ARMAZENAMENTO ESCALÁVEL DE DADOS COM OBJECT STORAGE E TOLERÂNCIA A FALHAS: UM ESTUDO DE CASO EM AMBIENTE ON-PREMISE

Allesson Carneiro de Queiroz
allessonqueiroz@gmail.com
Josino Rodrigues
josino.neto@jaboatao.ifpe.edu.br.com

RESUMO

Este artigo explora o armazenamento escalável de dados utilizando Object Storage, com foco em tolerância a falhas. A análise teórica abrange os paradigmas de armazenamento e destaca a importância do Object Storage para escalabilidade eficiente. Um estudo de caso local exemplifica a implementação prática, evidenciando a resiliência do sistema frente a falhas. Os resultados destacam a viabilidade e benefícios do Object Storage em ambientes que exigem confiabilidade e flexibilidade, contribuindo para a compreensão prática e teórica do armazenamento de dados.

Palavras-chave: Armazenamento de objeto; Escalabilidade; Tolerância a falhas;

ABSTRACT

This article explores scalable data storage using Object Storage, with a specific focus on fault tolerance. The theoretical analysis covers storage paradigms, emphasizing the significance of Object Storage for efficient scalability. A local case study illustrates practical implementation, showcasing the system's resilience in the case of faults. Results highlight the viability and benefits of Object Storage in environments requiring reliability and flexibility, contributing to both practical and theoretical understanding of data storage.

Keywords: Object Storage; Scalability; fault tolerance;

1 Introdução

Nosso mundo vem sendo cada vez mais conectado e enriquecido com mais e mais tecnologias e informações, mais dados são gerados a cada segundo, sendo necessário guardar grande parte dessas informações. De acordo com a revista Exame a sociedade tem gerado, nos últimos anos, mais dados do que nunca. Só em 2019 foram gerados 40 trilhões de gigabytes e precisamos organizar cada vez mais esse acervo de informações. (“Temos mais dados do que nunca. Como usá-los a nosso favor?”, 2021)

Instituto Federal de Pernambuco. Campus Jaboatão dos Guararapes Curso de analise e desenvolvimento de sistemas 15 de janeiro de 2025.

O estudo realizado na universidade de Illinois (AMARAVADI, 2022) concluiu que o acesso à informação torna-se um problema com o aumento do volume de dados no mundo. É necessário estarmos habilitados para tratar dados estruturados e não estruturados, como áudio e vídeo. Além disso, será de extrema importância a habilidade de identificar e interpretar o formato dos dados onde muitos deles são proprietários e frequentemente não documentados. Em particular, não relacionado à natureza dos dados está a necessidade de armazená-los, recuperá-los e se proteger contra a perda desses dados.

O armazenamento de dados estruturados já vem sendo endereçado pelos bancos de dados relacionais nas últimas décadas. Atualmente é possível encontrar em sistemas de dados relacionais funcionalidades nativas que tem o objetivo de resolver problemas como replicação, tolerância a falha e alta disponibilidade. Já no cenário de armazenamento de dados não estruturados várias pesquisas vêm sendo desenvolvidas nos últimos anos com o objetivo de criar formas eficientes e econômicas para armazenar dados não estruturados (LITCHFIELD; ALTHOUSE, 2014).

2 Trabalhos Relacionados

A pesquisa em armazenamento escalável e tolerante a falhas tem sido amplamente explorada na academia e na indústria, com enfoques variados dependendo do contexto tecnológico e das necessidades específicas. Esta seção consolida estudos relevantes, organizando-os em categorias temáticas para contextualizar as contribuições deste trabalho

2.1 Software-Defined Storage (SDS) e Flexibilidade

Macedo et al. (2021) Propuseram uma classificação para sistemas SDS, classificando-os por desempenho, segurança e escalabilidade. Os autores identificaram que soluções como o object storage permitem flexibilidade na alocação de recursos, especialmente em ambientes heterogêneos. Da et al. (2019) exploraram a aplicação de SDS em ambientes hiper convergentes com containers, demonstrando que a abstração do hardware reduz custos de infraestrutura física em até 30%, mantendo alