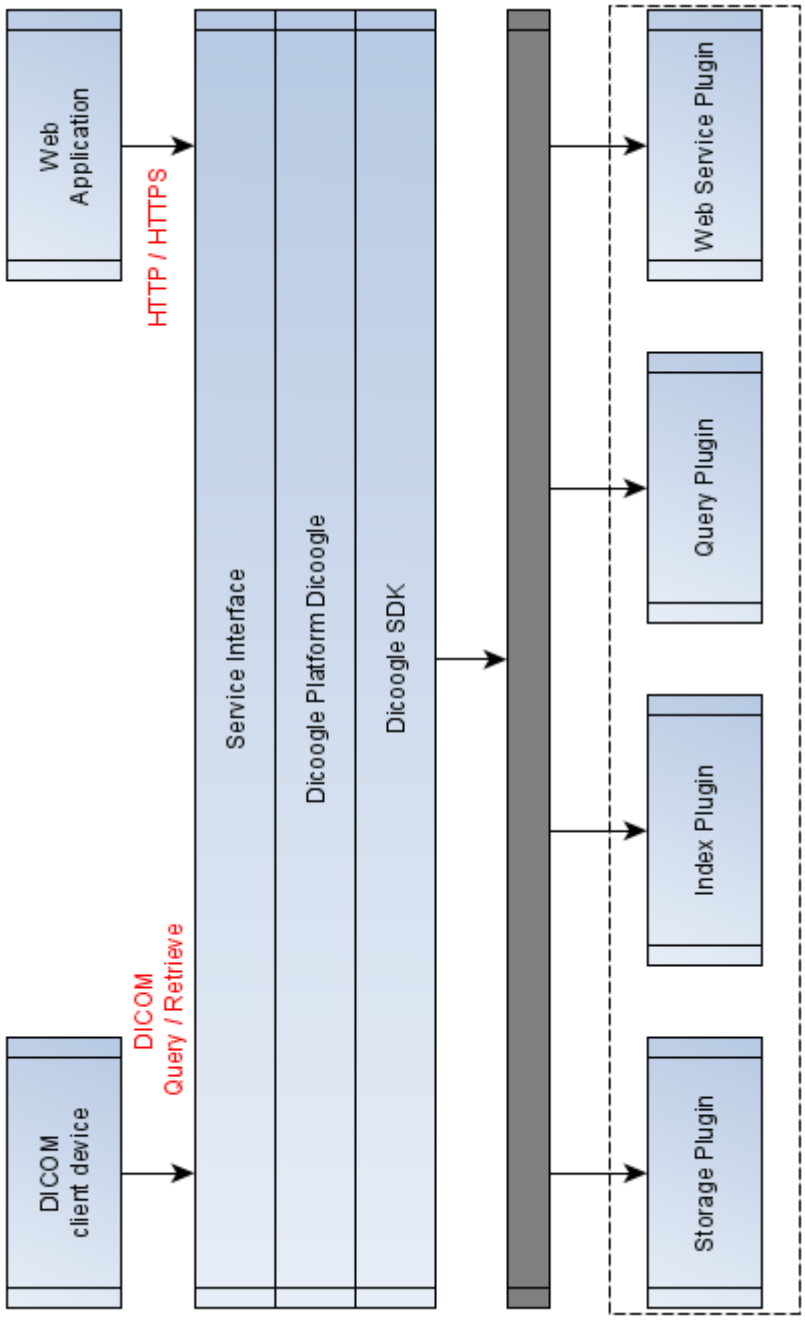


Figura 4.1: *A*rquitectura de Dicoogle PACS

original se pierdan o no estén disponibles. En [171] se realiza un estudio del funcionamiento de los procesos de codificación y de decodificación del algoritmo de código de borrado. En esencia un sistema de almacenamiento con código de borrado que codifica k bloques o discos de datos en m bloques o discos, cuando fallan hasta m bloques o discos de datos, su contenido se decodifica a partir de los bloques o discos disponibles para recuperar el total de información (véase. Fig. 4.2).

Los algoritmos RS son usados en sistemas de almacenamiento distribuido y proporcionan una implementación eficiente y una alta capacidad de tolerancia a fallos para un nivel de redundancia dado [185]. Los sistemas de almacenamiento han crecido hasta el punto de que los fallos son inevitables, y quienes diseñan los sistemas deben planificar con anticipación para no perder información cuando se producen fallos. La tecnología principal para proteger los datos frente a los fallos es la codificación de borrado, que tiene una historia de más de 50 años [171].

En [133], se realiza un análisis de la replicación del código de borrado y definen algunas situaciones en las que se prefiere la replicación de archivos completos. Además, en [133] se estudia y caracteriza el punto de cambio de preferir la replicación de archivos completos o la replicación del código de borrado. En [133] también se discuten las consideraciones adicionales en la construcción de sistemas de replicación de código de borrado.

Tal como se menciona en [166], en los últimos años se ha utilizado diversas versiones de la tecnología matriz redundante de discos independientes (RAID) con el fin de proporcionar la redundancia de archivos y la confiablidad necesaria. La codificación de borrado es una práctica estándar para los sistemas que almacenan datos de manera confiable, y muchos de ellos utilizan la codificación Reed-Solomon.

En base a lo anterior, los sistemas de almacenamiento centralizados y en red han crecido hasta el punto en que la tolerancia a fallos proporcionada por RAID 5 ya no es suficiente. Además, los sistemas de almacenamiento

RAID 6 protegen k discos de datos con dos discos de paridad para que el sistema de $k + 2$ discos puedan tolerar la falla de dos discos cualesquiera [170].

Por otro lado, el sistema RAID [166] integrado en Linux usa Reed-Solomon y tiene una implementación Reed-Solomon cuidadosamente ajustada en un lenguaje de alto nivel C que forma parte del módulo RAID. También, el servicio de computación en la nube, Microsoft Azure y el sistema de almacenamiento de Facebook utilizan Reed-Solomon [221].

En la literatura [203] y [202], se han encontrado diversas propuestas de esquemas de EC basados en hardware para aprovechar las capacidades avanzadas de cálculo de los centros de datos modernos. Además, la librería Backblaze Reed-Solomon como Open Source [18] tiene una licencia de software que se creó en el Instituto Tecnológico de Massachusetts (MIT). Esta es una licencia de software libre que se puede usar en proyectos libres y comerciales.

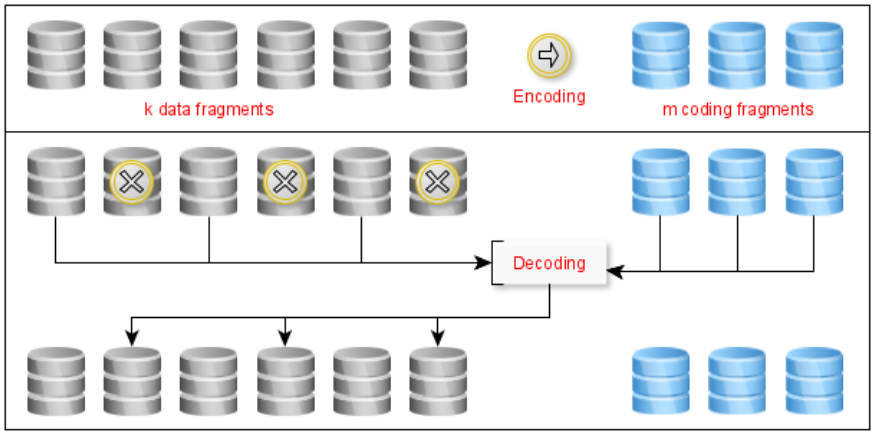


Figura 4.2: Código de borrado k = datos, m = pariedad.

4.6. Implementación de propuesta

En esta sección, se describe dos pasos para el proceso de configuración. En primer lugar, se describe la configuración del hardware para las pruebas realizadas. En segundo lugar, se describe la configuración de software del plugin de almacenamiento nativo implementado en Dicoogle PACS [119]. En el caso del software se usó el algoritmo Reed-Solomon de la biblioteca Backblaze [18], donse se asignan valores para k y m según la configuración de codificación y decodificación. En esta configuración (véase. Fig. 4.3) se observan los elementos:

- Almacenamiento conectado a la red (NAS) de las imágenes médicas.
- Modalidad médica como elemento generador de imágenes.
- Estación de trabajo A y Estación de trabajo A y B como dispositivos para visualización de imágenes generadas por la modalidad médica y almacenadas en el NAS (Network Attached Storage).

4.6.1. Hardware

La evaluación se ha realizado en un sistema de almacenamiento conectado a la red NAS HPE StoreEasy 1660 HPE PN. Q2P71A, con procesador Intel® Xeon-Silver® 4112 CPU @ 2,59 GHz, memoria RAM de 128GB, 5100 PRO M.2 SSD 240GB x 2 SATA 6 Gb/s HPE PN. 871627-001 y unidad de almacenamiento HPE 24 TB SAS LFF Smart Carrier 4-pack HDD Bundle HPE PN. Q2P82A. La velocidad de rotación de disco duro de 7.200 Revoluciones por Minuto (RPM).

Para la conectividad de la arquitectura se ha usado un Switch Cisco® Catalyst® 2960-XR Gigabit Ethernet, y para la conectividad inalámbrica un Access Point Cisco Aironet AIR-CAP1702I-A-K9 aplicando el protocolo IEEE 802.11n a la frecuencia 2,4 GHz.

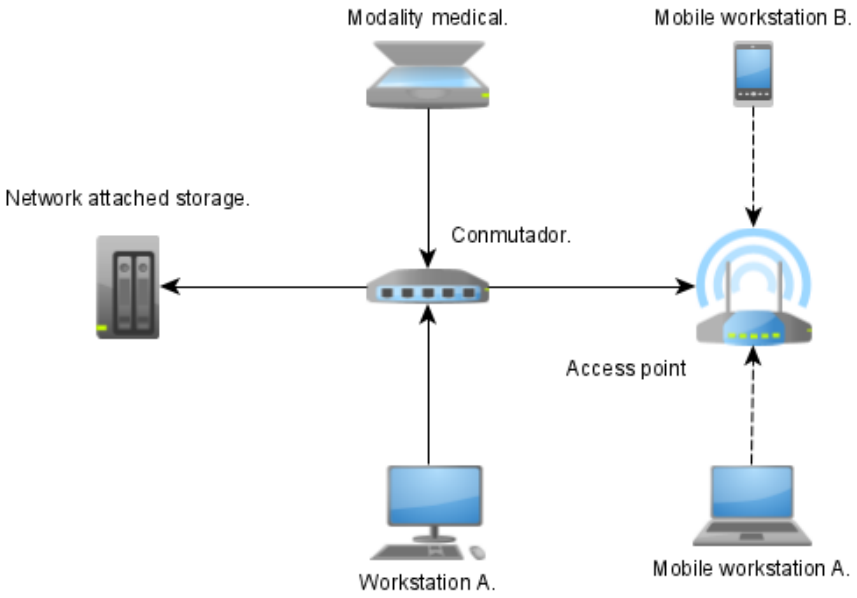


Figura 4.3: Arquitectura utilizada para validar la propuesta.

Por otro lado, se han realizado pruebas con ficheros DICOM usando el FDR Smart X [208], el cual es el nuevo sistema de Rayos X de Fujifilm. Así mismo, en esta investigación se usó un Tomógrafo marca Toshiba modelo TSX-031A, ActivionTM16. Este equipo, usa el sistema de Tomografía Computarizada (CT) helicoidal de 16 cortes que admite la exploración de todo el cuerpo humano [146]. El sistema genera un mínimo de 32 cortes por cada 1,5 segundos utilizando el detector multifilar de espesor de corte seleccionable (SSMD). Respecto a la arquitectura de la Fig. 4.3, Workstation A y Mobile workstation A y B corresponden a dispositivos para visualizar las imágenes DICOM.

4.6.2. Software

La propuesta principal en este trabajo radica en la implementación de un plugin para código de borrado a través del algoritmo Reed-Solomon de la biblioteca Backblaze. En comparativa con la arquitectura de Dicoogle PACS [168], en la Fig. 4.4 se observa la ubicación del plugin implementado en la propuesta.

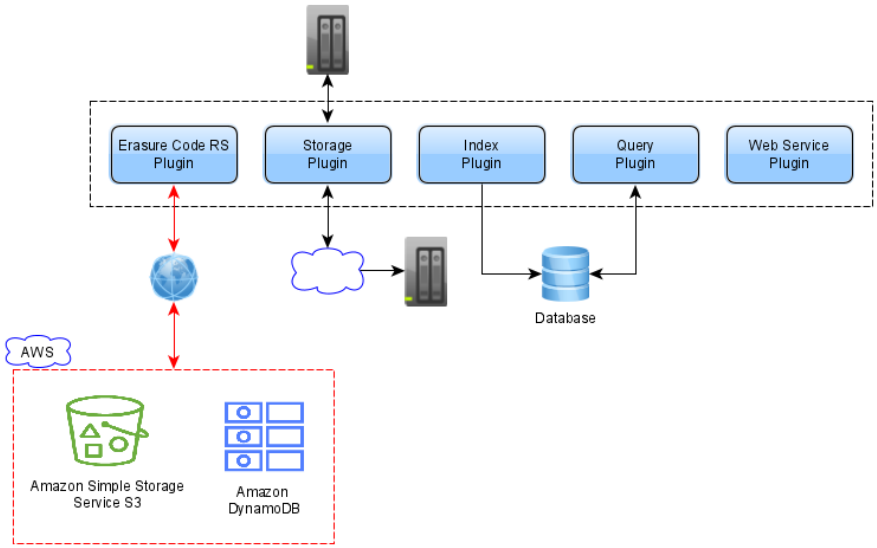


Figura 4.4: Plugin de código de borrado en Dicoogle PACS.

Una vez evaluada la biblioteca Backblaze Reed-Solomon, se implentó un plugin nativo en la capa de almacenamiento de Dicoogle PACS con el fin de garantizar tolerancia a fallos. En la Fig. 4.5 se muestra el esquema principal del algoritmo.

Para el desarrollo se usó el lenguaje Java y se usaron hebras con el fin de mejorar el rendimiento del algoritmo. A continuación se mencionan las características del desarrollo:

- Los ficheros utilizados en el plugin de almacenamiento fueron de

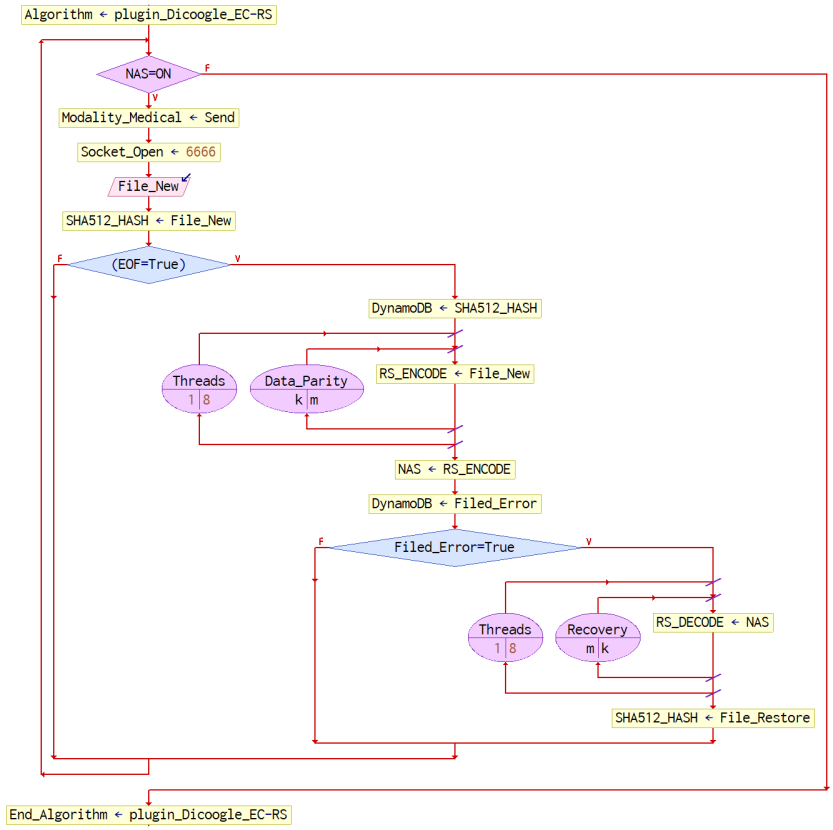


Figura 4.5: Algoritmo del plugin de código de borrado en Dicoogle PACS.

extensión *.dcm*, los cuales corresponden al formato de imágen médica DICOM. Los ficheros se producen desde los dispositivos de Rayos X Fujifilm FDR Smart X y el Tomógrafo Toshiba modelo TSX-031A.

- En cuanto a los servicios y complementos, es posible iniciar y/o detener los servicios en ejecución en tiempo real. Por esta razón, se define un puerto para su uso en Dicoogle PACS.

- Para cada fichero DICOM se usa una función hash SHA-512. Esto se

hace para llevar a cabo la verificación del hash SHA-512 con el fin de evitar la duplicidad de información mediante una estructura de datos.

- El hash SHA-512 se almacena en una base de datos NoSQL DynamoDB. La implementación de DynamoDB en el plugin de almacenamiento fue suministrada como un archivo *.jar* ejecutable java. La propuesta puede ejecutarse en diferentes sistemas operativos como Windows, Linux, macOS y otras plataformas compatibles con Java.

- La protección que ofrece la codificación de borrado puede representarse en forma simple mediante la siguiente ecuación: $n = k + m$, dónde, k es la cantidad original de datos y m representa los datos redundantes agregados para proporcionar protección contra los fallos a través de código de borrado.

- Backblaze Reed-Solomon, es una biblioteca de codificación de borrado para calcular la tolerancia a fallos y luego usarla para reconstruir archivos. Cuando un fichero DICOM ha sido almacenado en Dicoogle PACS, el plugin de almacenamiento puede dividir en 4, 6 y 10 fragmentos del mismo tamaño para cada fichero DICOM. Este proceso se denomina encode. Posteriormente, se crean 2, 3 y 5 fragmentos adicionales que mantienen la paridad respectivamente. Esto da como resultado un total de 6, 9 y 15 fragmentos para cada configuración (es decir, 4, 6, 10 fragmentos). Con esta configuración, es posible reconstruir el fichero original a partir de 4, 6 y 10 fragmentos. A este proceso se denomina decode de los 6, 9 y 15 fragmentos.

4.7. Resultados

Como parte de la propuesta que se plantea en esta investigación, el entorno construido ha precisado de la configuración del plugin de

almacenamiento del código de borrado Reed-Solomon implementado en Dicoogle PACS. En este sentido, se realizaron dos tipos de experimentos empleando encode y decode. En el proceso de encode, fueron usadas configuraciones en relación al tamaño de bloque de datos originales k y del tamaño de bloques de paridad m para la recuperación de cualquier bloque perdido. En cuanto al proceso de decode, la recuperación de datos se realiza desde los bloques de paridad m para garantizar la fiabilidad. Los principales resultados para ambos procesos se presentan en las figuras de esta sección. Dónde, en todos los casos, el eje x representa el tamaño de bloque de datos en Megabytes (MB) y el eje y, representa el consumo de tiempo en milisegundos para cada tamaño de bloque. Para validar la recuperación de bloques perdidos, se ha inducido a provocar fallos en la configuración para obtener los tiempos y consumo de recursos (memoria y procesador) en la transmisión/recepción de las imágenes DICOM. Así mismo, en todas las configuraciones se usan de 1 a 8 hebras. Explicado lo anterior, a continuación se presentan diferentes experimentos con configuraciones respectivas.

Experimento 1.- En esta prueba se usaron ficheros DICOM de diversos tamaños (véase. Tabla 4.1) que contienen parte de la extremidad inferior del cuerpo humano. Los resultados de esta configuración se observan en la Fig. 4.6. En esta figura, se puede observar el tiempo consumido por cada hebra en una relación a 4 bloques de datos y 2 bloques de paridad (es decir, RS(4,2)) para recuperación de datos completos en caso de pérdida de 1 o 2 bloques de datos. En el caso de la codificación, se observó un consumo aproximadamente de 0,9 segundos utilizando 8 hebras con el grupo de ficheros más significativo.

Experimento 2.- En la Fig. 4.7 se hace referencia a los resultados de tolerancia de 2 fallos del grupo de ficheros DICOM del experimento 1. En esta figura, se observa que el proceso de decode consume aproximadamente 0,8 segundos cuando se usan 8 hebras para la recuperación de 1 o 2 bloques de datos.

Experimento 3.- En esta prueba, se usó un segundo grupo de ficheros DICOM (véase. Tabla 4.2). En la Fig. 4.8, se muestra el tiempo que ha consumido la configuración RS(6,3) en función del número de hebras para el encode. En este caso, se observa que el consumo de tiempo se aproxima a 1,5 segundos usando 8 hebras para el tamaño máximo del bloque de datos original de 1024 MB.

Experimento 4.- En la Fig. 4.9, se hace referencia a los resultados de tolerancia de 1 a 3 fallos del grupo de ficheros DICOM del experimento 3. Se observa que el proceso de decode consume aproximadamente 1,2 segundos cuando se usan 8 hebras para la recuperación de 1 a 3 bloques de datos.

Experimento 5.- En esta prueba, se usó un tercer grupo de ficheros DICOM (véase Tabla 4.3). En la Fig. 4.10, se muestra el tiempo consumido para la configuración RS(10,5) en función del número de hebras para el encode. Esta configuración permite pérdida de 1 y 5 bloques de datos del fichero DICOM completo. En este caso, se observa que el consumo de tiempo se aproxima a 2,5 segundos usando 8 hilos para el tamaño máximo de bloque de 1024 MB.

Experimento 6.- Finalmente, en la Fig. 4.11, se muestran los resultados de tolerancia de 1 a 5 fallos del grupo de ficheros DICOM del experimento 5. Se observa que el proceso de decode consume aproximadamente 1,9 segundos cuando se usan 8 hebras para la recuperación de 1 a 5 bloques de datos.

Tomando como referencia los planteamientos presentados en el **experimento. 1**, estos permiten reducir el tiempo de procesamiento de datos para $k = 4$ y redundancia $m = 2$, alcanzando reducciones tanto en encode de 0,9 segundos y decode de 0,8 segundos. Por otro lado, el uso de diferentes hebras bajo las mismas características ha sido establecido para todos los casos $k = 4, 6, 10$ y $m = 2, 3, 5$, revelando una eficiencia de almacenamiento del 66,66 %.

Es importante mencionar que en el estudio realizado en [105] se utilizaron valores de $k = 2, 4, 8$ y $m = 2, 2, 2$ respectivamente. Así mismo, en [1]

Tabla 4.1: Tamaño del archivo utilizado en el entorno de prueba para codificar k=4, m=2.

Tamaño original	**Tamaño del fragmento**	**Tamaño de redundancia**
2 MB	512 KB	3 MB
4 MB	1 MB	6 MB
8 MB	2 MB	12 MB
16 MB	4 MB	24 MB
32 MB	8 MB	48 MB
64 MB	16 MB	96 MB
128 MB	32 MB	192 MB
256 MB	64 MB	384 MB
512 MB	128 MB	768 MB
1024 MB	256 MB	1536 MB

utilizaron $k = 6, 10$ y $m = 3, 4$ proporcionalmente. Además, en [217] emplearon en sus casos de estudio $k = 8, 9, 10$ y $m = 2$; $k = 8, 9, 10$ y $m = 3$ y $k = 8, 9, 10$ y $m = 4$ equitativamente. Por último, en [62] aplicaron $k = 12$ y $m = 4$. En toda la bibliografía citada se logran obtener tiempos de encode y decode en diferentes escenarios del algoritmo Reed-Solomon. Por tanto, en base a los resultados, consideramos que nuestro planteamiento aplicado a un sistema de almacenamiento de imágenes médicas tal como Dicoogle PACS es óptimo en cuanto a tiempos de procesamiento.

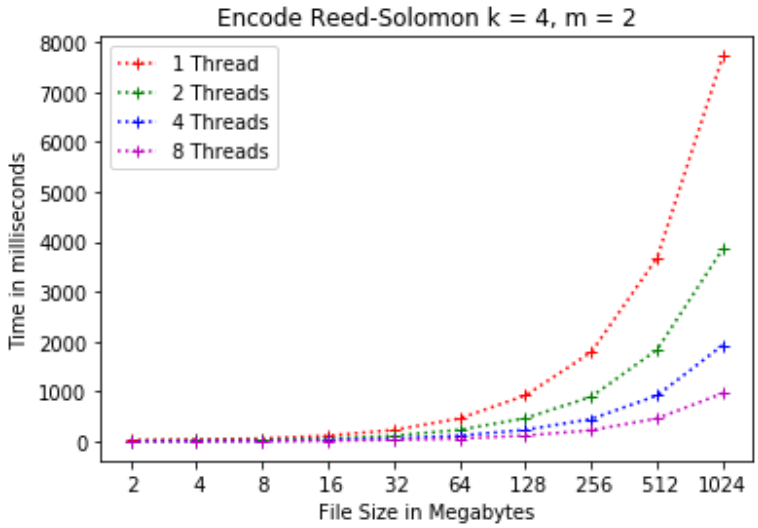


Figura 4.6: Configuración de la codificación, $RS(4, 2)$.

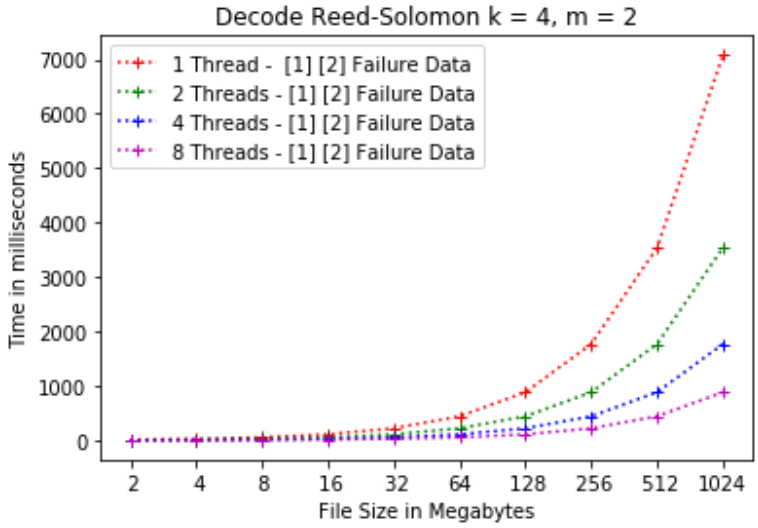


Figura 4.7: Configuración de la decodificación, $RS(4, 2)$.

Tabla 4.2: Tamaño del archivo utilizado en el entorno de prueba para codificar k=6, m=3.

Tamaño original	**Tamaño del fragmento**	**Tamaño de redundancia**
2 MB	341.33 KB	3 MB
4 MB	682.00 KB	6 MB
8 MB	1.33 MB	12 MB
16 MB	2.66 MB	24 MB
32 MB	5.33 MB	48 MB
64 MB	10.66 MB	96 MB
128 MB	21.33 MB	192 MB
256 MB	42.66 MB	384 MB
512 MB	85.33 MB	768 MB
1024 MB	170.66 MB	1536 MB

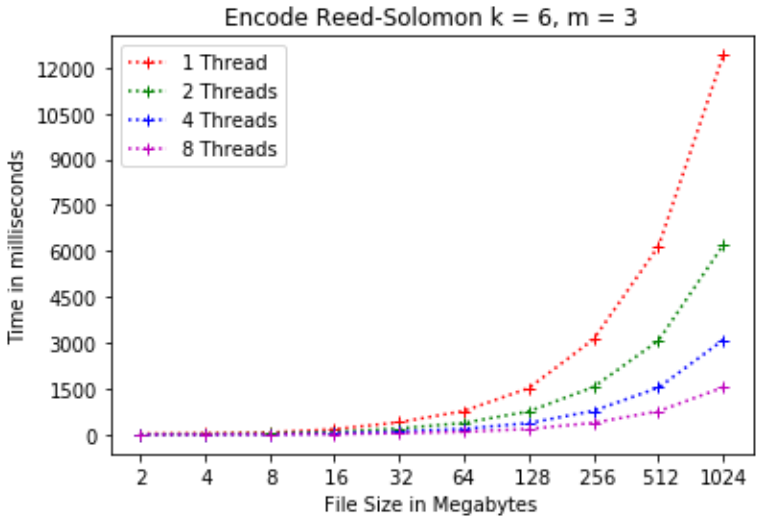


Figura 4.8: Configuración de la codificación, $RS(6, 3)$.

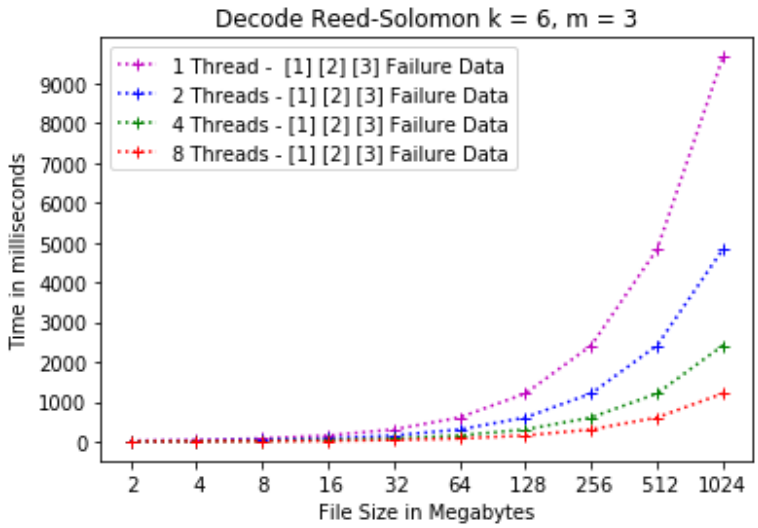


Figura 4.9: Configuración de la decodificación, $RS(6, 3)$.

Tabla 4.3: Tamaño del archivo utilizado en el entorno de prueba para codificar k=10, m=5.

Tamaño original	**Tamaño del fragmento**	**Tamaño de redundancia**
2 MB	204.80 KB	3 MB
4 MB	409.60 KB	6 MB
8 MB	819.20 KB	12 MB
16 MB	1.60 MB	24 MB
32 MB	3.20 MB	48 MB
64 MB	6.40 MB	96 MB
128 MB	12.80 MB	192 MB
256 MB	25.60 MB	384 MB
512 MB	51.20 MB	768 MB
1024 MB	102.40 MB	1536 MB

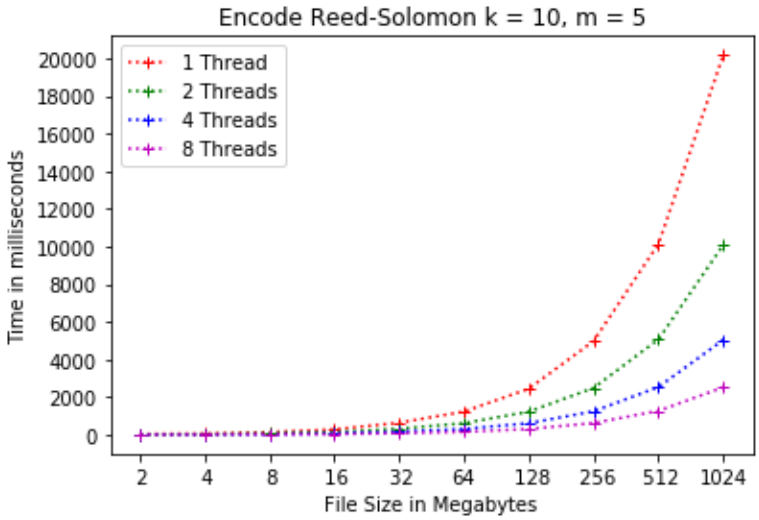


Figura 4.10: Configuración de la codificación, $RS(10, 5)$.

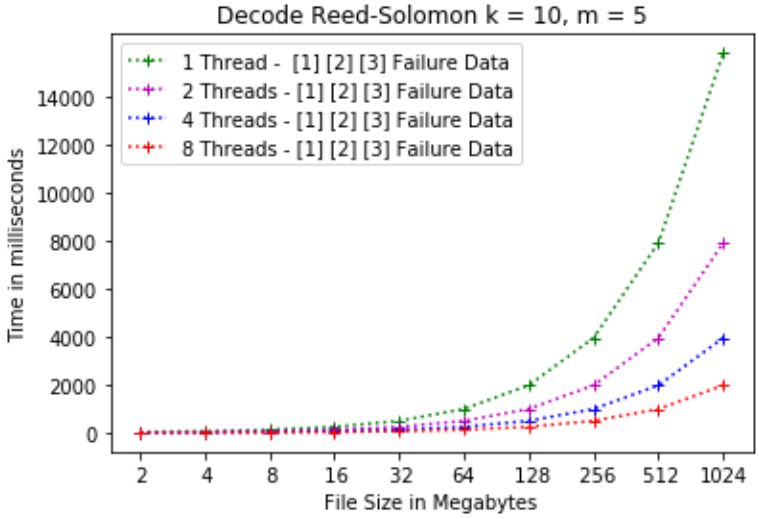


Figura 4.11: Configuración de la decodificación, $RS(10, 5)$.

Capítulo 5

Compresión y código de borrado: aplicado a imágenes dermatológicas.

Este capítulo está enfocado en la cuasiperiodicidad de los componentes principales (PCs, Principal Components) en la descomposición en PCs de imágenes médicas, que representan manifestaciones dermatológicas en pacientes paucisintomáticos de COVID-19. Para ello, se tomó un conjunto de fotografías de uno de los patrones más frecuentes en COVID-19, el patrón maculopapular, caracterizado por una erupción eritematopapular, y se realizó la compresión de una de las imágenes médicas. Dicha compresión se realizó de diferentes formas: 1) utilizando dos PCs, 2) utilizando tanto un PC periódico como un PC no periódico, 3) utilizando dos PCs periódicos, 4) utilizando un único PC, y 5) utilizando un único PC periódico. Además, se implementó una batería de pruebas para evitar falsos positivos o negativos en el diagnóstico de la enfermedad. Esta batería contiene el código de borrado Reed-Solomon de la biblioteca zfec. Los resultados de esta investigación demostraron que es posible trabajar con reconstrucciones

aceptables de imágenes comprimidas en el campo de la dermatología, sin perder la calidad y características que permiten llegar a un diagnóstico correcto. Además, los tiempos de procesamiento en la recuperación de archivos JPG vs PNG son óptimos, y el logro anterior permite clasificar correctamente muchas enfermedades sin temor a equivocarse.

5.1. Introducción

La COVID-19 es una enfermedad vírica con muchas manifestaciones clínicas, entre las cuales las manifestaciones dermatológicas llaman la atención por su alta incidencia de aparición [145] [48] [180] [211] [25]. Además, a veces los signos y síntomas de la COVID-19 son manifestaciones dermatológicas en pacientes sin manifestaciones respiratorias [100] [28]. Por ello, en estos momentos de pandemia, es de suma importancia para la medicina y especialmente para la especialidad de dermatología disponer de imágenes que permitan el diagnóstico de las enfermedades [242] [164] [46] [4] [7] .

En la actualidad, la adquisición y el procesamiento de imágenes médicas constituyen un área de investigación muy importante. Además, la reconstrucción de imágenes médicas ha encontrado muchas aplicaciones tanto en medicina como en la industria . En resumen, en la actualidad los médicos comparten grandes volúmenes de imágenes, muchas de las cuales se toman y guardan para el diagnóstico. Por ello, los médicos necesitan tecnologías robustas de interpretación y análisis de imágenes que eviten pérdidas de calidad y características, y que faciliten la identificación de patrones de comportamiento del organismo humano que sean manifestaciones de posibles contagios de enfermedades [138] [90] [141] [134] [232] [172] [173] [233] [19] [122] [144] [113] [224].

Sin embargo, para cumplir con éxito lo anterior, las imágenes médicas necesitan grandes cantidades de bits para ser almacenadas, en particular las

de alta resolución. Además, para la correcta transmisión de estas imágenes, las redes no pueden disponer de un ancho de banda pequeño y limitado. Por lo tanto, el elevado consumo de ancho de banda y la creciente demanda de almacenamiento de gran cantidad de información, hacen necesario contar con procedimientos que permitan una compresión eficiente a bajo bitrate de las imágenes médicas. Dicho esto, salen a la luz cuestiones a tener en cuenta a la hora de llevar a cabo la transmisión rápida y eficiente de estas imágenes.

Lo mencionado en el párrafo anterior implica realizar un proceso de compresión. En definitiva, hay que reducir la cantidad de datos para representar las imágenes de interés objeto de estudio, eliminando aquellos datos que no aportan información relevante sobre el contenido de dichas imágenes. Además, si bien las pérdidas debidas a la degradación de la imagen son tolerables, lo que se pierde es insignificante comparado con lo que se gana debido a la disminución del tamaño del archivo. Así, las pérdidas debidas a la degradación de la imagen se compensan con lo que se gana al reducir el tamaño de los ficheros que las contienen.

En este trabajo, la técnica de Análisis de Componentes Principales (PCA) se utiliza para llevar a cabo la reducción de la dimensión de imágenes médicas que representan manifestaciones dermatológicas en pacientes paucisintomáticos de COVID-19. El objetivo principal de este trabajo es aplicar PCA a la compresión de un conjunto de fotografías de uno de los patrones más frecuentes en COVID-19, el patrón maculopapular o morbiliforme, que se caracteriza por una erupción eritematopapular. La razón de ser de esta investigación es tratar de utilizar técnicas de compresión con pérdidas en imágenes médicas con vistas a resolver el problema de que es muy difícil estudiar imágenes médicas comprimidas para diagnosticar una enfermedad [95] [47] [58] [20] [63] [72] [22] [101] [84] [82] [83] [85] [81] [3].

Para los dermatólogos, la calidad de la asistencia depende de la visualización correcta y detallada de las lesiones que permiten un examen

exhaustivo del paciente para poder dar un diagnóstico preciso. En el caso de las enfermedades cutáneas, desde el cáncer hasta las patologías víricas y microbianas, el diagnóstico por imagen se ha convertido en una herramienta esencial de investigación y diagnóstico. En concreto, este tipo de diagnóstico es un instrumento preciso, altamente sensible y fiable, que puede reducir el número de biopsias innecesarias, la morbilidad asociada y los elevados costes de la asistencia. Además, mejora la tasa de supervivencia.

Por lo tanto, en el supuesto e hipotético caso de que tras aplicar algoritmos de compresión con pérdida, los médicos no puedan realizar un diagnóstico correcto de la enfermedad a partir de la imagen comprimida, se incluirá en nuestro estudio la conversión de formatos. Es decir, los archivos originales de tipo JPG (es decir, compresión con pérdidas), se migrarán a formato PNG (sin pérdidas). Y la razón de lo anterior es el hecho de que si hay una especialidad médica en la que la fotografía digital juega un papel fundamental es, sin duda, la dermatología. Dicho esto, con el fin de garantizar la integridad de los archivos JPG y PNG, en este trabajo se aplicará tolerancia a fallos a dichos archivos. Para ello, hemos incorporado la codificación de borrado (EC). Se trata de un método de protección de datos en el que los datos se dividen en fragmentos, se expanden y se cifran con fragmentos de datos redundantes, y se almacenan en un conjunto de ubicaciones diferentes, como: discos, nodos de almacenamiento o ubicación geográfica.

El código de borrado es una tecnología utilizada principalmente en sistemas distribuidos para mejorar la fiabilidad del almacenamiento. En comparación con la replicación multicopia, el código de borrado puede lograr una mayor fiabilidad de los datos con menos redundancia de datos, pero el método de codificación es más complejo y requiere muchos cálculos [62]. Además, el código de borrado es una tecnología de codificación que puede añadir k fragmentos de datos originales a m fragmentos de datos de paridad, y con ello se puede recuperar el archivo original a partir de $k + m$ fragmentos

[105] [1] [217]. Es decir, si cualquier fragmento de datos cuya paridad sea menor o igual que m es inválido, entonces el archivo original aún puede recuperarse a partir de los datos restantes.

Actualmente, existen tres tipos principales de tecnologías EC en los sistemas de almacenamiento distribuido: 1) códigos de borrado de matriz (código de matriz: RAID 5, RAID 6, etc.) [181], 2) códigos de borrado Reed-Solomon (RS), y 3) código de comprobación de paridad de baja densidad (LDPC) [152] [230]. El código de borrado RS [56] [98] [137] es un algoritmo con las propiedades necesarias para el almacenamiento y recuperación fiable de archivos. Se trata de una técnica fiable y de implementación sencilla. Además, utilizando esta técnica, se garantiza la recuperación de un elemento de datos completo. Es más, esta garantía se mantiene incluso en escenarios en los que una parte o partes del archivo de datos que se almacenó originalmente se pierden o no están disponibles.

En [171] [147], se realiza un estudio del funcionamiento de los procesos de codificación y decodificación del algoritmo de código de borrado Reed-Solomon. En resumen, [171] [147] propone un sistema de almacenamiento EC que codifica k fragmentos o discos de datos en m fragmentos o discos de datos, cuando fallan hasta m fragmentos o discos de datos. Su contenido se decodifica a partir de los fragmentos o discos disponibles para recuperar la información completa (véase Fig. 5.1).

Los algoritmos de código de borrado RS se utilizan en sistemas de almacenamiento distribuido, proporcionando una implementación eficiente y una alta capacidad de tolerancia a fallos para un determinado nivel de redundancia [185]. Los sistemas de almacenamiento han crecido hasta tal punto que los fallos son inevitables, y los diseñadores de sistemas deben planificar con antelación para que no se pierda información cuando se produzcan fallos. La principal tecnología para proteger los datos de los fallos son los códigos de borrado, que tiene una historia de más de 50 años [171].

En [203] y [202], se han encontrado varias propuestas de esquemas

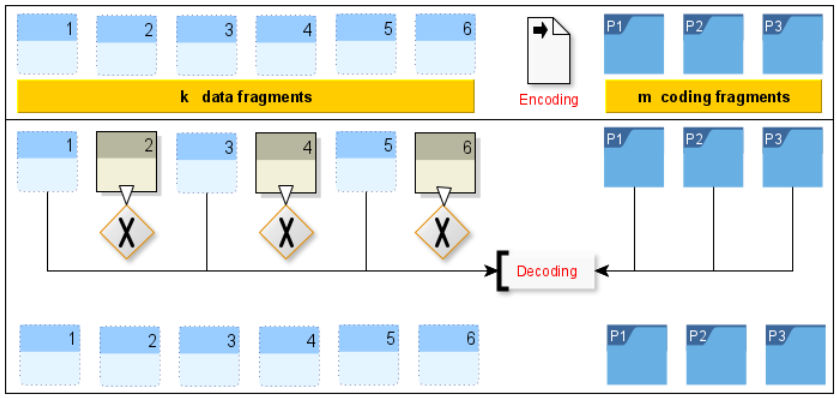


Figura 5.1: Un sistema de almacenamiento con codificación de borrado que codifica k discos de datos y m discos de tolerancia.

EC basados en hardware para aprovechar las avanzadas capacidades computacionales de los centros de datos modernos. La biblioteca zfec ofrece una implementación rápida, portátil y programable de EC [246]. Genera bloques redundantes de información de tal forma que si algunos de los bloques se pierden, los datos originales pueden recuperarse a partir de los bloques restantes [30] [107] [32]. El proyecto zfec incluye una biblioteca escrita en lenguaje C, una biblioteca escrita en lenguaje Python y una herramienta de línea de comandos. Este paquete se basa en gran medida en la antigua biblioteca fec, que es una implementación madura y optimizada de EC [154] [175] [123]. El paquete zfec hace varios cambios desde el paquete original fec, incluyendo la adición de la API de Python (interfaz de programación de aplicaciones), la refactorización de la API de C para apoyar la operación de copia cero, algunas limpiezas y optimizaciones del propio código central, y la adición de una herramienta de línea de comandos llamada zfec [31] [210] [29] [35].

La propuesta de esta investigación es proporcionar medios para la reconstrucción de imágenes médicas de forma que no perdamos cualidades

o características significativas de la enfermedad, ayudando a los médicos a interpretar y mejorar el diagnóstico. En este sentido, se pueden encontrar algunas aplicaciones recientes de PCA en medicina general en [115] [150] [6] [140] [195] [200]. Sin embargo, en este trabajo se presenta una novedosa aplicación de la técnica PCA en dermatología, orientada a la compresión de imágenes médicas de condiciones que involucran algunas enfermedades de la piel. En concreto, se utiliza la cuasi-periodicidad de las primeras componentes principales (PCs) [84] [85] [81] para realizar la compresión de imágenes médicas. Esencialmente, las PC que se consideran periódicas se sustituyen por su periodo más una tendencia, y la reconstrucción conseguida permite diagnosticar algunas enfermedades de la piel. Sin embargo, para evitar un falso positivo o negativo en el diagnóstico y tratamiento de enfermedades de la piel, este trabajo propone también la aplicación de técnicas de código de borrado para recuperar la imagen original. En concreto, se genera una copia fiel 100 % idéntica a la imagen original.

Finalmente, teniendo en cuenta todo lo dicho en este apartado, la motivación de este trabajo es posibilitar el diagnóstico sin dificultad del patrón maculopapular que se presenta en pacientes paucisintomáticos de COVID-19, mediante el uso de imágenes comprimidas de manifestaciones de este patrón. Este patrón representa una enfermedad de la piel.

5.2. Materiales y métodos

Para el estudio, disponemos de un conjunto de 22 imágenes de pacientes paucisintomáticos COVID-19, con diferentes resoluciones y de diferentes partes de su cuerpo, donde se muestran las manifestaciones dermatológicas. En la Fig. 5.2 se muestra un collage de las imágenes objeto de estudio. Para obtener reconstrucciones razonables de las imágenes comprimidas, éstas se comprimieron mediante la técnica PCA.

En este punto, es importante mencionar que para obtener las imágenes,

Figura 5.2: Conjunto de imágenes de pacientes con manifestaciones dermatológicas paucisintomáticas de COVID-19.

lo primero que se realizó fue contar con el consentimiento autorizado de cada paciente, para tomar las fotografías y publicarlas con fines de investigación científica. Las fotografías fueron tomadas con el teléfono inteligente Huawei P30 Lite MAR-LX3A junto con la aplicación Magnifier Camera 1.6.0 para Android. Por lo tanto, en este trabajo, el sensor es el sensor de imagen de la cámara del smartphone. En pocas palabras, este dispositivo produce una imagen digital a partir de la luz que atraviesa el objetivo de la cámara, y esta luz es captada por el sensor. Se trata de un sensor CMOS (Complementary Metal Oxide Semiconductor).

Para tomar las fotos, se eligieron las lesiones más recientes que son ilustrativas del patrón maculopapular y con buena iluminación, a una distancia razonable entre 50 milimetros y 105 milimetros, se tomaron las

fotos.

En este trabajo, del conjunto de imágenes objeto de estudio (véase Fig. 5.2), se eligió una con buena resolución y cuyo tamaño en píxeles era divisible por 8, tanto en filas como en columnas. Además, esta imagen era representativa de un patrón maculopapular. En la Fig. 5.3 se muestra la imagen elegida, su resolución es de 720×12800, y en el resto del trabajo se denominará Img.

Figura 5.3: Imagen representativa de un patrón maculopapular.

Una vez obtenidas las imágenes en color, se convierten del modelo de color RGB a YUV. Este modelo define 3 componentes: 1 luminancia y 2 crominancias. Por lo tanto, ahora cada imagen consta de 3 matrices de iguales dimensiones. Y la imagen antes mencionada (es decir, Img) tiene tres componentes: Img_1, Img_2 e Img_3 (es decir, 1 luminancia y 2 crominancia). Cada una de estas tres matrices se divide en submatrices no superpuestas de 8 por 8, y cada uno de los bloques formados por las submatrices forma un vector de dimensión 64.

A continuación, para la matriz de luminancia (es decir, la matriz

Img_1) se construye una matriz de bloques de dimensión 144000×64, que denominaremos X_1 y en la que cada fila representa los vectores previamente formados de dimensión 64. Según el procedimiento anterior, como la resolución de Img es 720×12800, y esta imagen se divide en bloques de dimensión 8×8, entonces la matriz Img_1 antes mencionada tiene 144000 filas (es decir, $\frac{720 \cdot 12800}{8 \cdot 8} = \frac{9{,}21 \cdot 10^6}{64} = 144000$ filas)

Del mismo modo, este proceso se repite para las dos matrices de crominancia (es decir, Img_2 e Img_3), construyendo las matrices X_2 y X_3 para las matrices Img_2 e Img_3, respectivamente. Así pues, ahora, para cada imagen tendremos tres matrices de bloques (es decir, $X_i \ \forall \ i = 1, \ldots, 3$), y a cada una de estas matrices le aplicamos el método de componentes principales con periodicidad desarrollado por Hernández y Méndez en [84] [85] [81].

Después de analizar la descomposición de Img para cada una de las matrices Img_1, Img_2 e Img_3, para decidir el número de PCs a elegir, analizamos primero el crecimiento de la variabilidad explicada. En este sentido, el número de PCs a tener en cuenta sería aquel que superase un determinado umbral. Trabajamos con un umbral igual al 90 %. Conociendo el número de PCs, conoceremos la tasa de compresión, CR. En concreto, si para comprimir sin pérdidas necesitamos 64 PCs, y comprimimos m PCs, entonces la tasa de compresión vendrá dada por (5.1).

$$CR = \frac{64 - m}{64} \tag{5.1}$$

Otro aspecto a tener en cuenta es comprobar la calidad de la compresión para saber si el proceso es eficiente. Para medir la calidad de la compresión utilizaremos el coeficiente pico señal-ruido (PSNR) tal y como se muestra en [84] [81], entre otras publicaciones científicas que también muestran este parámetro. En definitiva, este parámetro compara las diferencias entre la imagen comprimida y la imagen original para verificar la calidad. Además, también calcularemos el error cuadrático medio (MSE) de las diferencias

entre la imagen original y la comprimida para medir la calidad de la compresión. Para ello, utilizaremos el parámetro MSE tal y como se muestra en [84] [81]. Asimismo, cabe mencionar que existen numerosas publicaciones científicas que también describen este parámetro.

Además, en este trabajo se llevó a cabo un análisis del rendimiento del algoritmo de código de borrado Reed-Solomon de la biblioteca zfec [30] [107] [32] [154]. Este análisis se realizó en un entorno de resiliencia, en el que se aplicó la reconstrucción de fragmentos perdidos a partir de fragmentos distribuidos en el almacenamiento de la configuración.

Para el análisis de rendimiento, se probó una biblioteca práctica de alto rendimiento, zfec [175] [123] [31] [210] [29] [35], implementando un conjunto de pruebas. Esta suite de pruebas realiza el proceso de codificación y decodificación del código de borrado Reed-Solomon. Vale la pena mencionar que cuando se trata de códigos de borrado [62], dos métricas de rendimiento son importantes: la eficiencia de almacenamiento y tolerancia a fallos. El código de borrado Reed-Solomon implica un compromiso entre ambas.

La eficiencia del almacenamiento es un indicador del almacenamiento adicional necesario para garantizar la resiliencia, mientras que la tolerancia a fallos es un indicador de la recuperabilidad en caso de fallos de fragmentos. Para aprovechar al máximo el rendimiento del hardware, se utilizó un servidor con High Performance Computing (HPC) [200] [229] [159] [204]. Este servidor era un Cisco UCS 5108-AC2 Blade Server Chassis System, con un procesador Intel® Xeon® CPU E5-2630 v4 @ 2,2 GHz, 128 GB de RAM, unidad de almacenamiento SCSI con capacidad de almacenamiento de 1,2 TB, velocidad de rotación del disco duro de 10.000 RPM y velocidad de transferencia de la interfaz de disco de 12 Gbps. En el entorno local, esto permite al investigador cambiar rápidamente el tamaño de los archivos y optimizar los procesos de codificación y decodificación, respectivamente. Además, para garantizar la integridad de los archivos JPG restaurados, se aplicó el algoritmo SHA-512 [104] [197] [37] [55] . La Fig. 5.4 muestra la

implementación del algoritmo para todos los casos de grupos de ficheros utilizados en este trabajo.

Para el estudio, disponemos de un conjunto de 22 imágenes de pacientes paucisintomáticos de COVID-19, con diferentes resoluciones de compresión con y sin pérdida, y archivos de diferente tamaño (véase Tabla. 5.1). Estas imágenes muestran diferentes partes del cuerpo en las que se observan manifestaciones dermatológicas de una enfermedad cutánea. Y, como ya se ha mencionado, la Fig. 5.2 muestra un collage de las imágenes objeto de estudio. En primer lugar, estas imágenes se comprimen mediante la técnica PCA. Pero si el médico percibe que la imagen comprimida no tiene la calidad suficiente para diagnosticar correctamente la enfermedad, entonces se utiliza la batería de pruebas mostrada en la Fig. 5.4 para recuperar la imagen original. Es importante señalar que la batería de pruebas tiene implementado el código de borrado Reed-Solomon de la librería nativa zfec.

Finalmente, como debe ser, será un médico quien dé el visto bueno para saber si la compresión de imágenes realizada cumple las expectativas, y cuándo utilizar la reconstrucción de las imágenes comprimidas.

5.3. Resultados

5.3.1. Compresión de imágenes de manifestaciones dermatológicas mediante componentes principales (PCs)

Para aplicar el método descrito en la Sección 5.2, la imagen objeto de estudio (es decir, Img) se divide en bloques y se descompone en PCs. La Fig. 5.5 muestra los primeros mayores valores propios y la variabilidad explicada seleccionando esos valores propios para la primera matriz de Img (es decir, Img_1) con su matriz de bloques X_1. Realizamos este análisis para las tres matrices de Img, y construimos también las matrices de bloques X_2 y X_3. Estas matrices de bloques resultaron ser análogas, en el sentido de que el primer valor propio es muy grande (es decir, $\lambda_1 = 99,1427$) en comparación

Tabla 5.1: Tamaño del archivo utilizado en el entorno de ensayo.

Item	**formato PNG**	**formato JPG**
A	1075.26 KB	224.28 KB
B	262.00 KB	9.60 KB
C	554.00 KB	115.00 KB
D	1280.00 KB	62.90 KB
E	622.00 KB	62.40 KB
F	671.00 KB	62.40 KB
G	409.00 KB	44.00 KB
H	909.00 KB	186.00 KB
I	977.00 KB	194.00 KB
J	978.00 KB	194.00 KB
K	823.00 KB	174.00 KB
L	836.00 KB	176.00 KB
M	485.00 KB	51.50 KB
N	360.00 KB	35.20 KB
O	360.00 KB	35.20 KB
P	1003.52 KB	195.00 KB
Q	1003.52 KB	205.00 KB
R	820.00 KB	164.00 KB
S	341.00 KB	32.10 KB
T	1003.52 KB	205.00 KB
U	407.00 KB	39.90 KB
V	359.00 KB	33.00 KB

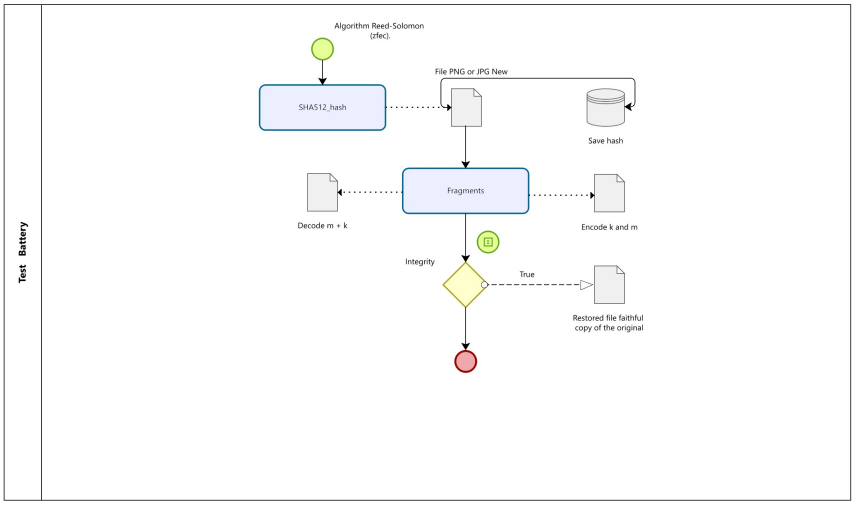
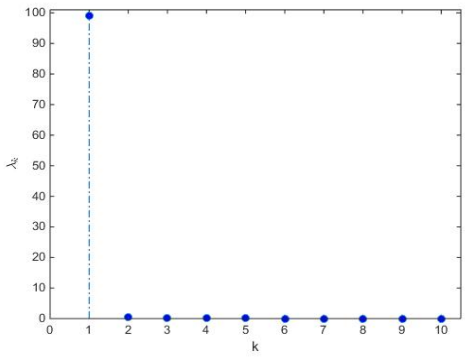


Figura 5.4: Batería de ensayo: usando biblioteca $zfec$ aplicando código de borrado Reed-Solomon.

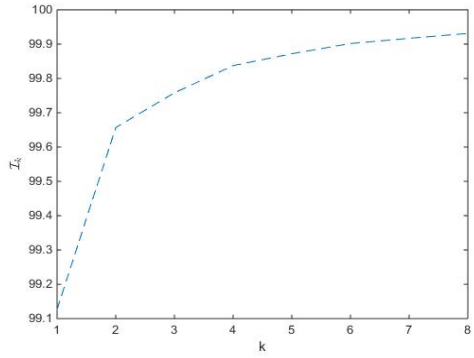
con el segundo (es decir, $\lambda_2 = 0,5270$) (véase Fig. 5.5(a)). Además, el crecimiento de la variabilidad explicada se estabilizó a partir de dos PC, y con las dos primeras PC se obtiene más del 99,6 % de la variabilidad mostrada por la matriz de los bloques de imágenes (véase Fig. 5.5(b)).

A continuación, se calculan la PSNR y el MSE de la imagen original frente a las imágenes comprimidas, ya que los PCs se añaden a la compresión. La Fig. 5.6 muestra estos coeficientes para la imagen seleccionada (es decir, Img). Además, esta figura muestra que a partir de los primeros PCs la PSNR es superior a 30 y que este valor crece, como es lógico, hasta llegar a 100, que es el valor obtenido al considerar todos los PCs. Esta figura también muestra una caída muy fuerte en el valor del MSE, al tomar los diez primeros PCs. A partir de este valor, el MSE cae gradualmente hasta 0. Las dos gráficas mostradas en la Fig. 5.6 permiten seleccionar el número de PCs con criterios basados en la calidad de la compresión.

Por otro lado, la tasa de compresión (véase Fig. 5.1) disminuye

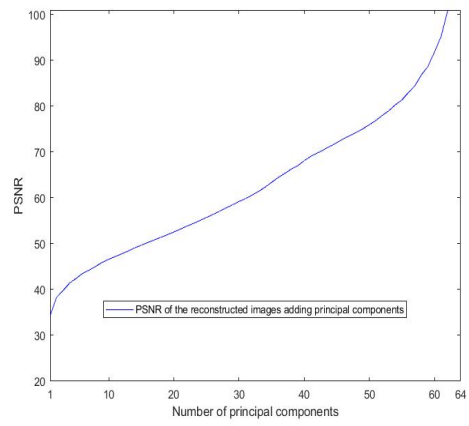


(a) Magnitude (λ_k) of the first 10 eigenvalues (k, $k = 1, \ldots, 10$).

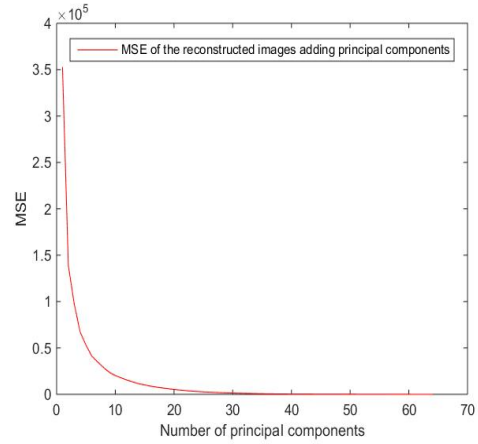


(b) Cumulative variability (I_k) explained with the first 8 eigenvalues (k, $k = 1, \ldots, 8$).

Figura 5.5: Magnitud de los primeros valores propios y varianza acumulada explicativa.



(a) PSNR of the original image versus the compressed images.



(b) MSE of the original image versus the compressed
images.

Figura 5.6: PSNR y MSE de la imagen original frente a las imágenes comprimidas a medida que se añaden PC (componentes principales) a la compresión.

linealmente a medida que se utilizan más PCs para la compresión. La Fig. 5.7 muestra la evolución de la tasa de compresión a medida que se añaden más PCs.

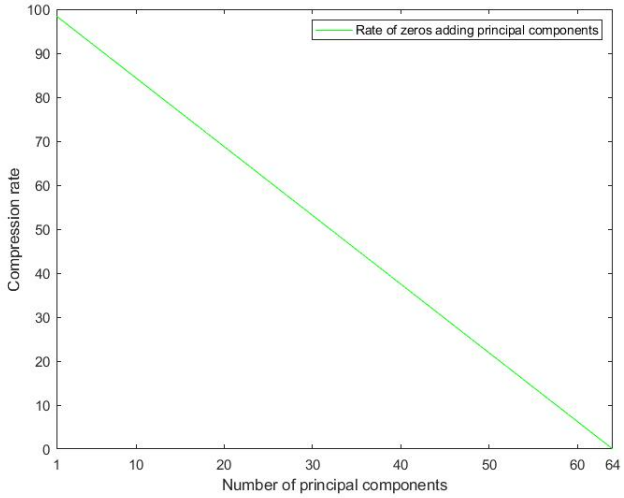


Figura 5.7: Tasa de compresión de la imagen original a medida que se añaden los PC (componentes principales).

A la vista de los resultados obtenidos, decidimos comprimir con uno y dos PCs. Al comprimir la imagen Img con dos PCs, se obtiene una variabilidad explicada del 99,6 %, una PSNR de 38,2797, y una tasa de compresión del 96,8 %. Mientras que al comprimir la misma imagen con un solo PC, la variabilidad explicada desciende a 99,1 %, la PSNR también desciende a 34,2335 y la tasa de compresión aumenta a 98,4 %. La Fig. 5.8 muestra el resultado de las compresiones.

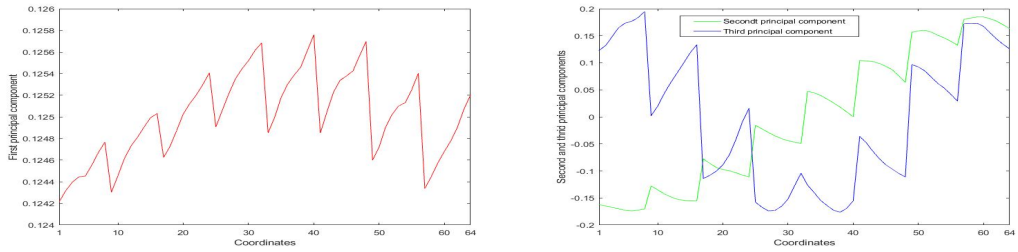
Al guardar las imágenes comprimidas en formato .jpg, el tamaño de la imagen original es de 45.060 bytes y el de la compresión con dos PCs es de 24.274 bytes, siendo éste el 53,9 % del tamaño de la imagen original. En cuanto a la compresión con un PC, el tamaño de ésta es de 22.561 bytes,

(a) Original image: A maculopapular pattern.

(b) Original image compressed by using two PCs.

(c) Original image compressed by using one PC.

Figura 5.8: Imagen original comprimida utilizando uno y dos PC.



(a) First PC. (b) Second and third PCs.

Figura 5.9: Gráficos de los tres primeros PC (componentes principales) de X_1.

siendo el 50,1 % del tamaño de la imagen original.

A continuación comprobamos que, efectivamente, al realizar las gráficas de las primeras PCs, éstas presentan características periódicas. En la Fig. 5.9 se muestran las gráficas de las tres primeras CP de X_1. Como puede observarse, los valores del primer PC son casi todos iguales, oscilando entre 0,1242 y 0,1258. Por tanto, casi forman una línea horizontal. Así pues, calculamos los 8 valores medianos con retardo 8, generamos un vector de período 8 y tamaño 64, y lo sustituimos en el primer PC. Se hizo esto para cada una de las tres matrices X_1, X_2 y X_3.

En Fig. 5.9(b), el segundo PC muestra una tendencia lineal. Tras eliminar esa tendencia, se consideró los residuos, recalculamos la mediana de los valores con retardo 8, se generó un vector de periodo 8 de dimensión 64 y añadimos a este vector la tendencia eliminada anteriormente.

La Fig. 5.10(a) muestra la compresión de la imagen original con un solo PC periódico. En este caso, la PSNR es de 24,0421 y la tasa de compresión es del 99,8 %. Además, como la PSNR es inferior a 30, la compresión tiene una calidad baja. Esto se debe a que esta imagen médica reconstruida carece prácticamente de color suficiente para mostrar las manifestaciones de la enfermedad dermatológica.

Por otro lado, la imagen comprimida con dos PCs, siendo el primero

de ellos fue sustituido por la periodicidad, se muestra en la Fig. 5.10(b). Esta imagen tiene un valor PSNR de 38,2492 y su tasa de compresión es del 98,2 %. Además, la imagen comprimida con dos PCs periódicos se muestra en la Fig. 5.10(c). La PSNR de esta imagen es de 33,7260 y la tasa de compresión es del 99,6 %.

La imagen comprimida en formato .jpg con un PC periódico tiene un tamaño de 19.677 bytes, lo que supone el 43,7 % del tamaño de la imagen original. El tamaño de la imagen comprimida con dos PCs, siendo el primero periódico, es igual a 21.240 bytes, siendo el 47,1 % del tamaño de la imagen original. Y, para la compresión con dos PCs periódicos, el tamaño de esta imagen es igual a 20.642 bytes, que es el 45,8 % del tamaño de la imagen original. La Tabla. 5.2 muestra un resumen de las características de las compresiones realizadas.

Tabla 5.2: Características de las imágenes comprimidas.

Compresión	**Explicación acumulativa de variabilidad**	**PSNR**	**Ratio de compresión (%)**	**Reducción del tamaño (%)**
One PC	99.1	34.2335	98.4	49.9
Two PCs	99.6	38.2797	96.8	46.1
One periodic PC	-	24.0421	99.8	56.3
Two PCs, with the first periodic	-	38.2492	98.2	52.9
Two PCs, with both periodics	-	33.7260	99.6	54.2

Desde el punto de vista de la dermatología, los médicos consultados expresaron que en todas las fotos se observa con muy buena nitidez y claridad uno de los patrones más frecuentes en el COVID-19, el patrón

(a) One periodic PC.

(b) Two PCs, with the first periodic.

(c) Two PCs, with both periodic.

Figura 5.10: Diferentes compresiones de la imagen original, con uno y dos PC (componentes principales), y utilizando la periodicidad de los primeros PC.

maculopapular o morbiliforme. Este patrón fue descrito por primera vez en 2020, tras la aparición del COVID-19. Por lo tanto, en la literatura no hay información al respecto hasta ahora en el curso de esta enfermedad [106] [96] [66] [76] [215].

Este patrón se manifiesta clínicamente en el paciente mostrado en la Fig. 5.3, y se caracteriza por una erupción eritematopapular diseminada en la parte inferior de la espalda. Esta erupción se extiende hasta la nalga derecha en casi su totalidad. Además, en las imágenes médicas mostradas, tanto en las originales como en las comprimidas, se aprecian pápulas eritematosas. Algunas de estas pápulas son del color normal de la piel, pequeñas y de unos pocos milímetros de longitud. Además, algunas de ellas se elevan y otras no. Además, se observa que estas pápulas convergen, dejando en su interior zonas de piel sana. Todo ello permite llegar al diagnóstico de la patología sin ninguna dificultad, tanto en la foto original como en las que están comprimidas. Como se ha comentado en este apartado, la imagen de menor calidad es la compresión con un PC periódico, pero incluso esta imagen transmite información relevante [180] [106] [96] [66] [76] [215] [64].

5.3.2. Tolerancia a fallos de imágenes de manifestaciones dermatológicas mediante códigos de borrado

Teniendo en cuenta lo dicho al final del epígrafe 5.3.1, es importante añadir que el proceso diagnóstico en dermatología ha evolucionado históricamente al mismo tiempo que el de la medicina. Esto ha sucedido de acuerdo con la teoría filosófica y doctrinal imperante en cada momento. Es por ello que la dermatología utiliza las mismas herramientas que en otras especialidades. Por ejemplo, se utiliza la anamnesis, la exploración física, la imagen médica (es decir, la medicina basada en la evidencia) y, en ocasiones, pruebas complementarias. Estas últimas, sumadas al estudio y a los conocimientos médicos previos, tanto teóricos como basados en la

experiencia, permiten llegar al diagnóstico correcto de la enfermedad. En este subapartado (es decir, en la Sección 5.3.2), se demuestra en la práctica la utilidad de la aplicación de algoritmos de tolerancia a fallos a imágenes con manifestaciones dermatológicas (véase Fig. 5.2). Esto se hace con el fin de evitar falsos positivos o falsos negativos en el diagnóstico y tratamiento de enfermedades de la piel.

La Fig. 5.11 muestra los formatos de archivo utilizados en esta subsección. El eje horizontal muestra el conjunto de imágenes y el eje vertical los distintos tamaños de los archivos, representados en KB (es decir, en kilobytes).

Los archivos originales son del tipo JPG, que es el formato gráfico más común para fotografías. La compresión de imágenes con este formato ya fue definida en 1992 por la norma ISO/IEC 10918-1. Es decir, cuanto mayor sea la relación de compresión de una imagen JPG, peor será su calidad. En este contexto, la compresión se indica en los programas de edición de imágenes como un valor de calidad. En otras palabras, un valor de 100 indica una calidad de 100 %, sin compresión. Pero el nivel de compresión y el tamaño del archivo no están relacionados linealmente. Reducir la compresión manteniendo una alta calidad puede suponer una notable reducción del tamaño del archivo.

Según lo descrito anteriormente, incluimos en nuestra investigación los mismos archivos pero en formato PNG, aumentando considerablemente su tamaño (véase Fig. 5.11). El formato PNG es un formato gráfico especialmente adecuado para gráficos de píxeles, capturas de pantalla y logotipos. El formato PNG son las siglas de Portable Network Graphics. Este formato ha sido desarrollado por un grupo de trabajo del World Wide Web Consortium (W3C), a partir de 1994. Posteriormente, este formato se incluyó en la norma ISO/IEC 15948:2003. En pocas palabras, PNG no tiene pérdidas y ofrece la posibilidad de determinar diferentes profundidades y gamas de color, seleccionadas con flexibilidad.

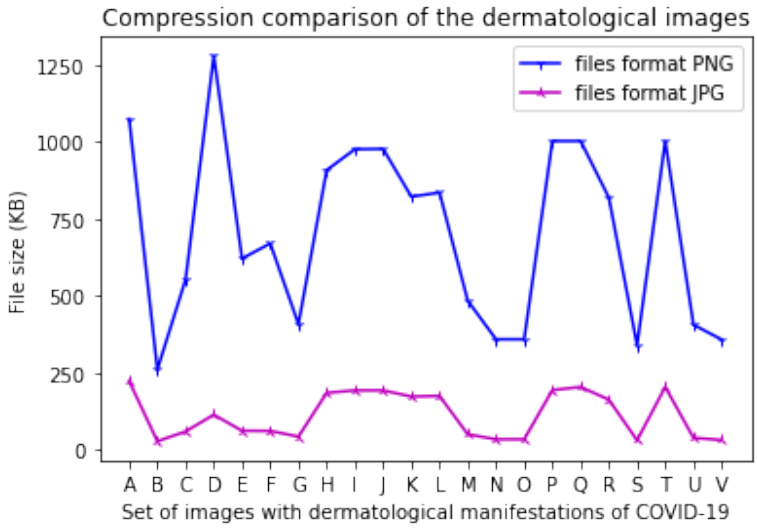


Figura 5.11: PNG vs JPG una aproximación desde el almacenamiento. El eje x representa la imagen y el eje y su tamaño.

El código fuente para pasar del formato JPG al formato PNG se describe a continuación:

```
# Ejemplo de guardar una imagen en otro formato
from PIL import Image
# cargar la imagen
image = Image.open('file_name.jpg')
# guardar en formato PNG
image.save('file_name.png', format='PNG')
# cargar la imagen de nuevo e inspeccionar el formato
image2 = Image.open('file_name.png')
print(image2.format)
```

A continuación se muestran los resultados obtenidos para las configuraciones utilizadas en los procesos de codificación y decodificación. En primer

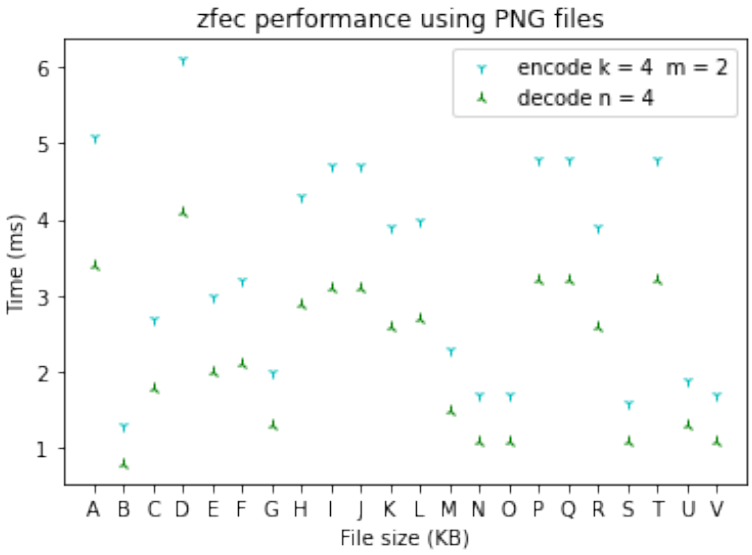


Figura 5.12: Métricas en tiempos de procesamiento: Archivos PNG. El eje x representa el tamaño de la imagen y el eje y el tiempo de procesamiento de la imagen.

lugar, tal y como se mencionó en el apartado 5.1, en el proceso de codificación se utilizaron configuraciones en relación con el tamaño de los bloques de datos originales k y el tamaño de los bloques de paridad m, para la recuperación de los posibles bloques perdidos. Esta misma relación se utiliza para realizar la decodificación de los datos [147].

Test 1.- En esta prueba se utilizaron archivos PNG de diferente tamaño (véase Tabla. 5.1). Estos archivos contienen parte del cuerpo humano con manifestaciones dermatológicas en pacientes paucisintomáticos de COVID-19 (véase Fig. 5.2). Los resultados de esta configuración se muestran en la Fig. 5.12. Esta figura muestra el tiempo consumido por cada archivo, en una proporción de $k = 4$ fragmentos de datos, y $m = 2$ fragmentos de redundancia, respectivamente.

En el caso de la codificación $RS(4, 2)$, se observó un consumo de

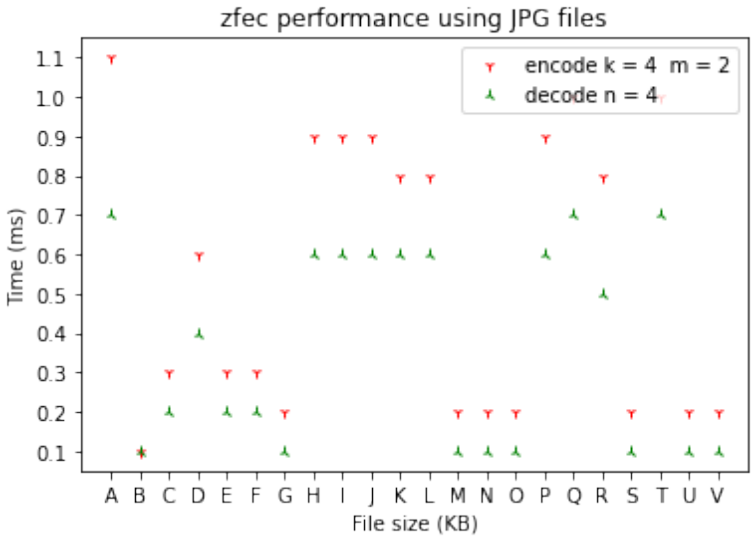


Figura 5.13: Métricas en tiempos de procesamiento: Archivos JPG. El eje x representa el tamaño de la imagen y el eje y el tiempo de procesamiento de la imagen.

aproximadamente 74,40 ms al procesar 22 archivos PNG. Además, en el caso de la decodificación $RS(2,2)$, se observó un consumo de aproximadamente 49,03 ms.

Test 2.- En esta prueba se utilizaron archivos JPG de diferente tamaño (véase Tabla. 5.1). Estos archivos contienen parte del cuerpo humano con manifestaciones dermatológicas en pacientes paucisintomáticos de COVID-19 (véase Fig. 5.2). Los resultados de esta configuración se muestran en la Fig. 5.13. Esta figura muestra el tiempo consumido por cada archivo, en una proporción de $k = 4$ fragmentos de datos y $m = 2$ fragmentos de redundancia, respectivamente.

En el caso de la codificación $RS(4,2)$, se observó un consumo de aproximadamente 12,03 ms al procesar 22 archivos JPG. Además, en el caso de la decodificación $RS(2,2)$, se observó un consumo de aproximadamente

8,01 ms.

5.4. Análisis de Resultados

Tratar de identificar los patrones típicos de las manifestaciones dermatológicas, con el fin de diagnosticar la enfermedad COVID-19, desempeña un papel relevante cuando los médicos no pueden contar con las manifestaciones respiratorias. Esto ocurre en el caso de los pacientes paucisintomáticos de COVID-19. De hecho, los pacientes paucisintomáticos, en muchas ocasiones, tienen como único elemento clínico las manifestaciones dermatológicas. Muchos de estos pacientes son luego confirmados como COVID-19 positivos. Por ello, es importante desarrollar procedimientos que contribuyan a resolver el problema planteado.

La aplicación de métodos como los mostrados en este artículo ayuda a los médicos a aprovechar al máximo la capacidad de memoria de las herramientas que utilizan para guardar fotografías de las manifestaciones de la enfermedad. Es como si el médico dispusiera ahora de mayor capacidad tecnológica para almacenar mucha más información sobre la enfermedad en el mismo espacio. Por lo tanto, esto contribuye a la identificación precoz de la presencia de la enfermedad. En consecuencia, ayuda a prevenir las complicaciones respiratorias, las infecciones, la morbilidad y la mortalidad que genera, la discapacidad funcional resultante y la percepción de la calidad de vida relacionada con la salud de los pacientes que padecen esta enfermedad. Esto representa un avance en términos de prevención de la salud. En concreto, representa un avance en la posible ralentización de la transmisión del COVID-19.

Este trabajo parte de un conjunto de imágenes de manifestaciones dermatológicas de COVID-19 (véase Fig. 5.2) y muestra un procedimiento para reducir su tamaño, sin perder la información relevante que transmiten. Este procedimiento se basa en el análisis de componentes principales. En

pocas palabras, se utilizan combinaciones lineales de variables para explicar la estructura de varianza y covarianza de las imágenes. Además, esto se hace preservando las partes significativas de la información original.

Los médicos que formaron parte del equipo de investigación identificaron una imagen representativa del patrón maculopapular (véase Fig. 5.3), dentro del conjunto de imágenes (véase Fig. 5.2), y procedimos a comprimirla. Para ello, mediante componentes principales, se busca el conjunto de ejes ortogonales que mejor refleje la cantidad de información de la imagen original. Lo que se busca es resaltar las diferencias para que no todos los píxeles sean iguales, mostrando mejor la cantidad de información que contiene la imagen. A continuación, se reduce la dimensión del problema y se realiza una transformación para facilitar la interpretación [84].

Como se explica en el apartado 5.2, la imagen RGB tomada como ejemplo (véase Fig. 5.3) se convierte a un modelo de color YUV, y se obtienen 3 matrices: una de luminancia y dos de crominancia. A continuación, se aplica el método de componentes principales a cada una de estas matrices y se analiza la variabilidad explicada. La Fig. 5.5 muestra la magnitud de los primeros valores propios y la variabilidad explicada por ellos. Esto nos permitió obtener un valor satisfactorio para la tasa de compresión, y medir la calidad de la compresión utilizando el coeficiente pico de la relación señal/ruido y el error cuadrático medio de las diferencias entre las imágenes original y comprimida. La Fig. 5.6 muestra que la calidad de la imagen comprimida es muy buena incluso trabajando con 2 PCs. Por lo tanto, procedimos a comprimir la imagen original utilizando sólo 2 PCs. Además, los resultados mostrados en la Fig. 5.6 demostraron que incluso con 1 PC, la calidad de la imagen comprimida era satisfactoria. Esto se corroboró experimentalmente observando la Fig. 5.8. Además, la Tabla. 5.2 muestra la reducción conseguida en el tamaño de la imagen, y al final de la Sección 5.3.1 los médicos aclararon que las imágenes comprimidas contienen toda la información necesaria para diagnosticar correctamente la

enfermedad.

En este punto, podríamos haber terminado la investigación en términos de compresión con pérdida de las imágenes. Sin embargo, descubrimos que las primeras componentes principales eran periódicas, y lo aprovechamos para reducir aún más el tamaño de las imágenes. Esto coincide con lo descubierto y desarrollado en [84] [82] [83] [85] [81]. Por lo tanto, basándonos en la información mostrada en la Fig. 5.9, decidimos comprimir las imágenes utilizando componentes principales periódicas. La Fig.5.10 demuestra que esto es posible en la práctica, y una vez más, al final de la Sección 5.3.1, los doctores miembros del equipo de investigación confirmaron su aceptación de los resultados experimentales. La Tabla. 5.2 muestra la reducción adicional del tamaño de la imagen conseguida al utilizar 1 y 2 PCs periódicos.

Antes de pasar a la discusión de la parte dedicada a la recuperación de imágenes, generando imágenes 100 % idénticas a las imágenes originales, queremos mencionar que la Fig. 5.10(a) es la de peor calidad. Sin embargo, pensamos que esto se debe a que las imágenes originales, tomadas con una cámara de rendimiento medio, no tenían una alta resolución. En consecuencia, es lógico que al aplicar una compresión tan grande, utilizando sólo 1 PC periódico, se pierdan algunas cualidades del color. Por lo tanto, una futura dirección de investigación será probar teléfonos móviles con cámaras mucho más potentes, e intentar mejorar la compresión de imágenes utilizando componentes principales periódicas, pero manteniendo la sencillez del algoritmo presentado en [84].

En esta investigación, la recuperación de imágenes se basó en el uso de códigos de borrado. En pocas palabras, estos códigos son transformaciones que proporcionan tolerancia a fallos y fiabilidad. Además, juegan un papel importante para diferentes sistemas de almacenamiento de datos a gran escala, garantizando la disponibilidad de la información, tales como: Amazon S3, Google File System, Microsoft Azure y los sistemas de almacenamiento de Facebook.

Para un código de borrado (n, k, m), un mensaje o dato original consta de k datos de igual tamaño. A continuación, se calculan m datos de paridad a partir de los k datos, mediante el proceso de codificación. En este caso, los k fragmentos de datos y los m fragmentos de paridad forman un n grupo de fragmentos de datos, donde $n = k + m$, de tal forma que se puede tolerar la pérdida o eliminación de cualquier fragmento de k. Es decir, los k fragmentos originales de datos pueden ser recuperados exactamente de los m fragmentos, por el proceso de decodificación [62] [105] [1] [217] [181].

El código de borrado Reed-Solomon (RS) utilizado en nuestra investigación se remonta a los años 60, desarrollado por Irving S. Reed y Gustave Solomon. El código RS original se describía como códigos polinómicos sobre determinados campos finitos. El esquema de codificación original utilizaba un polinomio basado en el mensaje a codificar. En él, la codificación y decodificación sólo conocen un conjunto fijo de valores, denominados puntos de evaluación, para ser codificados. Sin embargo, hoy en día la mayoría de sus implementaciones adoptan la forma matricial para facilitar su comprensión e implementación. Además, los códigos RS son mucho más que simples códigos y pueden corregir errores en diversos sistemas de comunicación y almacenamiento [152] [230] [56] [98] [137] [171] [147] [185] [203] [202].

La batería de pruebas construida en esta investigación tenía como misión fundamental proporcionar una segunda opinión médica y evitar falsos positivos o negativos en el diagnóstico por parte del facultativo. Esta investigación utilizó como entrada principal archivos JPG (compresión con pérdida) y luego aplicó compresión sin pérdida, formato PNG. Estos archivos contienen información relevante sobre medicina dermatológica basada en evidencia. Aquí, la batería de pruebas se realizó aplicando el algoritmo de código de borrado Reed-Solomon de la biblioteca zfec [246] [30] [107] [32] [154] [175] [123] [31] [210] [29] [35]. Además, las técnicas utilizadas para evaluar el rendimiento fueron la codificación y la decodificación, con una sobrecarga de almacenamiento de 50 %. Es decir, $k = 4$; $m = 2$. Esta fue

la técnica utilizada por los autores en [147], consiguiendo obtener resultados experimentales con un rendimiento óptimo.

Los resultados experimentales mostraron que en el primer grupo de archivos, JPG vs. PNG tiene un mejor desempeño en términos de tiempos de procesamiento (véase Tabla. 5.1). Asimismo, se demostró que en eventos adversos, tales como: daño, pérdida, deterioro parcial o deterioro total, es posible recuperar completamente los 22 archivos JPG originales de imágenes médicas (véase Fig. 5.2). En esta investigación, dicha recuperación es una copia exacta de las imágenes originales. Para ello se utilizó la técnica de decodificación del algoritmo de borrado Reed-Solomon de la librería zfec [246] [30] [107] [32] [154] [175] [123] [31] [210] [29] [35]. Aquí, la suma del tamaño de las 22 imágenes médicas utilizadas era igual a 2,4 Megabytes (MB), y se verificó que la recuperación era una copia 100 % igual a las imágenes originales. Esta verificación se realizó aplicando el algoritmo hash SHA-512 [104] [197] [37] [55], con un tiempo de procesamiento igual a 8,01 milisegundos.

Por último, pero no por ello menos importante, cabe mencionar que sería inconcebible leer una publicación científica sobre dermatología sin fotografías clínicas. Además, cabe señalar que hoy en día los archivos de tipo JPG siguen siendo la piedra angular del diagnóstico dermatológico. Por lo tanto, estos archivos son de suma importancia. Además, gracias al uso de algoritmos tolerantes a fallos, como el propuesto en este trabajo, el panorama de la práctica clínica ha cambiado drásticamente en las últimas décadas. Todo ello ha brindado la oportunidad de mejorar tanto la calidad de la asistencia como la prontitud de los diagnósticos. Por ello, es fundamental que la dermatología evolucione a la par que la tecnología y que se proporcionen diagnósticos oportunos a los pacientes, con el fin de lograr un mejor manejo de su enfermedad.

Capítulo 6

Capa de almacenamiento distribuida genérica de ficheros.

Este capítulo pretende explicar la puesta en marcha de una capa de almacenamiento distribuida resiliente, implementada en un sistema operativo basado en Debian con el fin de exponer el rendimiento de los algoritmos de compresión sin perdida y las técnicas de codificación y decodificación de los diferentes códigos de borrado de la biblioteca PyECLib nativa de Python. Además, con convergencia en los servicios de AWS como las bases de datos DynamoDB y Amazon Quantum Ledger Database combinado con Amazon Simple Storage Service S3, con el fin de obtener tolerancia a fallos, con un alto nivel de durabilidad de los fragmentos y la visualización de la inmutabilidad de los registros generados. Los resultados obtenidos muestran tiempos de procesamiento favorables que pueden ser aplicados a cualquier plataforma de sistema de almacenamiento distribuido open source.

6.1. Introducción

Los sistemas de almacenamiento distribuido son sistemas que se utilizan para almacenar datos y programas en redes de computadoras, estos sistemas son muy útiles porque permiten ahorrar espacio en equipos móviles y servidores centrales, y también ofrecen mayor seguridad a los usuarios [80].

El uso de sistemas de almacenamiento distribuido es cada vez más común, y se está convirtiendo en una herramienta esencial para la protección de los datos, los sistemas de almacenamiento distribuido han tenido un gran impacto en la industria, estos sistemas permiten a las empresas compartir y almacenar datos de manera más eficiente y asequible, esto les ha permitido ahorrar tiempo y costos a la hora de acceder a los datos [205].

Por ejemplo, la adopción de almacenamiento zonificado en sistemas de almacenamiento distribuido puede proporcionar numerosos beneficios. En [5], explican que el almacenamiento zonificado es una técnica de almacenamiento de bloques que divide los bloques en grupos de tamaño fijo y asigna una zona a cada grupo. Esto permite a los sistemas de almacenamiento distribuido realizar operaciones de lectura y escritura más rápidas y eficientes al reducir el número de bloques que se deben leer o escribir. Además, los sistemas de almacenamiento zonificado también permiten a los usuarios aprovechar los recursos de almacenamiento de forma más eficiente al asegurar que los bloques se almacenan en los mejores dispositivos de almacenamiento [60]. Por último, el almacenamiento zonificado también mejora la seguridad almacenando los bloques más importantes en dispositivos de almacenamiento seguros [45].

Además, estos sistemas también han mejorado la seguridad de los datos al permitir la replicación de datos en varios servidores [198]. Esto significa que, si uno de los servidores se cae, los datos seguirán estando disponibles en el resto de servidores [132]. Otra de las principales ventajas de los sistemas de almacenamiento distribuido es su escalabilidad. Es decir, si una empresa necesita más capacidad de almacenamiento, puede añadir más servidores

sin necesidad de cambiar todo el sistema. Esto les ahorra tiempo y recursos a las empresas, lo que les permite ahorrar costes [160].

En [179], se trata sobre la importancia de la localidad y la disponibilidad en el almacenamiento distribuido, explicando cómo la localidad es una forma de reducir el tiempo de acceso a los datos almacenados, mientras que la disponibilidad se refiere a la estabilidad y la fiabilidad de los datos. También [179], discuten los desafíos de mantener una combinación de localidad y disponibilidad en entornos de almacenamiento distribuido, así como las técnicas para alcanzar este objetivo.

Por otro lado, para mantener disponibilidad en los sistemas, es común usar técnicas de Erasure Code (EC). EC es un método de almacenamiento de datos que se utiliza para proporcionar redundancia y recuperación de datos en caso de pérdida de información [128]. Utiliza códigos de corrección de errores para dividir los datos en fragmentos y distribuirlos entre varios servidores. Esto hace que los datos sean más resistentes a fallos y errores, ya que si un servidor se cae, los datos se pueden recuperar de los otros servidores [129] [236]. Esto también mejora la recuperación de datos en caso de fallos en el disco, ya que los datos se pueden recuperar de otros servidores [169].

En concreto, la esencia de este trabajo se centra en tres etapas, la primera etapa es evaluar los algoritmos de compresión sin perdida [34] [139] [209] a ficheros con información de genomas, dado que el tamaño de los ficheros de los genomas es considerablemente alto [79] [61], al aplicar los algoritmos de compresión sin perdida se alcanzan resultados favorables con la reducción del tamaño, la segunda etapa es utilizar el fichero más significativo con compresión, y procedemos a aplicar una evaluación de la biblioteca de código de borrado PyECLib [116], dos métricas aquí en esta etapa son aplicadas el encode y decode respectivamente para cada algoritmo de esta biblioteca, obteniendo resultados en tiempo de procesamiento óptimo para la tolerancia a fallos. La tercera etapa con base en lo anterior, el objetivo principal del

artículo, consistió en el desarrollo de una capa resiliente de almacenamiento distribuida, para ello fue necesario utilizar los algoritmos de compresión y código de borrado con mejor rendimiento. Además, una colección de servicios de computación en la nube, específicamente de Amazon Web Services (AWS) [247] [131] [136] [237].

Cabe señalar que el hardware local utilizado en todas las etapas, fue un equipo de alto rendimiento (HPC) Cisco UCS 5108-AC2 Blade Server Chassis [131].

Los ficheros utilizados contienen información de genomas, que son aplicados en investigación científica con énfasis a la ingeniería genética, la ingeniería bioquímica y la bioinformática [91] [165]. La ingeniería genética se basa en los conocimientos adquiridos de la estructura y función de los genomas para manipular la información genética de los organismos. Estas manipulaciones pueden incluir la modificación de genes para producir sustancias específicas, la ingeniería de organismos para producir bienes y servicios, la producción de nuevas cepas de plantas y animales para mejorar su resistencia a enfermedades y la producción de organismos transgénicos para producir productos farmacéuticos [2] [235]. La ingeniería bioquímica se centra en el diseño y construcción de enzimas sintéticas para catalizar reacciones químicas deseadas, así como en la síntesis de moléculas naturales y sintéticas para uso en medicina y biotecnología. La bioinformática se centra en el uso de la información genética para estudiar los mecanismos subyacentes de la biología molecular [26] [157] [206].

Otros sistemas de almacenamiento, cómo un clúster son vulnerables a los fallos de los componentes, pero los EC pueden alcanzar el mismo nivel de fiabilidad con una cantidad mucho menor de redundancia [129] [239]. Los sistemas de almacenamiento empresariales utilizan la replicación y EC para mantener la fiabilidad de los datos; sin embargo, la codificación de borrado suele ser más eficiente que la replicación. HDFS (Hadoop Distributed File System) utiliza la replicación en tres direcciones (R3), que introduce

una sobrecarga de almacenamiento del 300 %, mientras que EC reduce la sobrecarga al 133 %. Como resultado, EC es ampliamente utilizado por los sistemas de almacenamiento empresarial, ya que logra niveles similares de fiabilidad sin requerir tanto espacio redundante [236] [209].

La codificación de borrado es una tecnología de almacenamiento de datos que puede utilizarse para ahorrar espacio. Sin embargo, la codificación de borrado incurre en una mayor sobrecarga de entrada / salida (I/O) que la replicación cuando disminuye el número de escrituras por entidad [231]. Los sistemas de codificación de borrado suelen emplear la reconstrucción-escritura o la lectura-modificación-escritura para mantener la coherencia y la paridad en cada escritura parcial. En la reconstrucción-escritura, el sistema lee primero todos los datos no modificados sobre un grupo de entidades y, a continuación, calcula la nueva paridad utilizando esta información, así como los nuevos datos escritos [238]. Por último, escribe esta nueva paridad en distintos servidores con nombres diferentes, de modo que si alguien quiere acceder a un elemento desde su rango original necesitaría tanto las versiones antiguas como las actualizadas de esos elementos almacenadas en máquinas distintas. Aunque RMW (Read Modify Write) requiere más recursos de I/O que RMW para cada pequeña cantidad de escrituras, reduce el tráfico de red porque las cargas de trabajo empresariales suelen estar dominadas por cambios diminutos en lugar de grandes [44].

Las escrituras basadas en registros son más eficientes que las escrituras in situ cuando se escriben datos en sistemas de almacenamiento en clúster. Esto se debe a que las operaciones de búsqueda para las escrituras basadas en registros son mucho menos costosas que las correspondientes operaciones de escritura in situ. Los sistemas de almacenamiento en clúster actuales, como Azure [40] [130] [43] y HDFS [239] [236], utilizan escrituras basadas en registro para reducir la sobrecarga de búsqueda en escenarios de escritura intensiva de pequeño tamaño.

Los sistemas de almacenamiento empresarial utilizan RMW (Read Mo-

dify Write) y escrituras basadas en registros para acelerar las actualizaciones parciales [238]. El registro completo FL (Full Logging) mitiga la sobrecarga de lectura en disco de los fragmentos de datos anexando los últimos datos, el fragmento de paridad y los registros delta. Es decir, después de enviar los últimos datos al servidor de base de datos, este envía un delta de paridad a su servidor de paridad; a continuación, añade el nuevo contenido al final de su archivo de registro [234].

De este modo, cualquier lectura que se produzca posteriormente únicamente tendrá que acceder a los archivos adicionales una vez, en lugar de varias veces por actualización, como ocurriría con los métodos de escritura normales. Sin embargo, (FL) incurre en muchas búsquedas cuando se realizan lecturas secuenciales lo que puede ser especialmente costoso en escenarios dónde muchas pequeñas peticiones de escritura ocurren frecuentemente. También [125], utiliza un esquema de escritura parcial especulativa (PARIX) que admite pequeñas escrituras rápidas en sistemas de almacenamiento con código de borrado.

PARIX Block Storage (PBS) [248], es un eficiente sistema de almacenamiento en bloque que proporciona un servicio de disco virtual de alto rendimiento para máquinas virtuales que ejecutan aplicaciones no dependientes de la nube. PBS no sólo soporta escrituras parciales rápidas, sino que también realiza escrituras completas eficientes, repetición de diario en segundo plano y recuperación rápida de fallos con fuertes garantías de consistencia. Es decir, transforman la fórmula original de cálculo de paridad para utilizar los deltas de datos (entre los valores de datos actuales / originales), en lugar de los deltas de paridad, para calcular las paridades en la repetición del diario. Para cada escritura parcial, esto permite a PARIX registrar especulativamente sólo el nuevo valor de los datos sin leer su valor original [227] [234] [16].

6.2. Antecedentes

En esta sección se presenta en tres subsecciones. En la primera se presenta la esencia de genome y la relación en los últimos años con las tecnologías de la información y comunicación; posteriormente se tratan los algoritmos de compresión sin pérdida de datos que son fundamentales para esta investigación y; finalmente se detalla la librería Python erasure coding library, la cual contiene los algoritmos de código de borrado estudiados.

6.2.1. Genomas

El genoma es el conjunto del material hereditario de un organismo, la secuencia de nucleótidos que especifican las instrucciones genéticas para el desarrollo y funcionamiento del mismo y que son transmitidas de generación en generación, de padres a hijos. En él, además de los genes propiamente dichos, se incluyen regiones espaciadoras, regiones reguladoras, restos de genes antaño funcionales y muchas otras secuencias de función o papel todavía desconocido, si es que tienen alguno. De hecho, en el genoma humano, apenas el 1,5 % del material hereditario tiene una función codificante, es decir, corresponde a lo que solemos entender por genes. Por tanto, el genoma de un organismo es el depositario de la información que permite que cada organismo se desarrolle y responda a las exigencias impuestas por el medio. Pero, además, el genoma es depositario de los cambios que, a lo largo de la historia de la especie correspondiente y de todas sus antecesoras, han permitido su supervivencia hasta nuestros días. En consecuencia, en el genoma se almacena información de dos tipos: una de inmediata utilidad para el organismo y otra que sirve como registro histórico de éste y de sus ancestros (véase. Fig. 6.1). Ambos tipos de información son explotados por la biología actual, tanto en su vertiente funcional como en la histórica o evolutiva [68].

El ADN (ácido desoxirribonucleico) es una biomolécula que se replica

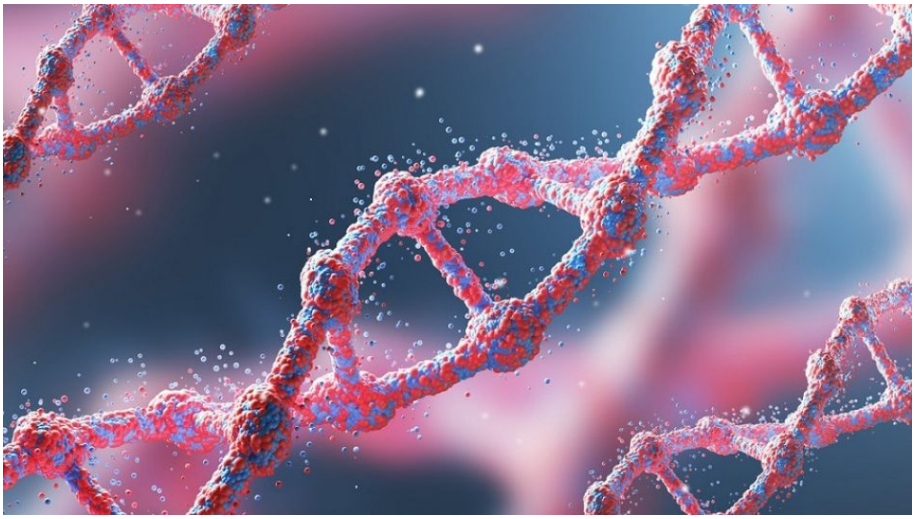


Figura 6.1: Genoma, secuenciación genómica [92]

en el núcleo de la células, se transcribe como ARNm y se codifica como proteína. Dentro del ADN, la información genómica de los seres vivos se encuentra almacenada en formato de secuencia por bases químicas que son la adenina (A), timina (T), citosina (C) y guanina (C) cada base química está unida a una molécula de desoxirribosa (azúcar) y una molécula de fosfato, cada complemento entre una base química, azúcar y grupo fosfato se denomina nucleótido. Los nucleótidos forman parte estructural de una secuencia de ADN que provee información genómica es decir una secuencia completa de ADN se denomina genoma [201].

El ADN está almacenado en los genes y estos a su vez se encuentran en los diferentes loci de cada cromosoma, el número de cromosomas varia según sea de dominio taxonómico eucariota (plantas, animales y el ser humano) o procariota (bacterias y arqueobacterias). Los organismos eucariotas se consideran organismos superiores debido a su complejidad ya que poseen mayor número de organelos en la célula y esto hace que su genoma sea de gran tamaño considerando el número de nucleótidos que se encuentran [201].

Las bases de datos biológicas almacenan información de secuencias de nucleótidos y proteínas, se subdividen en dos grupos: primarias, secundarias y compuesta. Las bases de datos primarias corresponden a las bases de datos que almacenan información genómica representada en nucleótidos o aminoácidos mientras que las bases de datos secundarias corresponden a las bases de datos que almacenan la funcionalidad de la información obtenida a partir de una base de datos primaria. Las bases de datos compuestas se refieren aquellas que combinan información de bases de datos primarías con secundarias y permiten realizar diferentes tipos de análisis por ejemplo NCBI (National Center for Biotecnology Information) posee un conjunto de bases de datos dónde se puede encontrar secuencias, taxonomía, genomas, mutaciones y se puede analizar dicha información [23].

Hoy en día, los genomas son almacenados en bases de datos dónde se pueden encontrar fragmentados en ficheros por regiones de estudio de relevante, es decir ficheros con información específica a analizar. Almacenar y analizar genomas completos representa un problema debido al gran tamaño que poseen y consumo de recursos de hardware como CPU, memoria RAM y almacenamiento que representan.

6.2.2. Algoritmos de compresión sin pérdidas.

6.2.2.1. zstd

Zstandard también conocido como zstd es un programa de compresión de datos en tiempo real rápido y gratuito de código abierto con mejores relaciones de compresión, desarrollado por Facebook. Es un algoritmo de compresión sin pérdidas escrito en Lenguaje C. También hay una reimplementación en Java, por lo que es un programa nativo del sistema operativo Linux [77].

Cuando sea necesario, puede cambiar la velocidad de compresión por relaciones de compresión más fuertes (la compensación entre la velocidad de compresión y la relación de compresión se puede configurar mediante

pequeños incrementos), y viceversa [33]. Tiene un modo especial para la compresión de datos pequeños, conocido como compresión de diccionario, y puede crear diccionarios a partir de cualquier conjunto de muestra proporcionado [10]. Viene con una utilidad de línea de comandos para crear y decodificar archivos .zst, .gz, .xz y .lz4.

Es importante destacar que Zstandard tiene una rica colección de API, es compatible con casi todos los lenguajes de programación populares, incluidos Python, Java, JavaScript, Nodejs, Perl, Ruby, C #, Go, Rust, PHP, Switft y muchos más.

Se utiliza activamente para comprimir grandes volúmenes de datos en múltiples formatos y casos de uso en Facebook, servicios como el almacenamiento de datos de Amazon Redshift, bases de datos como Hadoop y Redis, la red Tor y muchas otras aplicaciones, incluidos juegos [17].

6.2.2.2. gzip

GNU Zip, o GZIP, es una forma de compresión de datos sin pérdidas, de forma que es posible reducir el tamaño de todos los elementos de una web sin que con ello se pierda información o funcionalidades. Basada en el algoritmo DEFLATE, la compresión GZIP coge la información en bruto y reduce al máximo el tamaño de todos esos archivos [12] [11].

Para ello, primero se buscan patrones para eliminar los datos redundantes mediante el algoritmo LZ77; a continuación, se sustituyen las cadenas repetidas por una lista de elementos o tupla. Esa información se comprimirá aún más con el algoritmo de codificación Huffman, que asigna menos bits a los elementos que más se repiten, dejando más bits a los caracteres únicos o menos redundantes. Con estos datos se crea el llamado árbol de Huffman, este sitúa en la zona más alta los valores más repetidos que tienen un valor igual al de resto de caracteres que forman la cadena [245][199].

A partir del árbol se generará el código de Huffman que consiste en convertir las ramas en 0 y 1 y trazar los caminos, desde la raíz a las hojas,

uniendo todos los números. Esto genera códigos binarios que pueden variar en función de las frecuencias, lo que no se altera es el tamaño de la secuencia [162][57].

6.2.2.3. zip

ZIP o zip es un formato de compresión sin pérdida, muy utilizado para la compresión de datos como documentos, imágenes o programas. ZIP es un formato de fichero bastante simple, que comprime cada uno de los archivos de forma separada. Comprimir cada archivo independientemente del resto de archivos comprimidos permite recuperar cada uno de los ficheros sin tener que leer el resto, lo que aumenta el rendimiento [69] . El problema, es que el resultado de agrupar un número grande de pequeños archivos es siempre mayor que agrupar todos los archivos y comprimirlos como si fuera uno sólo. A cambio, esto permite extraer cada archivo de forma independiente sin tener que procesar el archivo desde el principio.

La especificación de ZIP indica que cada archivo puede ser almacenado, o bien sin comprimir, o utilizando una amplia variedad de algoritmos de compresión. Sin embargo, en la práctica, ZIP se suele utilizar casi siempre con el algoritmo de Phil Katz [121].

ZIP soporta un sistema de cifrado simétrico basado en una clave única. Sin embargo, este sistema de cifrado es débil ante ataques de texto plano, ataque de diccionario y fuerza bruta. También soporta distribuir las partes de un archivo comprimido en distintos medios de almacenamiento secundario [15].

Con el tiempo, se han ido incluyendo nuevas características, como nuevos métodos de cifrado. Sin embargo, estas nuevas características no están soportadas por las aplicaciones más utilizadas.

6.2.2.4. 7z

7z o 7-zip es un compresor de archivos gratuito y de código libre, al igual que Linux, tiene una licencia GNU LGPL y usa el formato de archivos 7z, que también es libre, con extensión de nombre igual al formato de compresión .7z [190] . Los principales formatos de compresión son LZMA, LZMA2 y PPMD, el LZMA es una mejora del algoritmo LZ77, el LZMA2 es una mejora del propio LZMA y PPMD incluye unos pequeños cambios al PPMdH y está más orientado a textos [222] .

7z es un formato de compresión de datos sin pérdida, con tasas muy altas que superan a las de los populares formatos zip y rar [187] [191].

6.2.3. Python erasure coding library

La tecnología de código de borrado (Erasure Code) utiliza el algoritmo del código de borrado para codificar los datos originales para redundancia y almacenar los datos y la redundancia para lograr tolerancia a fallas [112] [123]. La idea básica es que los elementos de datos originales del bloque n se calculan para obtener m elementos redundantes (bloques de verificación).

Para un elemento del bloque $n + m$, cuando se produce cualquier error en el elemento del bloque m (incluidos los datos originales y los datos redundantes), los datos originales del bloque n pueden recuperarse utilizando un algoritmo de reconstrucción correspondiente [24].

El proceso de generar la suma de verificación se codifica como (encode), y el proceso de restaurar el bloque de datos perdido se denomina decodificación (decode). El uso del disco es $\frac{n}{n+m}$. En comparación con el método de copia múltiple, el método basado en el código de borrado tiene las ventajas de baja redundancia y alta utilización del disco [99].

PyECLib es una biblioteca de codificación de borrado de Python diseñada y escrita originalmente como parte del esfuerzo por agregar soporte (Erasure Code) al proyecto Swift [41] [216], sin embargo, es un proyecto independiente. La biblioteca proporciona una interfaz de Python simple y

Tabla 6.1: Descripción de los ficheros utilizados

Code	Genomes	ID	Original size
A	Gorilla gorilla	2156	3.1 GB
B	Capra hircus	10731	3.0 GB
C	Bos taurus	82	2.7 GB
D	Felis catus	78	2.5 GB
E	Canis lupus familiaris	85	2.4 GB
F	Homo sapiens	51	3.3 GB

bien definida e implementa internamente una arquitectura de complemento que le permite aprovechar muchas bibliotecas de Lenguaje C conocidas como:

- liberasurecode rs vand: Vandermonde Reed-Solomon encoding, software-only backend implemented by liberasurecode
- jerasure rs vand: Vandermonde Reed-Solomon encoding, based on Jerasure
- jerasure rs cauchy: Cauchy Reed-Solomon encoding (Jerasure variant), based on Jerasure
- flat xor hd 3, flat xor hd 4: Flat-XOR based HD combination codes, liberasurecode
- isa l rs vand: Intel Storage Acceleration Library (ISA-L) - SIMD accelerated Erasure Coding backends
- isa l rs cauchy: Cauchy Reed-Solomon encoding (ISA-L variant)

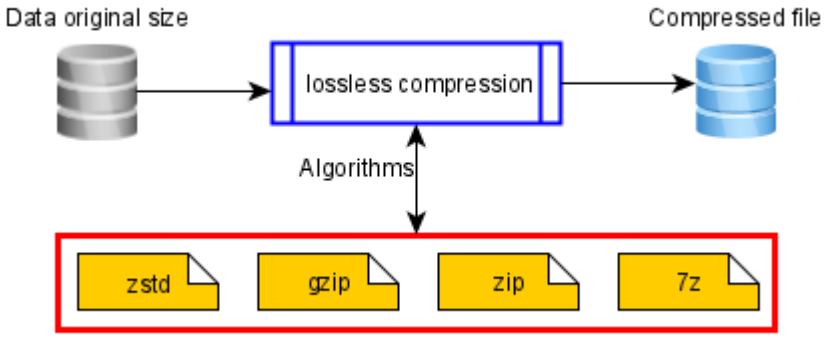


Figura 6.2: Algoritmos de compresión sin pérdidas

6.3. Materiales y métodos

En esta sección, describimos dos pasos para el proceso de configuración. En primer lugar, se describe la configuración del hardware para las pruebas de rendimiento realizadas. En segundo lugar, se describe la configuración del software implementado en el sistema operativo Linux Ubuntu 18.04.6 LTS basado en Debian.

En el caso del software, se utilizaron ficheros grandes que contienen información de genomas (véase. Tabla 6.1), debido al tamaño de los ficheros fue necesario aplicar algoritmos de compresión sin perdida (véase. Fig. 6.2) y evaluar el rendimiento de los algoritmos de códigos de borrado de la biblioteca PyECLib (véase. Fig. 6.3), dónde valores para $k = 10$ y $m = 5$ se asignan según la configuración de codificación y decodificación.

6.3.1. Hardware

La evaluación se ha realizado en un Sistema Cisco UCS 5108-AC2 Blade Server Chassis, con procesador Intel(R) Xeon(R) CPU E5-2630 v4 @ 2,2 GHz, memoria RAM de 128GB y unidad de almacenamiento SCSI con capacidad de almacenamiento de 1,2 TB, velocidad de rotación de disco

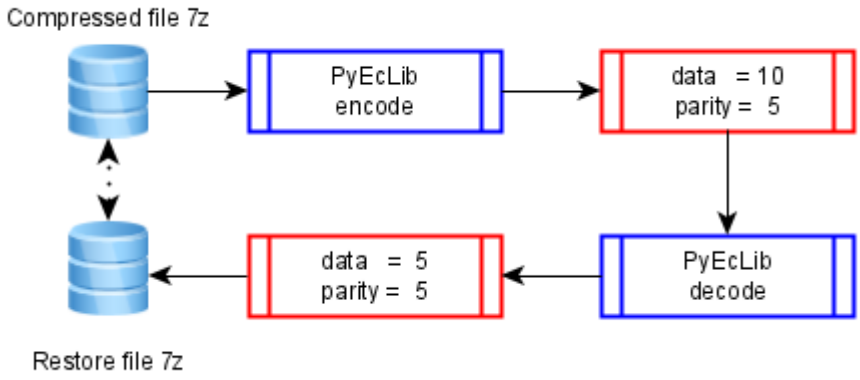


Figura 6.3: Algoritmos de codificación de borrado - PyECLib

duro de 10.000 RPM velocidad de transferencia interfaz del disco de 12 Gbps.

6.3.2. Software

La principal propuesta de este trabajo radica en la implementación de una capa resiliente de almacenamiento, que se ejecutará en segundo plano, fuera del control interactivo de los usuarios del sistema. Esta capa permitirá tener tolerancia ante el daño total o parcial de los ficheros. Para ello primeramente se realizó el rendimiento de los algoritmos de compresión sin perdida, una vez obtenido el mejor rendimiento en almacenamiento, este fichero comprimido será evaluado por los diferentes algoritmos de códigos de borrado que forman parte de la librería PyECLib.

Para aquellos familiarizados con la codificación de borrado, dos medidas de rendimiento importantes son: eficiencia de almacenamiento y tolerancia a fallos. La codificación de borrado de datos implica una compensación entre los dos. La eficiencia de almacenamiento es un indicador de almacenamiento adicional requerido para asegurar la resiliencia, mientras que la tolerancia

a fallos es un indicador de la posible recuperación en el caso de fallo de los elementos.

Esta capa resiliente de almacenamiento, tendrá la eficiencia del almacenamiento distribuido, es decir su almacenamiento adicional estará en diferentes buckets. Un bucket es un contenedor de objetos, que está diseñada para ofrecer un nivel de durabilidad de 99,999999999 % de los objetos en diferentes zonas de disponibilidad.

6.4. Resultados

Como parte de la propuesta planteada en esta investigación, el entorno construido requería la evaluación previa en 02 etapas:

- Evaluar el rendimiento de los algoritmos de compresión sin perdida con énfasis en la reducción del tamaño del almacenamiento.
- Medir los tiempos de la tolerancia a fallos mediante el uso de los algoritmos de códigos de borrado de la biblioteca PyECLib.

Con estos resultados se procederá a la etapa 3. Es decir, la implementación de la capa resiliente de almacenamiento que tendrá como finalidad su despliegue en una arquitectura distribuida.

6.4.1. Primera etapa

Para la primera etapa se utilizaron los ficheros originales de los genomas descritos en la Tabla 6.1, a estos ficheros fueron aplicados los algoritmos de compresion sin perdida como son: zstd, gzip, zip y 7z (véase. Tabla 6.2), tal como se muestra en la Fig. 6.4, el mejor rendimiento en terminos de reducción de tamaño de fichero es el algoritmo 7z.

También, en la Fig. 6.4, se describe en el eje de las abscisas los genomas representados con su respectivo código y en el eje de las ordenadas se

representó el tamaño en Gigabytes GB (véase. Tabla 6.1). En base al algoritmo 7z aplicado se observó que en el genoma A se alcanzó un 76,78 % de reducción del tamaño original, en el genoma B se alcanzó 80,00 % de reducción del tamaño original, en el genoma C se alcanzó un 76,29 % de reducción del tamaño original, en el genoma D se alcanzó un 75,20 % de reducción del tamaño original, en el genoma E se alcanzó un 74,16 % de reducción del tamaño original y en el genoma F se alcanzó un 76,96 % de reducción del tamaño original.

6.4.2. Segunda etapa

Para la segunda etapa se utilizó el fichero más grande 779,8 Megabytes (MB) con compresión 7z. En el proceso de codificación, los ajustes de datos originales $k = 10$ y el tamaño del bloque de paridad $m = 5$ para la recuperación de los bloques perdidos. En las (véase. Figuras. 6.5, 6.6, 6.7, 6.8, 6.9, 6.10, 6.11), el eje de las abscisas se ubican los genomas representados con su respectivo código y en el eje de las ordenadas se representó el consumo de tiempo en segundos, para cada proceso encode y decode respectivamente.

Experimento 1.- El rendimiento del algoritmo liberasure code rs vand erasure code, para el proceso encode alcanzó 15,8 segundos, como resultado de este proceso se han obtenido 15 fragmentos, cada uno de 74,37 MB, con un tamaño total de 1,08 GB incluidos los fragmentos de paridad (véase. Tabla 6.3) y para el proceso decode alcanzó 14,0 segundos, utilizando 5 fragmentos de data y 5 fragmentos de paridad (véase. Fig. 6.5).

Experimento 2.- El rendimiento del algoritmo jerasure code rs vand erasure code, para el proceso encode alcanzó 4,0 segundos, como resultado de este proceso se han obtenido 15 fragmentos, cada uno de 74,37 MB, con un tamaño total de 1,08 GB incluidos los fragmentos de paridad (véase. Tabla 6.3) y para el proceso decode alcanzó 3,3 segundos, utilizando 5 fragmentos de data y 5 fragmentos de paridad (véase. Fig. 6.6).

Experimento 3.- El rendimiento del algoritmo jerasure code rs cauchy

erasure code, para el proceso encode alcanzó 4,2 segundos, como resultado de este proceso se han obtenido 15 fragmentos, cada uno de 74,37 MB, con un tamaño total de 1,08 GB incluidos los fragmentos de paridad (véase. Tabla 6.3) y para el proceso decode alcanzó 3,5 segundos, utilizando 5 fragmentos de data y 5 fragmentos de paridad (véase. Fig. 6.7).

Experimento 4.- El rendimiento del algoritmo flat xor hd 3 erasure code, para el proceso encode alcanzó 3,4 segundos, como resultado de este proceso se han obtenido 15 fragmentos, cada uno de 74,37 MB, con un tamaño total de 1,08 GB incluidos los fragmentos de paridad (véase. Tabla 6.3) y para el proceso decode alcanzó 2,4 segundos, utilizando 5 fragmentos de data y 5 fragmentos de paridad (véase. Fig. 6.8).

Experimento 5.- El rendimiento del algoritmo flat xor hd 4 erasure code, para el proceso encode alcanzó 3,5 segundos, como resultado de este proceso se han obtenido 15 fragmentos, cada uno de 74,37 MB, con un tamaño total de 1,08 GB incluidos los fragmentos de paridad (véase. Tabla 6.3) y para el proceso decode alcanzó 2,4 segundos, utilizando 5 fragmentos de data y 5 fragmentos de paridad (véase. Fig. 6.9).

Experimento 6.- El rendimiento del algoritmo isa l rs vand erasure code, para el proceso encode alcanzó 3,3 segundos, como resultado de este proceso se han obtenido 15 fragmentos, cada uno de 74,37 MB, con un tamaño total de 1,08 GB incluidos los fragmentos de paridad (véase. Tabla 6.3) y para el proceso decode alcanzó 2,6 segundos, utilizando 5 fragmentos de data y 5 fragmentos de paridad (véase. Fig. 6.10).

Experimento 7.- El rendimiento del algoritmo isa l rs cauchy erasure code, para el proceso encode alcanzó 3,2 segundos, como resultado de este proceso se han obtenido 15 fragmentos, cada uno de 74,37 MB, con un tamaño total de 1,08 GB incluidos los fragmentos de paridad (véase. Tabla 6.3) y para el proceso decode alcanzó 2,8 segundos, utilizando 5 fragmentos de data y 5 fragmentos de paridad (véase. Fig. 6.11).

6.4.3. Tercera etapa

En la arquitectura utilizada para la implementación de la capa resiliente de almacenamiento (véase. Fig. 6.12), se pueden observar los siguientes elementos:

- **Cisco UCS 5108** es un hardware de altas prestaciones utilizado en nuestra investigación para medir el rendimiento de la capa resiliente de almacenamiento distribuida.
- **Amazon DynamoDB** es una base de datos NoSQL de clave-valor sin servidor y completamente administrada que está diseñada para ejecutar aplicaciones de alto rendimiento a cualquier escala, usada en nuestra investigación para guardar la huella digital del fichero original.
- **Amazon Quantum Ledger Database (QLDB)** es una base de datos de libro mayor completamente administrada, aplicada en nuestra investigación para proporcionar un registro de transacciones transparente, inmutable y que se puede verificar mediante criptografía.
- **Amazon Simple Storage Service (Amazon S3)** es un servicio de almacenamiento de objetos que ofrece escalabilidad, disponibilidad de datos, seguridad y rendimiento. Destinada en nuestra investigación para almacenar los fragmentos del proceso de encode.
- **Buckets** es un contenedor para objetos almacenados en Amazon S3. En nuestra investigación están desplegados tres contenedores en diferentes regiones.

La capa resiliente de almacenamiento (véase. Fig. 6.13) fue desarrollada para ejecutarse como una tarea programada independiente del sistema, los ficheros nuevos aquellos que ingresaron al sistema dentro de las últimas 24 horas serán procesados, a continuación se describe su funcionamiento:

- Luego de listar todos los ficheros que serán procesados por la capa resiliente, procedemos a obtener la huella digital de cada uno de ellos mediante el algoritmo SHA-512, cada huella que se obtiene se guarda en la base de datos DynamoDB.

- También de cada fichero, aplicamos compresión sin perdida especificamente el algoritmo 7z, que presenta los mejores ratios de compresión (véase. Tabla 6.2) (véase. Fig. 6.4).

- Además de cada fichero, aplicamos el algoritmo flat xor hd 4 erasure code que presta los mejores tiempos de encode (véase. Fig. 6.9). Con el fin de mantener un registro inmutable utulizamos Amazon Quantum Ledger Database, aquí se guardará todas las transacciones de cada fragmento de la técnica encode proceso del algoritmo flat xor hd 4 erasure code.

- Posteriormente los fragmentos son enviados en grupos de 5, al servicio S3 de Amazon, en 3 Buckets de diferentes regiones, esto con el fin de mitigar la perdida de cada fragmento al 0,000000001 %

Los registros de huellas digitales de los ficheros están almacenadas localmente en la base de datos DynamoDB y sincronizadas en Amazon Quantum Ledger Database, pero cuándo uno o varios ficheros se dañan total o parcialmente, la tolerancia fallos en la capa resiliente de almacenamiento (véase. Fig. 6.13) entra en ejecución y la misma se describe a continuación:

- Con el fin de listar cuántos ficheros estan deteriorados, la capa resiliente ejecuta un consulta a Amazon Quantum Ledger Database obteniendo las huellas digitales de los ficheros deteriorados y la ubicación de sus fragmentos en los Buckets de Amazon S3.

- Una vez obtenida esta información aplicamos el algoritmo flat xor hd 4 erasure code que presta los mejores tiempos de decode (véase. Fig.

Tabla 6.2: Tamaño del archivo utilizado en la compresión sin pérdidas

Código	**Tamaño Original**	**zstd**	**gzip**	**zip**	**7z**
A	3.1 GB	0.96 GB	0.88 GB	0.83 GB	0.72 GB
B	3.0 GB	0.92 GB	0.87 GB	0.82 GB	0.66 GB
C	2.7 GB	0.87 GB	0.82 GB	0.73 GB	0.64 GB
D	2.5 GB	0.80 GB	0.75 GB	0.67 GB	0.62 GB
E	2.4 GB	0.79 GB	0.73 GB	0.65 GB	0.62 GB
F	3.3 GB	1.00 GB	0.95 GB	0.89 GB	0.76 GB

6.9). Cabe indicar que solo necesitamos $n = 10$ fragmentos de cada fichero, para recuperar el o los ficheros dañados.

- Luego de recuperar el o los ficheros dañados, aplicamos el algoritmo SHA-512 con el fin de garantizar la integridad del fichero recuperado, esta huella digital debe ser igual a los registros de Amazon Quantum Ledger Database.
- Posteriormente se sincronizan las bases de datos DynamoDB y Amazon Quantum Ledger Database con el fin de mantener los registros del grupo de ficheros del sistema.

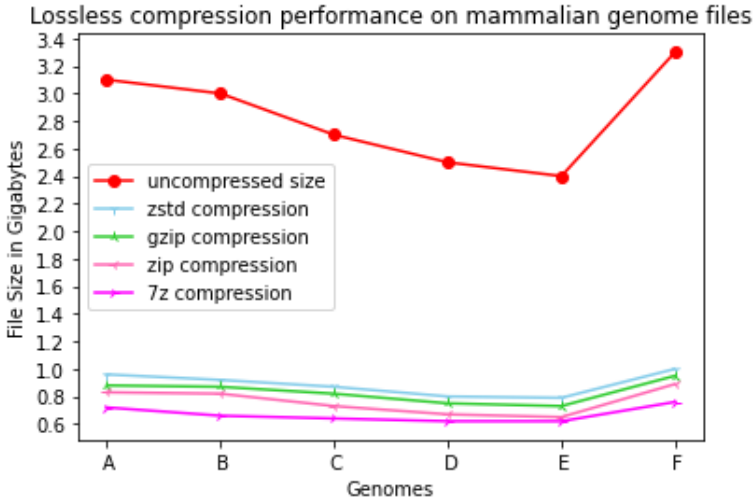


Figura 6.4: Compresión sin pérdidas, mejor 7z.

Tabla 6.3: Tamaño del archivo utilizado en el entorno de prueba para codificar k = 10, m = 5

Código	**Compresión 7z**	**Tamaño de los fragmentos**	**Tamaño Total**
A	740.1 MB	70.57 MB	1.03 GB
B	683.4 MB	65.17 MB	0.95 GB
C	658.6 MB	62.80 MB	0.92 GB
D	644.9 MB	61.50 MB	0.90 GB
E	635.6 MB	60.61 MB	0.88 GB
F	779.8 MB	74.37 MB	1.08 GB

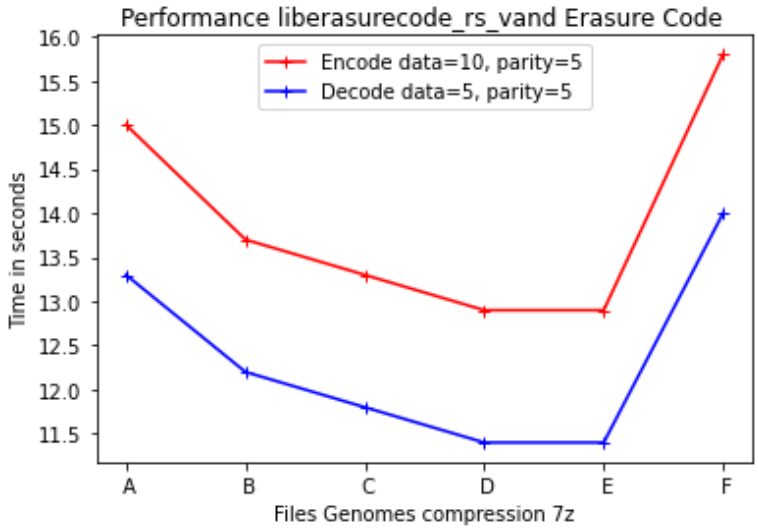


Figura 6.5: Liberasure code rs vand, código de borrado.

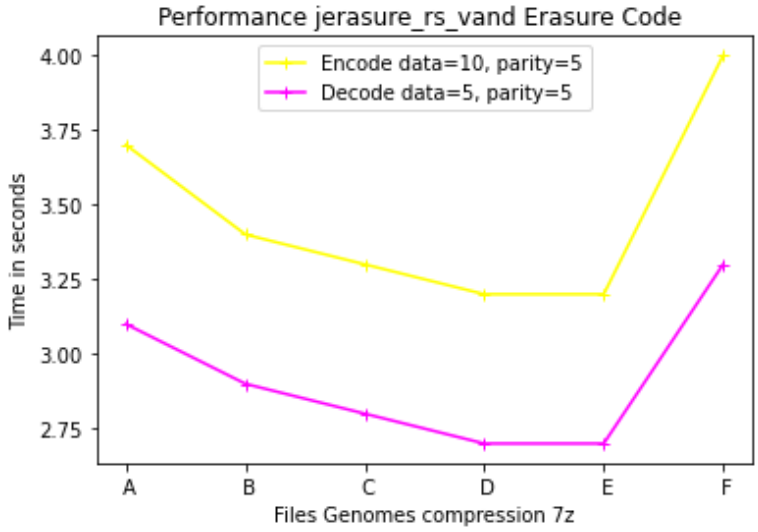


Figura 6.6: Jerasure rs vand, código de borrado.

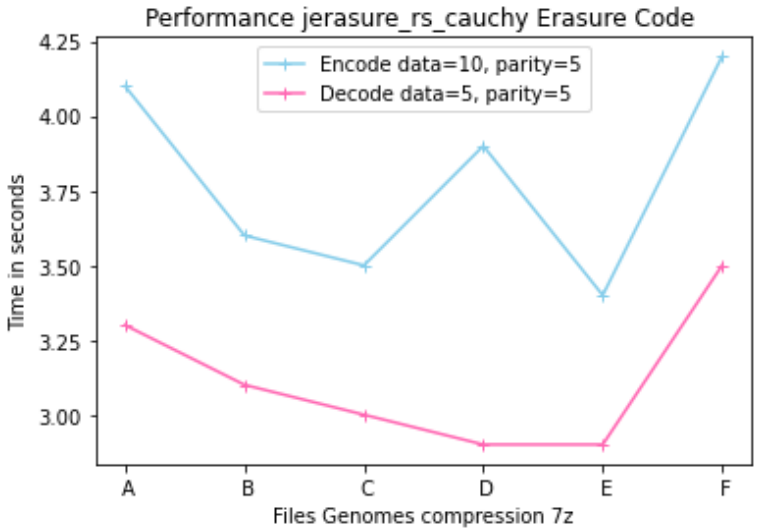


Figura 6.7: Jerasure rs cauchy, código de borrado.

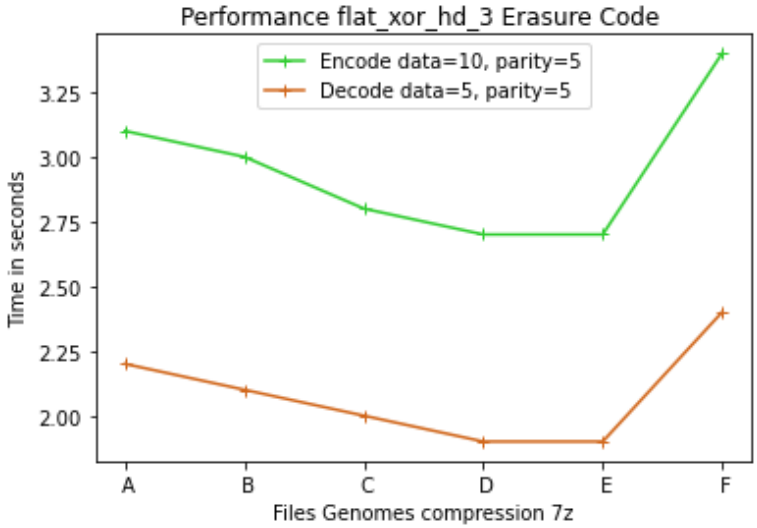


Figura 6.8: Flat xor hd 3, código de borrado.

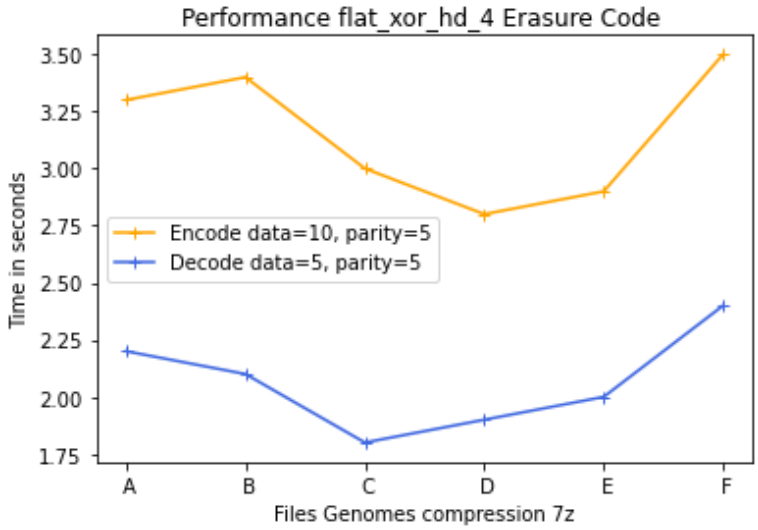


Figura 6.9: Flat xor hd 4, código de borrado.

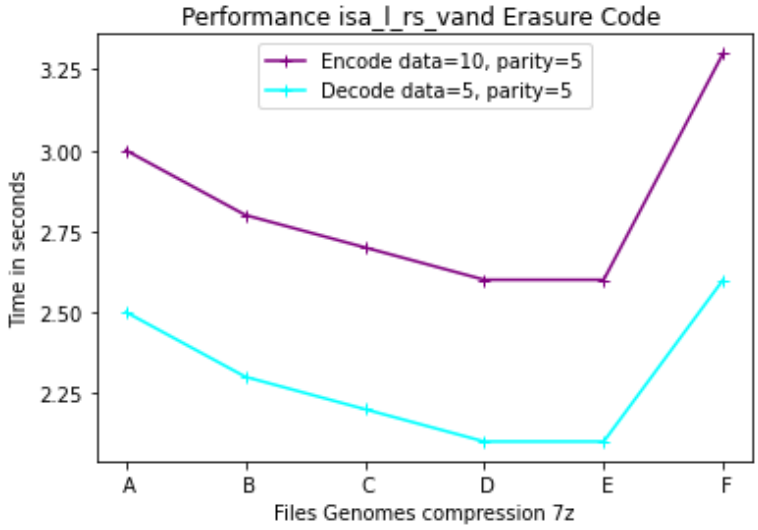


Figura 6.10: Isa l rs vand, código de borrado.

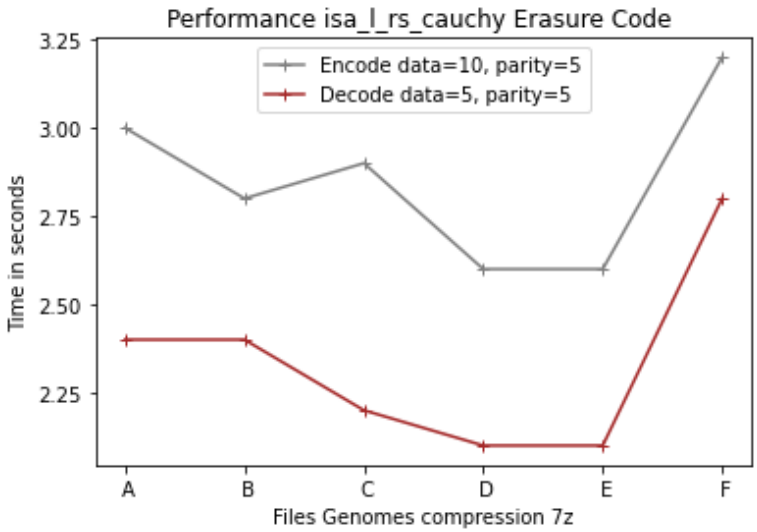


Figura 6.11: Isa l rs cauchy, código de borrado.

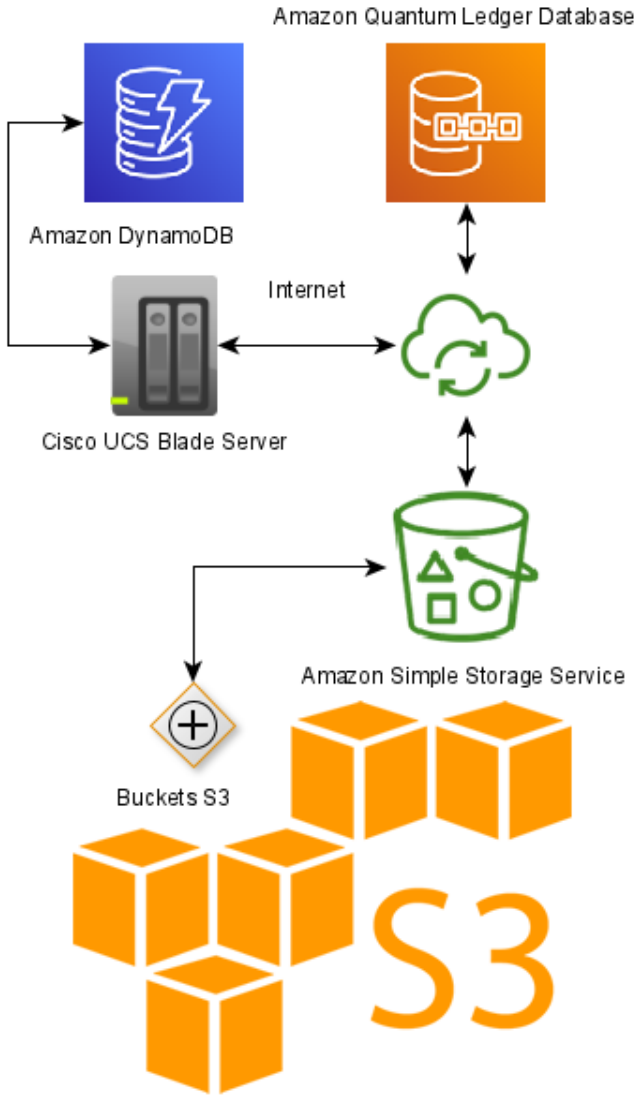


Figura 6.12: Arquitectura utilizada para validar la capa de almacenamiento distribuido resiliente.

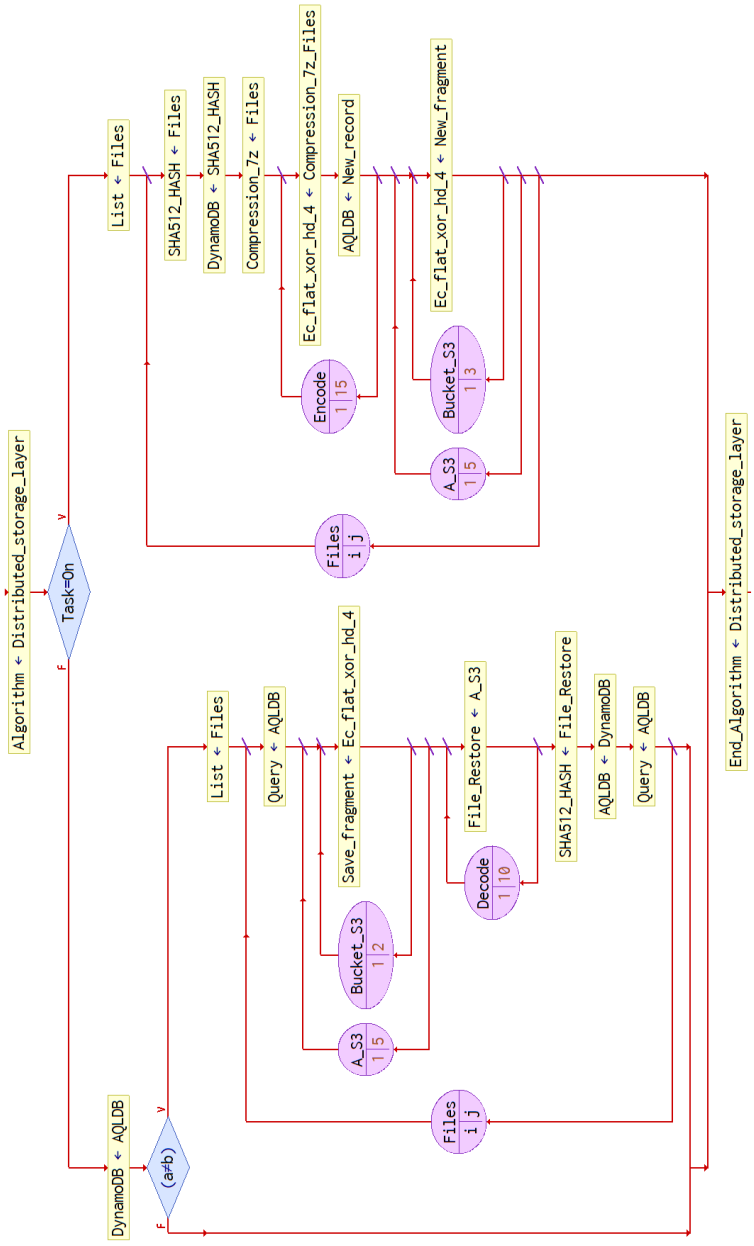


Figura 6.13: Algoritmo para la capa de almacenamiento distribuido resiliente.

Capítulo 7

Conclusiones y trabajos futuros

7.1. Conclusiones

Para lograr la implementación de algoritmos de código de borrado que permitan la resiliencia en los sistemas de almacenamiento distribuido. Primeramente, se ha realizado la revisión del estado de la bibliografía actual de las técnicas de códigos de borrado. Acto seguido se resumen las principales conclusiones y desarrollos realizados para optimizar la resiliencia en el marco de los sistemas de almacenamiento distribuido:

- El plugin nativo de almacenamiento distribuido para Dicoogle PACS, soporta una configuración de tolerancia a fallos de $m = 2; 3; 5$, que es superior a la propuesta en la tecnología nativa FlexProtect del sistema operativo OneFS. En pocas palabras, el sistema operativo OneFS utiliza la codificación Reed-Solomon para proporcionar redundancia y disponibilidad de datos. La tecnología FlexProtect soporta la protección de datos en hasta cuatro fallos simultáneos de nodos enteros o unidades individuales de la configuración. Esto significa que

FlexProtect es más eficiente que el RAID 6 utilizado habitualmente hoy en día cuando el objetivo es proporcionar disponibilidad de datos. Esta comparación se hace en términos del nivel máximo de RAID 6. Además, en la propuesta presentada en esta investigación, se ha incorporado al plugin de almacenamiento una función hash definida, que es un método para generar claves que representen unívocamente un fichero DICOM o un conjunto de datos de un fichero DICOM en este caso, este proceso opera matemáticamente como una función de compresión SHA-512 en ejecución sobre un conjunto de datos de cualquier longitud, y como resultado, genera una clave digital de un tamaño fijo que es independiente del tamaño del fichero DICOM original

- La batería de pruebas para la recuperación de ficheros de imágenes médicas dermatológicas, dentro de su peculiaridad se describen las compresiones realizadas que se basaron tanto en los componentes principales como en la cuasiperiodicidad de los primeros componentes principales. Se consiguieron buenas relaciones de compresión, manteniendo la calidad y las características relevantes. Además, la calidad de la compresión se midió utilizando el coeficiente pico de la relación señal-ruido y el error cuadrático medio de las diferencias entre la imágen original y la comprimida. Los resultados experimentales mostraron que la calidad de las imágenes comprimidas era satisfactoria y útil para que los médicos en esta investigación pudieran diagnosticar la enfermedad. Además, ante un evento adverso, daño, pérdida parcial o total de ficheros JPG (compresión con pérdida) o PNG (compresión sin pérdida), esta investigación propuso llevar a cabo la recuperación total de los ficheros dañados. Para ello, se utilizó el código de borrado Reed-Solomon de la librería zfec, con el fin de mitigar falsos positivos o negativos en el diagnóstico de la enfermedad.

- La capa de almacenamiento distribuida resiliente, fue implantada bajo un sistema operativo basado en Debian, en ejecución como una tarea programada cada 24 horas independiente del sistema, soporta una configuración de tolerancia a fallos $m = 5$, para validar la propuesta se utilizaron ficheros que contienen información de genomas, aunque funciona para cualquier tipo de fichero, en la primera etapa de experimentación utiliza algoritmos de compresión sin perdida, la experimentación muestra ratios de compresión sobre el 23 %, los tiempos de codificación y decodificación son óptimos 3,5 segundos y 2,4 segundos respectivamente con el fichero utilizado de ejemplo, la resiliencia radica en la distribución de los fragmentos, se utilizó 3 Buckets en diferentes regiones del servicio S3 de AWS mitigando al 0,000000001 % la perdida de cada fragmento, garantizando así la perpetuidad de los fragmentos. Adicionalmente, se registran los eventos en QLDB que es una base de datos de libro mayor que ofrece escalabilidad y disponibilidad de datos, cuyo valor agregado es la inmutabilidad.

7.2. Trabajos futuros

Desde mi perspectiva, hay muchas oportunidades de investigación entorno a las técnicas de códigos de borrado, desde la optimización de la eficiencia y la tolerancia a fallos hasta la implementación en hardware especializado y la integración con sistemas de almacenamiento distribuido. Estas investigaciones podrían mejorar la eficiencia del almacenamiento y la protección contra la pérdida de datos en diversos campos de la ciencia, desde el almacenamiento de datos en la nube hasta la transmisión de video en tiempo real.

Como trabajo de investigación futuro, la propuesta actual de investigación nos inspira a explorar otras implementaciones que utilicen tecnologías

de procesamiento gráfico dedicadas, como las GPU, para aprovechar la tecnología GPUDirect soportada por la GPU Nvidia.

La elección entre arquitecturas CPU, GPU y TPU dependerá de las necesidades y recursos de cada proyecto de investigación. La CPU es adecuada para tareas generales de computación, mientras que la GPU y TPU son especialmente útiles para tareas de procesamiento de gráficos y aprendizaje profundo. Sin embargo, la GPU es más económica que la TPU y más versátil, mientras que la TPU es más rápida y eficiente en el procesamiento de tareas de aprendizaje profundo.

Es decir, indistintamente de la tecnología y la técnica de código de borrado a usar el reto futuro será: mejorar la eficiencia de la codificación de borrado, optimizar de la tolerancia a fallos, implementar en hardware especializado, integrar con sistemas de almacenamiento distribuido y codificación de borrado en tiempo real.

Acrónimos

ACR	American College of Radiology
ADN	Deoxyribonucleic Acid
AFS	Andrew File System
aHDFS	An Erasure-Coded Data Archival System for Hadoop Clusters
AI	Artificial Intelligence
API	Application Programming Interface
AWS	Amazon Web Services
CBIR	Content Image Recovery System
CCPA	California Consumer Privacy Act
CD-ROM	Compact Disc Read Only Memory
CFS	Coda File System
CMOS	Complementary Metal Oxide Semiconductor
CPU	Central Processing Unit
CT	Computed Tomography
DC	Data Center
DFS	Distributed File System
DICOM	Digital Imaging and Communication in Medicine
DSS	Distributed Storage Service
DVD	Digital Video Disc
EC	Erasure Coding
EC2	Amazon Elastic Compute Cloud
FEC	Forward Error Correction
FL	Full Logging

FPGA	Field Programable Gate Array
GB	Gigabyte
GDPR	Reglamento General de Protecciń de Datos
GFS	Google File System
GNU	General Public License
HDD	Hard Disk Drive
HDFS	Hadoop Distributed File System
HPC	High Performance Computing
HPE	Hewlett Packard Enterprise
ICT	Information and Communication Technology
IEEE	Institute of Electrical and Electronics Engineers
IOPS	Input/Output Operations Per Second
IoT	Internet Of Things
ISAL	Intelligent Storage Library
iSCSI	Internet Small Computer Systems Interface
IT	Information Technology
JPG	Joint Photographic Experts Group
LAN	Local Area Network
LDPC	Low Density Parity Check
LR	Local Reconstruction
LRC	Local Reconstruction Codes
MBR	Minimum-Bandwidth Regenerating
MDS	Maximum Distance Separable
MinIO	Multi-Cloud Object Storage
MIT	Massachusetts Institute of Technology
MRI	Magnetic Resonance Imaging
MSCT	Multilayer Computed Tomography
MSE	Mean Squared Error
MSR	Reduction of Storage Overhead
MTTDL	Mean Time To Data Loss
NAS	Network Attached Storage

NCBI	National Center for Biotecnology Information
NEMA	National Electrical Manufacturers Association
NFS	Network File System
OneFS	Parallel Distributed Networked File System
OS	Operating System
P2P	Peer to Peer
PACS	Picture Archiving Communication System
PBS	PARIX Block Storage
PCA	Principal Component Analysis
PCIe	Peripheral Component Interconnect Express
PCs	Principal Components
PET	Positron Emission Tomography
PNG	Portable Network Graphics
PSNR	Peak Signal-to-Noise Ratio
QLDB	Amazon Quantum Ledger Database
RAID	Redundant Array of Independent Disks
RAM	Random Access Memory
RG	Regenerating Codes
RGB	Red Green Blue
RMW	Read Modify Write
RPM	Revolutions Per Minute
RS	Reed-Solomon
SAN	Storage Area Network
SAS	Server Attached Storage
SATA	Serial Advanced Technology Attachment
SBS	Scalable Block Store
SCSI	Small Computer System Interface
SDK	Software Development Kit
SEU	Single Event Upset
SHA-512	Secure Hash Algorithm
SPECT	Single-Photon Emission Computed Tomography

SSMD	Selectable Slice-thickness Multi-row Detector
TB	Terabyte
TCP	Transmission Control Protocol
US	Ultrasound
WSI	Whole Slide Imaging
ZB	Zettabyte

Bibliografía

[1] C. Aatish and M. A. U. Einstein. Performance enhancing secure erasure code (pesec) for cloud storage using random n blocks encryption. In *2021 8th IEEE International Conference on Cyber Security and Cloud Computing (CSCloud)/2021 7th IEEE International Conference on Edge Computing and Scalable Cloud (EdgeCom)*, pages 13–17. IEEE, 2021.

[2] Z. Abbas, H. Tayara, and K. to Chong. Spinenet-6ma: A novel deep learning tool for predicting dna n6-methyladenine sites in genomes. *IEEE Access*, 8:201450–201457, 2020.

[3] H. Abdi and L. J. Williams. Principal component analysis. *Wiley interdisciplinary reviews: computational statistics*, 2(4):433–459, 2010.

[4] A. A. Adeyemo, S. A. Bashir, A. D. Mohammed, and O. O. Abisoye. Impact of pixel scaling on classification accuracy of dermatological skin disease detection. In *2020 IEEE 2nd International Conference on Cyberspac (CYBER NIGERIA)*, pages 36–43. IEEE, 2021.

[5] A. Aghayev. Adopting zoned storage in distributed storage systems. *Ph. D. dissertation*, 2020.

[6] Z. G. Al-Mekhlafi, E. M. Senan, B. A. Mohammed, M. Alazmi, A. M. Alayba, A. Alreshidi, and M. Alshahrani. Diagnosis of

histopathological images to distinguish types of malignant lymphomas using hybrid techniques based on fusion features. *Electronics*, 11(18):2865, 2022.

[7] E. Aloupogianni, M. Ishikawa, N. Kobayashi, and T. Obi. Hyperspectral and multispectral image processing for gross-level tumor detection in skin lesions: a systematic review. *Journal of Biomedical Optics*, 27(6):060901, 2022.

[8] L. R. Alvarez and R. V. Solis. Dicom ris/pacs telemedicine network implementation using free open source software. *IEEE Latin America Transactions*, 11(1):168–171, 2013.

[9] H. P. Anvin. The mathematics of raid-6, 2007.

[10] M. Aslanyürek and A. Mesut. A static dictionary-based approach to compressing short texts. In *2021 6th International Conference on Computer Science and Engineering (UBMK)*, pages 342–347. IEEE, 2021.

[11] K. Aurangzeb, M. Alhussein, and M. OÑils. Analysis of binary image coding methods for outdoor applications of wireless vision sensor networks. *IEEE Access*, 6:16932–16941, 2018.

[12] K. Aurangzeb, M. Alhussein, and M. OÑils. Data reduction using change coding for remote applications of wireless visual sensor networks. *IEEE Access*, 6:37738–37747, 2018.

[13] S. Balaji, M. N. Krishnan, M. Vajha, V. Ramkumar, B. Sasidharan, and P. V. Kumar. Erasure coding for distributed storage: An overview. *Science China Information Sciences*, 61(10):1–45, 2018.

[14] M. Balakrishnan, A. Kadav, V. Prabhakaran, and D. Malkhi. Differential raid: Rethinking raid for ssd reliability. *ACM Transactions on Storage (TOS)*, 6(2):1–22, 2010.

[15] A. Bansal, S. Haque, and C. McMillan. Project-level encoding for neural source code summarization of subroutines. In *2021 IEEE/ACM 29th International Conference on Program Comprehension (ICPC)*, pages 253–264. IEEE, 2021.

[16] H. Bao and Y. Wang. Esdu: An elastic stripe-based delta update method for erasure-coded cross-data center storage systems. *Journal of Parallel and Distributed Computing*, 167:173–186, 2022.

[17] D. Barina. X3: Lossless data compressor. *arXiv preprint arXiv:2201.01727*, 2022.

[18] B. Beach. Backblaze open sources reed-solomon erasure coding source code, 2015.

[19] S. Bonechi, P. Andreini, A. Mecocci, N. Giannelli, F. Scarselli, E. Neri, M. Bianchini, and G. M. Dimitri. Segmentation of aorta 3d ct images based on 2d convolutional neural networks. *Electronics*, 10(20):2559, 2021.

[20] D. E. Booth. Multivariate statistical inference and applications, 1998.

[21] A. Boufahja, S. Nichols, and V. Pangon. Quantitative evaluation of pacs query/retrieve capabilities. *Journal of Digital Imaging*, 34(5):1302–1315, 2021.

[22] R. Bro and A. K. Smilde. Principal component analysis. *Analytical methods*, 6(9):2812–2831, 2014.

[23] A. N. Burian, W. Zhao, T.-W. Lo, and D. M. Thurtle-Schmidt. Genome sequencing guide: An introductory toolbox to whole-genome analysis methods. *Biochemistry and Molecular Biology Education*, 49(5):815–825, 2021.

[24] D. Burihabwa, P. Felber, H. Mercier, and V. Schiavoni. A performance evaluation of erasure coding libraries for cloud-based data stores: (practical experience report). In *Distributed Applications and Interoperable Systems: 16th IFIP WG 6.1 International Conference, DAIS 2016, Held as Part of the 11th International Federated Conference on Distributed Computing Techniques, DisCoTec 2016, Heraklion, Crete, Greece, June 6-9, 2016, Proceedings 16*, pages 160–173. Springer, 2016.

[25] A. Campanati, V. Brisigotti, F. Diotallevi, G. -DÁgostino", M. Paolinelli, G. Radi, G. Rizzetto, C. Sapigni, C. Tagliati, and A. Offidani. Active implications for dermatologists in "sars-cov-2 era": personal experience and review of literature. *Journal of the European Academy of Dermatology and Venereology*, 2020.

[26] P. A. Carr and G. M. Church. Genome engineering. *Nature biotechnology*, 27(12):1151–1162, 2009.

[27] CEPH. Locally repairable erasure code plugin. https://docs.ceph.com/docs/master/rados/operations/erasure-code?lrc/, último acceso: 17/04/2023.

[28] J. F.-W. Chan, A. J. Zhang, S. Yuan, V. K.-M. Poon, C. C.-S. Chan, A. C.-Y. Lee, W.-M. Chan, Z. Fan, H.-W. Tsoi, L. Wen, et al. Simulation of the clinical and pathological manifestations of coronavirus disease 2019 (covid-19) in a golden syrian hamster model: implications for disease pathogenesis and transmissibility. *Clinical infectious diseases*, 71(9):2428–2446, 2020.

[29] H.-b. Chen and S. Fu. Parallel erasure coding: exploring task parallelism in erasure coding for enhanced bandwidth and energy efficiency. In *2016 IEEE International Conference on Networking, Architecture and Storage (NAS)*, pages 1–4. IEEE, 2016.

[30] H.-b. Chen, G. Grider, J. Inman, P. Fields, and J. A. Kuehn. An empirical study of performance, power consumption, and energy cost of erasure code computing for hpc cloud storage systems. In *2015 IEEE International Conference on Networking, Architecture and Storage (NAS)*, pages 71–80. IEEE, 2015.

[31] H.-B. Chen, G. Grider, J. Inman, J. A. Kuehn, et al. The impact of vectorization on erasure code computing in cloud storages-a performance and power consumption study. In *2015 IEEE 8th International Conference on Cloud Computing*, pages 781–788. IEEE, 2015.

[32] H. C. Chen, Y. Hu, P. P. Lee, and Y. Tang. Nccloud: A network-coding-based storage system in a cloud-of-clouds. *IEEE Transactions on computers*, 63(1):31–44, 2013.

[33] J. Chen, M. Daverveldt, and Z. Al-Ars. Fpga acceleration of zstd compression algorithm. In *2021 IEEE International Parallel and Distributed Processing Symposium Workshops (IPDPSW)*, pages 188–191. IEEE, 2021.

[34] R. Chen. *Practical Algorithms for High-Performance Data Storage Systems*. PhD thesis, Wayne State University, 2021.

[35] R. Chen and L. Xu. Practical performance evaluation of space optimal erasure codes for high-speed data storage systems. *SN Computer Science*, 1(1):1–14, 2020.

[36] S. Chen and D. Towsley. A performance evaluation of raid architectures. *IEEE Transactions on computers*, 45(10):1116–1130, 1996.

[37] Y. Chen, S. Xie, and J. Zhang. A hybrid domain image encryption algorithm based on improved henon map. *Entropy*, 24(2):287, 2022.

[38] Y. Chen, Y. Zhou, S. Taneja, X. Qin, and J. Huang. ahdfs: An erasure-coded data archival system for hadoop clusters. *IEEE Transactions on Parallel and Distributed Systems*, 28(11):3060–3073, 2017.

[39] A. Chiniah and A. Mungur. Dynamic erasure coding policy allocation (decpa) in hadoop 3.0. In *2019 6th IEEE International Conference on Cyber Security and Cloud Computing (CSCloud)/2019 5th IEEE International Conference on Edge Computing and Scalable Cloud (EdgeCom)*, pages 29–33. IEEE, 2019.

[40] A. Chiniah and A. Mungur. On the adoption of erasure code for cloud storage by major distributed storage systems. *EAI Endorsed Transactions on Cloud Systems*, 7(21):e1–e1, 2022.

[41] V. Chouhan and S. K. Peddoju. Investigation of optimal data encoding parameters based on user preference for cloud storage. *IEEE Access*, 8:75105–75118, 2020.

[42] Colossus. Successor to the Google file System. http://www.highlyscalablesystems.com/3202/colossus/, último acceso: 17/04/2023.

[43] A. Datta, A. A. Fahreza, and F. Oggier. Qloc: Quorums with local reconstruction codes. *IEEE Access*, 9:93298–93314, 2021.

[44] A. Datta and F. Oggier. Concurrency control and consistency over erasure coded data. *IEEE Access*, 10:118617–118638, 2022.

[45] S. H. Dau, W. Song, and C. Yuen. On block security of regenerating codes at the mbr point for distributed storage systems. In *2014 IEEE International Symposium on Information Theory*, pages 1967–1971. IEEE, 2014.

[46] S. J. Devaraj. Emerging paradigms in transform-based medical image compression for telemedicine environment. In *Telemedicine Technologies*, pages 15–29. Elsevier, 2019.

[47] K. I. Diamantaras and S. Y. Kung. *Principal component neural networks: theory and applications*. John Wiley & Sons, Inc., 1996.

[48] B. Diaz-Guimaraens, M. Dominguez-Santas, A. Suarez-Valle, C. Pindado-Ortega, G. Selda-Enriquez, S. Bea-Ardebol, and D. Fernandez-Nieto. Petechial skin rash associated with severe acute respiratory syndrome coronavirus 2 infection. *JAMA dermatology*, 156(7):820–822, 2020.

[49] Dicoogle. Dicoogle Pacs Source Code. https://dicoogle.com/downloads/, último acceso: 17/04/2023.

[50] A. G. Dimakis, P. B. Godfrey, Y. Wu, M. J. Wainwright, and K. Ramchandran. Network coding for distributed storage systems. *IEEE transactions on information theory*, 56(9):4539–4551, 2010.

[51] A. G. Dimakis, K. Ramchandran, Y. Wu, and C. Suh. A survey on network codes for distributed storage. *Proceedings of the IEEE*, 99(3):476–489, 2011.

[52] M. N. C. Douglas Gantenbein, Senior Writer. A better way to store data. microsoft research blog. https://www.microsoft.com/en-us/research/blog/better-way-store-data/, último acceso: 17/04/2023.

[53] A. Duminuco and E. W. Biersack. Hierarchical codes: A flexible trade-off for erasure codes in peer-to-peer storage systems. *Peer-to-peer Networking and Applications*, 3(1):52–66, 2010.

[54] M. Eichelberg, K. Kleber, and M. Kämmerer. Cybersecurity in pacs and medical imaging: an overview. *Journal of Digital Imaging*, 33(6):1527–1542, 2020.

[55] N. E. El-Meligy, T. O. Diab, A. S. Mohra, A. Y. Hassan, and W. I. El-Sobky. A novel dynamic mathematical model applied in hash function based on dna algorithm and chaotic maps. *Mathematics*, 10(8):1333, 2022.

[56] R. S. Elagooz, A. Mahran, S. Gasser, and M. Aboul-Dahab. Efficient low-complexity decoding of ccsds reed–solomon codes based on "justesen'sçoncatenation. *IEEE Access*, 7:49596–49603, 2019.

[57] S. Elnady, S. Sayed, and A. Salah. Hadc: A hybrid compression approach for dna sequences. *IEEE Access*, 10:106841–106848, 2022.

[58] J. B. Elsner and A. A. Tsonis. *Singular spectrum analysis: a new tool in time series analysis*. Springer Science & Business Media, 1996.

[59] K. Ericson and S. Pallickara. Survey of storage and fault tolerance strategies used in cloud computing. In *Handbook of Cloud Computing*, pages 137–158. Springer, 2010.

[60] T. Ernvall, S. El Rouayheb, C. Hollanti, and H. V. Poor. Capacity and security of heterogeneous distributed storage systems. *IEEE Journal on Selected Areas in Communications*, 31(12):2701–2709, 2013.

[61] R. R. Expósito, R. Galego-Torreiro, and J. González-Domínguez. Sequal: big data tool to perform quality control and data preprocessing of large ngs datasets. *IEEE Access*, 8:146075–146084, 2020.

[62] Y. Fang, S. Wang, H. Tan, X. Zhang, and J. Zhang. Clrc: a new erasure code localization algorithm for hdfs. In *2021 International Conference on Computer Engineering and Artificial Intelligence (ICCEAI)*, pages 62–65. IEEE, 2021.

[63] B. Flury. *A first course in multivariate statistics*. Springer Science & Business Media, 2013.

[64] E. E. Freeman, D. E. McMahon, J. B. Lipoff, M. Rosenbach, C. Kovarik, S. R. Desai, J. Harp, J. Takeshita, L. E. French, H. W. Lim, et al. The spectrum of covid-19–associated dermatologic manifestations: An international registry of 716 patients from 31 countries. *Journal of the American academy of dermatology*, 83(4):1118–1129, 2020.

[65] E. E. Gad, R. Mateescu, F. Blagojevic, C. Guyot, and Z. Bandic. Repair-optimal mds array codes over gf (2). In *2013 IEEE International Symposium on Information Theory*, pages 887–891. IEEE, 2013.

[66] C. Galván Casas, A. Catala, G. Carretero Hernández, P. Rodríguez-Jiménez, D. Fernández-Nieto, A. Rodríguez-Villa Lario, I. Navarro Fernández, R. Ruiz-Villaverde, D. Falkenhain-López, M. Llamas Velasco, et al. Classification of the cutaneous manifestations of covid-19: a rapid prospective nationwide consensus study in spain with 375 cases. *British Journal of Dermatology*, 183(1):71–77, 2020.

[67] Z. Gao, L. Zhang, Y. Cheng, K. Guo, A. Ullah, and P. Reviriego. Design of fpga-implemented reed–solomon erasure code (rs-ec) decoders with fault detection and location on user memory. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, 29(6):1073–1082, 2021.

[68] A. García, A. L. Palacio, J. F. R. Román, J. C. Casamayor, and O. Pastor. Towards the understanding of the human genome: a holistic conceptual modeling approach. *IEEE Access*, 8:197111–197123, 2020.

[69] J. Garcia. Duplications and misattributions of file fragment hashes in image and compressed files. In *2018 9th IFIP International Conference on New Technologies, Mobility and Security (NTMS)*, pages 1–5. IEEE, 2018.

[70] A. Ghani, A. Badshah, S. Jan, A. A. Alshdadi, and A. Daud. Issues and challenges in cloud storage architecture: a survey. *arXiv preprint arXiv:2004.06809*, 2020.

[71] G. A. Gibson. *Performance and reliability in redundant arrays of inexpensive disks*. University of California at Berkeley, 1999.

[72] R. Gnanadesikan. *Methods for statistical data analysis of multivariate observations*. John Wiley & Sons, 2011.

[73] T. M. Godinho, R. Lebre, L. B. Silva, and C. Costa. An efficient architecture to support digital pathology in standard medical imaging repositories. *Journal of biomedical informatics*, 71:190–197, 2017.

[74] D. Gomez-Barquero, J. J. Gimenez, G.-M. Muntean, Y. Xu, and Y. Wu. Ieee transactions on broadcasting special issue on: 5g media production, contribution, and distribution. *IEEE Transactions on Broadcasting*, 68(2):415–421, 2022.

[75] J. E. Grable and A. C. Lyons. An introduction to big data. *Journal of financial service professionals*, 72(5), 2018.

[76] W.-j. Guan, Z.-y. Ni, Y. Hu, W.-h. Liang, C.-q. Ou, J.-x. He, L. Liu, H. Shan, C.-l. Lei, D. S. Hui, et al. Clinical characteristics of coronavirus disease 2019 in china. *New England journal of medicine*, 382(18):1708–1720, 2020.

[77] B. Guler and O. Ozkasap. Compressed incremental checkpointing for efficient replicated key-value stores. In *2017 IEEE Symposium on Computers and Communications (ISCC)*, pages 76–81. IEEE, 2017.

[78] M. Gupta and C. A. Sastry. *Storage area network fundamentals*. Cisco Press, 2002.

[79] G. Harerimana, B. Jang, J. W. Kim, and H. K. Park. Health big data analytics: A technology survey. *Ieee Access*, 6:65661–65678, 2018.

[80] D. Harnik, D. Naor, and I. Segall. Low power mode in cloud storage systems. In *2009 IEEE International Symposium on Parallel & Distributed Processing*, pages 1–8. IEEE, 2009.

[81] W. Hernandez and A. Mendez. Image compression technique based on some principal components periodicity. In *Control and signal processing applications for mobile and aerial robotic systems*, pages 309–327. IGI Global, 2020.

[82] W. Hernandez, A. Mendez, and F. Ballesteros. Image noise cancellation by taking advantage of the principal component analysis technique. In *IECON 2018-44th Annual Conference of the IEEE Industrial Electronics Society*, pages 3176–3181. IEEE, 2018.

[83] W. Hernandez, A. Mendez, F. Ballesteros, V. Gonzalez-Posada, J. L. Jimenez, H. Chinchero, P. Acosta-Vargas, and R. Zalakeviciute. A method to classify digital images by means of statistics of a wavelet decomposition. In *2019 IEEE 28th International Symposium on Industrial Electronics (ISIE)*, pages 1669–1674. IEEE, 2019.

[84] W. Hernandez, A. Mendez, and T. Göksel. Application of principal component analysis to image compression. *Statistics-Growing Data Sets and Growing Demand for Statistics*, 2018.

[85] W. Hernandez, A. Mendez, P. A. Quezada-Sarmiento, L. A. Jumbo-Flores, P. Mercorelli, V. Tyrsa, P. Acosta-Vargas, I. M. Camejo, J. R. M. Cagua, and W. B. C. Cevallos. Image compression based on periodic principal components. In *IECON 2019-45th Annual Conference of the IEEE Industrial Electronics Society*, volume 1, pages 5634–5641. IEEE, 2019.

[86] R. Hernández Palacios et al. Análisis y optimización del interfaz de comunicación en sistemas de ficheros en red. 2016.

[87] G. Hong. A survey of distributed storage systems. *Supercomputing Frontiers and Innovations*, 5(1), 2004.

[88] Y. Hu, H. C. Chen, P. P. Lee, and Y. Tang. Nccloud: applying network coding for the storage repair in a cloud-of-clouds. In *FAST*, volume 21, 2012.

[89] C. Huang, H. Simitci, Y. Xu, A. Ogus, B. Calder, P. Gopalan, J. Li, and S. Yekhanin. Erasure coding in windows azure storage. In *2012 USENIX Annual Technical Conference (USENIX ATC 12)*, pages 15–26, 2012.

[90] Q. Huang, J. Lan, and X. Li. Robotic arm based automatic ultrasound scanning for three-dimensional imaging. *IEEE Transactions on Industrial Informatics*, 15(2):1173–1182, 2018.

[91] S. Huang, Y. Jiang, X. Qin, Y. Gao, Z. Feng, and P. Zhang. Automatic modulation classification of overlapped sources using multi-gene genetic programming with structural risk minimization principle. *Ieee access*, 6:48827–48839, 2018.

[92] IEQFB. Instituto de Química, Física y Biología. https://ieqfb.com/genomas-que-son-y-para-que-sirven/, último acceso: 17/04/2023.

[93] B. A. Ignacio, C. Wu, and J. Li. Warmcache: A comprehensive distributed storage system combining replication, erasure codes and buffer cache. In *Green, Pervasive, and Cloud Computing: 13th International Conference, GPC 2018, Hangzhou, China, May 11-13, 2018, Revised Selected Papers 13*, pages 269–283. Springer, 2019.

[94] Intel. Isa-l erasure code and recovery. https://www.intel.com/content/www/us/en/developer/articles/code-sample/

intel-isa-l-erasure-code-and-recovery.html/, último acceso: 17/04/2023.

[95] J. E. Jackson. *A user's guide to principal components*. John Wiley & Sons, 2005.

[96] J. L. Jia, M. Kamceva, S. A. Rao, and E. Linos. Cutaneous manifestations of covid-19: a preliminary review. *Journal of the American Academy of Dermatology*, 83(2):687, 2020.

[97] H. Jin and K. Hwang. Stripped mirroring raid architecture. *Journal of systems architecture*, 46(6):543–550, 2000.

[98] M. Jin, X. Chen, and S.-J. Lin. Reducing the bandwidth of block propagation in bitcoin network with erasure coding. *IEEE Access*, 7:175606–175613, 2019.

[99] H. Jo, Y. Kim, H. Lee, Y. C. Lee, H. Han, and S. Kang. On the trade-off between performance and storage efficiency of replication-based object storage. In *2019 IEEE international conference on cloud computing technology and science (CloudCom)*, pages 301–304. IEEE, 2019.

[100] K. D. Johnson, C. Harris, J. K. Cain, C. Hummer, H. Goyal, and A. Perisetti. Pulmonary and extra-pulmonary clinical manifestations of covid-19. *Frontiers in medicine*, 7:526, 2020.

[101] R. A. Johnson, D. W. Wichern, et al. Applied multivariate statistical analysis. 2002.

[102] M. Jones. Anatomy of a cloud storage infrastructure. ibm developer works (2010).

[103] J. Ju, J. Wu, J. Fu, Z. Lin, and J. Zhang. A survey on cloud storage. *J. Comput.*, 6(8):1764–1771, 2011.

[104] J. Keller and S. Wendzel. Reversible and plausibly deniable covert channels in one-time passwords based on hash chains. *Applied Sciences*, 11(2):731, 2021.

[105] J.-J. Kim. Erasure-coding-based storage and recovery for distributed exascale storage systems. *Applied Sciences*, 11(8):3298, 2021.

[106] A. Klimach, J. Evans, J. Stevens, and N. Creasey. Rash as a presenting complaint in a child with covid-19. *Pediatric Dermatology*, 37(5):966–967, 2020.

[107] H. Knudsen, J. Li, J. S. Notland, P. H. Haro, and T. B. Ræder. High-performance asynchronous byzantine fault tolerance consensus protocol. In *2021 IEEE International Conference on Blockchain (Blockchain)*, pages 476–483. IEEE, 2021.

[108] K. Kralevska. Applied erasure coding in networks and distributed storage. *arXiv preprint arXiv:1803.01358*, 2018.

[109] K. Kralevska, D. Gligoroski, R. E. Jensen, and H. Øverby. Hashtag erasure codes: From theory to practice. *IEEE Transactions on Big Data*, 4(4):516–529, 2017.

[110] K. Kralevska, D. Gligoroski, and H. Øverby. General sub-packetized access-optimal regenerating codes. *IEEE Communications Letters*, 20(7):1281–1284, 2016.

[111] M. N. Krishnan, N. Prakash, V. Lalitha, B. Sasidharan, P. V. Kumar, S. Narayanamurthy, R. Kumar, and S. Nandi. Evaluation of codes with inherent double replication for hadoop. In *6th USENIX Workshop on Hot Topics in Storage and File Systems (HotStorage 14)*, 2014.

[112] B. Kulkarni and V. Bhosale. Efficient storage utilization using erasure codes in openstack cloud. In *2016 International Conference on*

Inventive Computation Technologies (ICICT), volume 3, pages 1–5. IEEE, 2016.

[113] T. Kurzendorfer, P. Fischer, N. Mirshahzadeh, T. Pohl, A. Brost, S. Steidl, and A. Maier. Rapid interactive and intuitive segmentation of 3d medical images using radial basis function interpolation. *Journal of Imaging*, 3(4):56, 2017.

[114] S. Landset, T. M. Khoshgoftaar, A. N. Richter, and T. Hasanin. A survey of open source tools for machine learning with big data in the hadoop ecosystem. *Journal of Big Data*, 2(1):1–36, 2015.

[115] M. Lasalvia, G. Perna, and V. Capozzi. Dna-related modifications in a mixture of human lympho-monocyte exposed to radiofrequency fields and detected by raman microspectroscopy analysis. *Applied Sciences*, 9(18):3700, 2019.

[116] V. P. Latha, N. S. Reddy, and A. S. Babu. Enhancing performance of multi-cloud storage environment using modified erasure coding technique. *Webology (ISSN: 1735-188X)*, 18(6), 2021.

[117] R. Lebre, L. Bastião, and C. Costa. Shared medical imaging repositories. In *Building Continents of Knowledge in Oceans of Data: The Future of Co-Created eHealth*, pages 411–415. IOS Press, 2018.

[118] R. Lebre, E. Pinho, J. M. Silva, and C. Costa. Dicoogle framework for medical imaging teaching and research. In *2020 IEEE Symposium on Computers and Communications (ISCC)*, pages 1–7. IEEE, 2020.

[119] R. Lebre, L. B. Silva, and C. Costa. A cloud-ready architecture for shared medical imaging repository. *Journal of Digital Imaging*, 33(6):1487–1498, 2020.

[120] R. Lebre, L. B. Silva, and C. Costa. Decentralizing the storage of a dicom compliant pacs. In *2021 IEEE International Conference on*

Bioinformatics and Biomedicine (BIBM), pages 2749–2756. IEEE, 2021.

[121] M. Ledwon, B. F. Cockburn, and J. Han. Design and evaluation of an fpga-based hardware accelerator for deflate data decompression. In *2019 IEEE Canadian Conference of Electrical and Computer Engineering (CCECE)*, pages 1–6. IEEE, 2019.

[122] D.-G. Lee, Y. Jang, and Y.-S. Seo. Intelligent image synthesis for accurate retinal diagnosis. *Electronics*, 9(5):767, 2020.

[123] O. T. Lee, S. M. Kumar, and P. Chandran. Erasure coded storage systems for cloud storage - challenges and opportunities. In *2016 International Conference on Data Science and Engineering (ICDSE)*, pages 1–7. IEEE, 2016.

[124] H. Li, M. Hao, S. Novakovic, V. Gogte, S. Govindan, D. R. Ports, I. Zhang, R. Bianchini, H. S. Gunawi, and A. Badam. Leapio: Efficient and portable virtual nvme storage on arm socs. In *Proceedings of the Twenty-Fifth International Conference on Architectural Support for Programming Languages and Operating Systems*, pages 591–605, 2020.

[125] H. Li, Y. Zhang, Z. Zhang, S. Liu, D. Li, X. Liu, and Y. Peng. Parix: Speculative partial writes in erasure-coded systems. In *USENIX Annual Technical Conference*, pages 581–587, 2017.

[126] J. Li and B. Li. Beehive: erasure codes for fixing multiple failures in distributed storage systems. *IEEE Transactions on Parallel and Distributed Systems*, 28(5):1257–1270, 2016.

[127] J. Li, X. Tang, and C. Tian. A generic transformation for optimal repair bandwidth and rebuilding access in mds codes. In *2017 IEEE International Symposium on Information Theory (ISIT)*, pages 1623–1627. IEEE, 2017.

[128] X. Li, R. Li, P. P. Lee, and Y. Hu. Openec: Toward unified and configurable erasure coding management in distributed storage systems. In *FAST*, pages 331–344, 2019.

[129] Z. Li, M. Lv, Y. Xu, Y. Li, and L. Xu. D3: Deterministic data distribution for efficient data reconstruction in erasure-coded distributed storage systems. In *2019 IEEE International Parallel and Distributed Processing Symposium (IPDPS)*, pages 545–556. IEEE, 2019.

[130] L. Liang, H. He, J. Zhao, C. Liu, Q. Luo, and X. Chu. An erasure-coded storage system for edge computing. *IEEE Access*, 8:96271–96283, 2020.

[131] N. Liang, X. Zhang, H. Yang, X. Dong, and C. Zhang. An optimal recovery approach for liberation codes in distributed storage systems. *IEEE Access*, 8:137631–137645, 2020.

[132] W. Liang, Y. Fan, K.-C. Li, D. Zhang, and J.-L. Gaudiot. Secure data storage and recovery in industrial blockchain network environments. *IEEE Transactions on Industrial Informatics*, 16(10):6543–6552, 2020.

[133] W. Lin, D. M. Chiu, and Y. Lee. Erasure code replication revisited. In *Proceedings. Fourth International Conference on Peer-to-Peer Computing, 2004. Proceedings.*, pages 90–97. IEEE, 2004.

[134] X. Lin, W. Zhuo, H. Liu, and T. Xie. Potential of fluid dynamic bowtie filter for dose reduction and image quality improvement of cone-beam ct. *Applied Sciences*, 12(18):9346, 2022.

[135] A. Liu and T. Yu. Overview of cloud storage and architecture. *International Journal of Scientific & Technology Research*, 2018.

[136] K. Liu, J. Peng, J. Wang, Z. Huang, and J. Pan. Adaptive and scalable caching with erasure codes in distributed cloud-edge storage systems. *IEEE Transactions on Cloud Computing*, 2022.

[137] P. Liu, Z. Pan, and J. Lei. Parameter identification of reed-solomon codes based on probability statistics and galois field fourier transform. *IEEE Access*, 7:33619–33630, 2019.

[138] S. Liu, H. Wu, Y. Huang, Y. Yang, and J. Jia. Accelerated structure-aware sparse bayesian learning for three-dimensional electrical impedance tomography. *Ieee transactions on industrial informatics*, 15(9):5033–5041, 2019.

[139] T. Liu. *Efficient Data Reduction in HPC and Distributed Storage Systems*. Temple University, 2021.

[140] W. Liu, M. Guo, P. Liu, and Y. Du. Mfdcmodel: A novel classification model for classification of benign and malignant breast tumors in ultrasound images. *Electronics*, 11(16):2583, 2022.

[141] F. P.-W. Lo, Y. Sun, J. Qiu, and B. P. Lo. Point2volume: A vision-based dietary assessment approach using view synthesis. *IEEE Transactions on Industrial Informatics*, 16(1):577–586, 2019.

[142] M. Luby. Capacity bounds for distributed storage. *arXiv preprint arXiv:1610.03541*, 2016.

[143] M. Luby, R. Padovani, T. J. Richardson, L. Minder, and P. Aggarwal. Liquid cloud storage. *ACM Transactions on Storage (TOS)*, 15(1):1–49, 2019.

[144] J. Ma. Algorithms for non-negatively constrained maximum penalized likelihood reconstruction in tomographic imaging. *Algorithms*, 6(1):136–160, 2013.

[145] L. M. Madigan, R. G. Micheletti, and K. Shinkai. How dermatologists can learn and contribute at the leading edge of the covid-19 global pandemic. *JAMA dermatology*, 156(7):733–734, 2020.

[146] C. Manopoulos, A. Raptis, W. Krishan, C. Mavratzas, M. Drandakis, S. Astraka, I. Kouerinis, and N. Vaxevanidis. Reconstruction, processing and smoothing of surface geometry of a patient specific ascending aortic aneurysm. In *IOP Conference Series: Materials Science and Engineering*, volume 1037, page 012020. IOP Publishing, 2021.

[147] F. Marcillo, A. F. Diaz, R. H. Palacios, and W. Hernandez. Evaluating erasure codes in dicoogle pacs. *IEEE Access*, 10:71874–71885, 2022.

[148] D. C. Marinescu. *Cloud computing: theory and practice*. Morgan Kaufmann, 2022.

[149] S. Mazumdar, D. Seybold, K. Kritikos, and Y. Verginadis. A survey on data storage and placement methodologies for cloud-big data ecosystem. *Journal of Big Data*, 6(1):1–37, 2019.

[150] F.-M. Melgarejo-Meseguer, F.-J. Gimeno-Blanes, M.-E. Salar-Alcaraz, J.-R. Gimeno-Blanes, J. Martínez-Sánchez, A. García-Alberola, and J.-L. Rojo-Álvarez. Electrocardiographic fragmented activity (i): physiological meaning of multivariate signal decompositions. *Applied Sciences*, 9(17):3566, 2019.

[151] MINIO. Minio erasure code quickstart guide. https://docs.min.io/docs/minio-erasure-code-quickstart-guide.html/, último acceso: 17/04/2023.

[152] D. G. Mitchell, M. Lentmaier, A. E. Pusane, and D. J. Costello. Randomly punctured ldpc codes. *IEEE Journal on Selected Areas in Communications*, 34(2):408–421, 2015.

[153] L. J. Mohan, P. I. S. Caneleo, U. Parampalli, and A. Harwood. Geo-aware erasure coding for high-performance erasure-coded storage clusters. *Annals of Telecommunications*, 73(1):139–152, 2018.

[154] S. Mu, K. Chen, P. Gao, F. Ye, Y. Wu, and W. Zheng. μlibcloud: Providing high available and uniform accessing to multiple cloud storages. In *2012 ACM/IEEE 13th International Conference on Grid Computing*, pages 201–208. IEEE, 2012.

[155] K. Nansai, X. Chen, S. Chen, and J. Zhang. Hdfs erasure coding in production. *Accessed: Jun*, 10, 2019.

[156] S. Narayanamurthy. Modern erasure codes for distributed storage systems. In *Storage Developer Conference (SDC), SNIA, Santa Clara*, 2016.

[157] M. U. Nasir, M. Gollapalli, M. Zubair, M. A. Saleem, S. Mehmood, M. A. Khan, A. Mosavi, et al. Advance genome disorder prediction model empowered with deep learning. *IEEE Access*, 10:70317–70328, 2022.

[158] F. Oggier and A. Datta. Self-repairing homomorphic codes for distributed storage systems. In *2011 Proceedings IEEE INFOCOM*, pages 1215–1223. IEEE, 2011.

[159] F. Oggier and A. Datta. On grid quorums for erasure coded data. *Entropy*, 23(2):177, 2021.

[160] A. Olabi, C. Onumaegbu, T. Wilberforce, M. Ramadan, M. A. Abdelkareem, and A. H. Al-Alami. Critical review of energy storage systems. *Energy*, 214:118987, 2021.

[161] D. E. I. ONEFS. A technical overview, 2017.

[162] R. Palacios, A. F. Fernández-Portillo, E. F. Sánchez-Úbeda, and P. García-De-Zúñiga. Htb: A very effective method to protect web servers against breach attack to https. *IEEE Access*, 10:40381–40390, 2022.

[163] L. Pamies-Juarez, F. Blagojevic, R. Mateescu, C. Gyuot, E. E. Gad, and Z. Bandic. Opening the chrysalis: On the real repair performance of {MSR} codes. In *14th USENIX conference on file and storage technologies (FAST 16)*, pages 81–94, 2016.

[164] G. H. Park, Y. Song, and H. J. Choi. Compression algorithms for imaging instruments–a mini review. *Journal of Multidisciplinary Engineering Science and Technology (JMEST),[Internet]*, 9(1):14923–9, 2022.

[165] J. Pati. Gene expression analysis for early lung cancer prediction using machine learning techniques: An eco-genomics approach. *IEEE Access*, 7:4232–4238, 2018.

[166] D. A. Patterson, G. Gibson, and R. H. Katz. A case for redundant arrays of inexpensive disks (raid). In *Proceedings of the 1988 ACM SIGMOD international conference on Management of data*, pages 109–116, 1988.

[167] D. Peck. Digital imaging and communications in medicine (dicom): a practical introduction and survival guide, 2009.

[168] E. Pinho, T. Godinho, F. Valente, and C. Costa. A multimodal search engine for medical imaging studies. *Journal of digital imaging*, 30(1):39–48, 2017.

[169] J. S. Plank. T1: erasure codes for storage applications. In *Proc. of the 4th USENIX Conference on File and Storage Technologies*, pages 1–74, 2005.

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

[171] J. S. Plank. Erasure codes for storage systems: A brief primer. *; login:: the magazine of USENIX & SAGE*, 38(6):44–50, 2013.

[172] A. B. Popescu, I. A. Taca, A. Vizitiu, C. I. Nita, C. Suciu, L. M. Itu, and A. Scafa-Udriste. Obfuscation algorithm for privacy-preserving deep learning-based medical image analysis. *Applied Sciences*, 12(8):3997, 2022.

[173] N. E. Protonotarios, A. S. Fokas, A. Vrachliotis, V. Marinakis, N. Dikaios, and G. A. Kastis. Reconstruction of preclinical pet images via chebyshev polynomial approximation of the sinogram. *Applied Sciences*, 12(7):3335, 2022.

[174] W. Pu, N. Chen, and Q. Zhong. Sdcup: Software-defined-control based erasure-coded collaborative data update mechanism. *IEEE Access*, 8:180646–180660, 2020.

[175] A. Rabinowitz and D. Rawitz. Overflow management with self-eliminations. In *International Symposium on Algorithms and Experiments for Sensor Systems, Wireless Networks and Distributed Robotics*, pages 124–139. Springer, 2021.

[176] K. Rashmi, P. Nakkiran, J. Wang, N. B. Shah, and K. Ramchandran. Having your cake and eating it too: Jointly optimal erasure codes for {I/O}, storage, and network-bandwidth. In *13th USENIX Conference on File and Storage Technologies (FAST 15)*, pages 81–94, 2015.

[177] K. Rashmi, N. B. Shah, and K. Ramchandran. A piggybacking design framework for read-and download-efficient distributed storage codes. *IEEE Transactions on Information Theory*, 63(9):5802–5820, 2017.

[178] K. V. Rashmi, N. B. Shah, D. Gu, H. Kuang, D. Borthakur, and K. Ramchandran. A"hitchhiker's"guide to fast and efficient data reconstruction in erasure-coded data centers. In *Proceedings of the 2014 ACM conference on SIGCOMM*, pages 331–342, 2014.

[179] A. S. Rawat, D. S. Papailiopoulos, A. G. Dimakis, and S. Vishwanath. Locality and availability in distributed storage. *IEEE Transactions on Information Theory*, 62(8):4481–4493, 2016.

[180] S. Recalcati et al. Cutaneous manifestations in covid-19: a first perspective. *J Eur Acad Dermatol Venereol*, 34(5), 2020.

[181] I. S. Reed and G. Solomon. Polynomial codes over certain finite fields. *Journal of the society for industrial and applied mathematics*, 8(2):300–304, 1960.

[182] L. Rizzo. Effective erasure codes for reliable computer communication protocols. *ACM SIGCOMM computer communication review*, 27(2):24–36, 1997.

[183] R. Roe. Storage advances for hpc and ai: Robert roe looks at storage technologies being developed to suit both ai and hpc workloads. *Scientific Computing World*, (165):8–10, 2019.

[184] E. W. Rozier, W. Belluomini, V. Deenadhayalan, J. Hafner, K. Rao, and P. Zhou. Evaluating the impact of undetected disk errors in raid systems. In *2009 IEEE/IFIP International Conference on Dependable Systems & Networks*, pages 83–92. IEEE, 2009.

[185] W. Ryan and S. Lin. Channel codes: Classical and modern, cambridge univ, 2009.

[186] A. Saleh. *Daten-und Dimensionsreduktionstechniken für Big Data Analytics*. PhD thesis, Universität Rostock, 2016.

[187] M. Santamaria, F. Cricri, J. Lainema, R. G. Youvalari, H. Zhang, and M. M. Hannuksela. Content-adaptive neural network post-processing filter with nnr-coded weight-updates. In *2022 IEEE International Conference on Image Processing (ICIP)*, pages 2251–2255. IEEE, 2022.

[188] M. Santos, S. de Francesco, L. A. B. Silva, A. Silva, C. Costa, and N. Rocha. Multi vendor dicom metadata access a multi site hospital approach using dicoogle. In *2013 8th Iberian Conference on Information Systems and Technologies (CISTI)*, pages 1–7. IEEE, 2013.

[189] M. Santos, J. Pavão, T. Godinho, and N. P. Rocha. Dicom metadata aggregation from multiple healhtcare facilities. In *2017 IEEE 5th Portuguese Meeting on Bioengineering (ENBENG)*, pages 1–4. IEEE, 2017.

[190] G. Sarasa, A. Granados, and F. B. Rodriguez. An approach of algorithmic clustering based on string compression to identify bird songs species in xeno-canto database. In *2017 3rd International Conference on Frontiers of Signal Processing (ICFSP)*, pages 101–104. IEEE, 2017.

[191] A. Sarinova, R. Lisnevskyi, A. Biloshchytskyi, and A. Akizhanova. The lossless compression algorithm of hyperspectral aerospace images with correlation and bands grouping. In *2022 International Conference on Smart Information Systems and Technologies (SIST)*, pages 1–5. IEEE, 2022.

[192] B. Sasidharan, M. Vajha, and P. V. Kumar. An explicit, coupled-layer construction of a high-rate msr code with low sub-packetization level, small field size and all-node repair. *arXiv preprint arXiv:1607.07335*, 2016.

[193] M. Sathiamoorthy, M. Asteris, D. Papailiopoulos, A. G. Dimakis, R. Vadali, S. Chen, and D. Borthakur. Xoring elephants: Novel erasure codes for big data. *arXiv preprint arXiv:1301.3791*, 2013.

[194] F. Schmuck and R. Haskin. {GPFS}: A {Shared-Disk} file system for large computing clusters. In *Conference on File and Storage Technologies (FAST 02)*, 2002.

[195] V. Sciortino, S. Pasta, T. Ingrassia, and D. Cerniglia. A population-based 3d atlas of the pathological lumbar spine segment. *Bioengineering*, 9(8):408, 2022.

[196] R. Settlage, B. Chalker, E. G. Franz, E. Moore, and D. Hudak. Hpc for everyone: Open ondemand isc19 workshop on interactive high-performance computing. *ISC 19*, 2019.

[197] Z. A. Shaikh, A. A. Khan, L. Baitenova, G. Zambinova, N. Yegina, N. Ivolgina, A. A. Laghari, and S. E. Barykin. Blockchain hyperledger with non-linear machine learning: A novel and secure educational accreditation registration and distributed ledger preservation architecture. *Applied Sciences*, 12(5):2534, 2022.

[198] A. Shakarami, M. Ghobaei-Arani, A. Shahidinejad, M. Masdari, and H. Shakarami. Data replication schemes in cloud computing: a survey. *Cluster Computing*, 24:2545–2579, 2021.

[199] P. Shantharama, A. S. Thyagaturu, A. Yatavelli, P. Lalwaney, M. Reisslein, G. Tkachuk, and E. J. Pullin. Hardware acceleration for container migration on resource-constrained platforms. *IEEE Access*, 8:175070–175085, 2020.

[200] J. Shen, Y. Li, G. Sheng, Y. Zhou, and X. Wang. Efficient memory caching for erasure coding based key-value storage systems. In *CCF Conference on Big Data*, pages 512–539. Springer, 2018.

[201] R. M. Sherman and S. L. Salzberg. Pan-genomics in the human genome era. *Nature Reviews Genetics*, 21(4):243–254, 2020.

[202] H. Shi and X. Lu. Designing high-performance erasure coding schemes for next-generation storage systems. In *Colorado Convention Center*, 2019.

[203] H. Shi, X. Lu, D. Shankar, and D. K. Panda. High-performance multi-rail erasure coding library over modern data center architectures: early experiences. In *Proceedings of the ACM Symposium on Cloud Computing*, pages 530–531, 2018.

[204] D.-J. Shin and J.-J. Kim. Deep reinforcement learning-based network routing technology for data recovery in exa-scale cloud distributed clustering systems. *Applied Sciences*, 11(18):8727, 2021.

[205] S. Shirinbab, L. Lundberg, and D. Erman. Performance evaluation of distributed storage systems for cloud computing. *Int. J. Comput. Their Appl.*, 20(4):195–207, 2013.

[206] M. Sikandar, R. Sohail, Y. Saeed, A. Zeb, M. Zareei, M. A. Khan, A. Khan, A. Aldosary, and E. M. Mohamed. Analysis for disease gene association using machine learning. *IEEE Access*, 8:160616–160626, 2020.

[207] L. A. B. Silva, C. Costa, and J. L. Oliveira. A pacs archive architecture supported on cloud services. *International journal of computer assisted radiology and surgery*, 7(3):349–358, 2012.

[208] B. Snaith. expanding the diagnostic imaging workforce: how assistants are supporting uk service delivery. *MULTIPARAMETRIC MRI MAPS OF THE LIVER*, page 64, 2019.

[209] Y. Song, T. Mu, and B. Wang. Hv-snsp: A low-overhead data recovery method based on cross-checking. *IEEE Access*, 2023.

[210] M. Su, L. Zhang, Y. Wu, K. Chen, and K. Li. Systematic data placement optimization in multi-cloud storage for complex requirements. *IEEE Transactions on Computers*, 65(6):1964–1977, 2015.

[211] A. Tammaro, G. Adebanjo, F. Parisella, A. Pezzuto, and J. Rello. Cutaneous manifestations in covid-19: the experiences of barcelona and rome. *Journal of the European Academy of Dermatology and Venereology: JEADV*, 34(7):e306–e307, 2020.

[212] BMD Software Open Source Projects Strong Collaboration With Universities. Rd Institutes and Other Companies to Improve the Processes and Change Healthcare. https://www.bmd-software.com/research/, último acceso: 17/04/2023.

[213] Universidade de Aveiro Portugal. UA. PT. Bioinformatics. A Computational Biology and Biomedical Informatics Research Group. https://bioinformatics.ua.pt/, último acceso: 17/04/2023.

[214] A. Thusoo, Z. Shao, S. Anthony, D. Borthakur, N. Jain, J. Sen Sarma, R. Murthy, and H. Liu. Data warehousing and analytics infrastructure at facebook. In *Proceedings of the 2010 ACM SIGMOD International Conference on Management of data*, pages 1013–1020, 2010.

[215] M. P. Toader, D. C. Branisteanu, M. Glod, I. M. Esanu, C. I. Branisteanu, M.-S. Capsa, A. Dimitriu, A. C. Nicolescu, A. C. Pinzariu, and D. E. Branisteanu. Mucocutaneous lesions associated with sars-cov-2 infection. *Experimental and Therapeutic Medicine*, 23(4):1–8, 2022.

[216] A. Trigeorgi, N. Nicolaou, C. Georgiou, T. Hadjistasi, E. Stavrakis, V. Cadambe, and B. Urgaonkar. Towards practical atomic distributed shared memory: An experimental evaluation. In *Stabilization,*

Safety, and Security of Distributed Systems: 24th International Symposium, SSS 2022, Clermont-Ferrand, France, November 15–17, 2022, Proceedings, pages 35–50. Springer, 2022.

[217] Y. Uezato. Accelerating xor-based erasure coding using program optimization techniques. In *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*, pages 1–14, 2021.

[218] M. Vajha, V. Ramkumar, B. Puranik, G. Kini, E. Lobo, B. Sasidharan, P. V. Kumar, A. Barg, M. Ye, S. Narayanamurthy, et al. Clay codes: Moulding {MDS} codes to yield an {MSR} code. In *16th USENIX Conference on File and Storage Technologies (FAST 18)*, pages 139–154, 2018.

[219] F. Valente, C. Costa, and A. Silva. Dicoogle, a pacs featuring profiled content based image retrieval. *PloS one*, 8(5):e61888, 2013.

[220] F. Valente, L. A. B. Silva, T. M. Godinho, and C. Costa. Anatomy of an extensible open source pacs. *Journal of digital imaging*, 29(3):284–296, 2016.

[221] J. van der Krogt. Comparing the reed solomon code to two recently found codes for distributed storage. 2019.

[222] R. Vestergaard, D. E. Lucani, and Q. Zhang. A randomly accessible lossless compression scheme for time-series data. In *IEEE INFOCOM 2020-IEEE Conference on Computer Communications*, pages 2145–2154. IEEE, 2020.

[223] C. Viana-Ferreira, C. Costa, and J. L. Oliveira. Dicoogle relay-a cloud communications bridge for medical imaging. In *2012 25th IEEE International Symposium on Computer-Based Medical Systems (CBMS)*, pages 1–6. IEEE, 2012.

[224] P. Virtue and M. Lustig. The empirical effect of gaussian noise in undersampled mri reconstruction. *Tomography*, 3(4):211–221, 2017.

[225] Y. Wang, Y. Yang, C. Han, L. Ye, Y. Ke, and Q. Wang. Lr-lru: A pacs-oriented intelligent cache replacement policy. *IEEE Access*, 7:58073–58084, 2019.

[226] H. Weatherspoon and J. D. Kubiatowicz. Erasure coding vs. replication: A quantitative comparison. In *International Workshop on Peer-to-Peer Systems*, pages 328–337. Springer, 2002.

[227] B. Wei, J. Wu, X. Su, Q. Huang, and Y. Liu. Adaptive updates for erasure-coded storage systems based on data delta and logging. In *Parallel and Distributed Computing, Applications and Technologies: 22nd International Conference, PDCAT 2021, Guangzhou, China, December 17–19, 2021, Proceedings*, pages 187–197. Springer, 2022.

[228] M. Xia, M. Saxena, M. Blaum, and D. A. Pease. A tale of two erasure codes in {HDFS}. In *13th USENIX conference on file and storage technologies (FAST 15)*, pages 213–226, 2015.

[229] C. Xiao, Y. Xia, Q. Zhang, X. Gong, and L. Zhu. Cbase-ec: Achieving optimal throughput-storage efficiency trade-off using erasure codes. *Electronics*, 10(2):126, 2021.

[230] X. Xiao, B. Vasić, S. Lin, J. Li, and K. Abdel-Ghaffar. Quasi-cyclic ldpc codes with parity-check matrices of column weight two or more for correcting phased bursts of erasures. *IEEE Transactions on Communications*, 69(5):2812–2823, 2021.

[231] Y. Xiao, S. Zhou, and L. Zhong. Erasure coding-oriented data update for cloud storage: a survey. *IEEE Access*, 8:227982–227998, 2020.

[232] E. Xie, P. Ni, R. Zhang, and X. Li. Limited-angle ct reconstruction with generative adversarial network sinogram inpainting and unsupervised artifact removal. *Applied Sciences*, 12(12):6268, 2022.

[233] Y. Xie and Q. Li. A review of deep learning methods for compressed sensing image reconstruction and its medical applications. *Electronics*, 11(4):586, 2022.

[234] B. Xu, J. Huang, Q. Cao, X. Qin, and P. Xie. F-write: Fast rdma-supported writes in erasure-coded in-memory clusters. In *2021 IEEE International Parallel and Distributed Processing Symposium (IPDPS)*, pages 817–826. IEEE, 2021.

[235] J. Xu, P. Wu, Y. Chen, Q. Meng, H. Dawood, and M. M. Khan. A novel deep flexible neural forest model for classification of cancer subtypes based on gene expression data. *IEEE Access*, 7:22086–22095, 2019.

[236] L. Xu, M. Lv, Z. Li, C. Li, and Y. Xu. Pdl: A data layout towards fast failure recovery for erasure-coded distributed storage systems. In *IEEE INFOCOM 2020-IEEE Conference on Computer Communications*, pages 736–745. IEEE, 2020.

[237] L. Xu, M. Lyu, Q. Li, L. Xie, C. Li, and Y. Xu. Selectiveec: Towards balanced recovery load on erasure-coded storage systems. *IEEE Transactions on Parallel and Distributed Systems*, 33(10):2386–2400, 2021.

[238] L. Xu, M. Lyu, Z. Li, C. Li, and Y. Xu. A data layout and fast failure recovery scheme for distributed storage systems with mixed erasure codes. *IEEE Transactions on Computers*, 71(8):1740–1754, 2021.

[239] L. Xu, M. Lyu, Z. Li, Y. Li, and Y. Xu. Deterministic data distribution for efficient recovery in erasure-coded storage systems.

IEEE Transactions on Parallel and Distributed Systems, 31(10):2248–2262, 2020.

[240] L. Yan. Dicom standard and its application in pacs system. *Medical Imaging Process & Technology*, 1(1), 2018.

[241] L. Yao, J. Lu, J. Liu, D. Wang, and B. Meng. A secure and efficient distributed storage scheme saont-rs based on an improved aont and erasure coding. *IEEE Access*, 6:55126–55138, 2018.

[242] B. Ye, X. Yuan, Z. Cai, and T. Lan. Severity assessment of covid-19 based on feature extraction and v-descriptors. *IEEE Transactions on Industrial Informatics*, 17(11):7456–7467, 2021.

[243] M. Ye and A. Barg. Explicit constructions of optimal-access mds codes with nearly optimal sub-packetization. *IEEE Transactions on Information Theory*, 63(10):6307–6317, 2017.

[244] P. N. Yianilos and S. Sobti. The evolving field of distributed storage. *IEEE Internet Computing*, 5(5):35–39, 2001.

[245] A. Zakari, S. P. Lee, and I. A. T. Hashem. A community-based fault isolation approach for effective simultaneous localization of faults. *IEEE Access*, 7:50012–50030, 2019.

[246] zfec. An efficient, portable erasure coding tool. https://pypi.org/project/zfec/, último acceso: 17/04/2023.

[247] X. Zhang and Y. Hu. Efficient storage scaling for mbr and msr codes. *IEEE Access*, 8:78992–79002, 2020.

[248] Y. Zhang, H. Li, S. Liu, J. Xu, and G. Xue. Pbs: An efficient erasure-coded block storage system based on speculative partial writes. *ACM Transactions on Storage (ToS)*, 16(1):1–25, 2020.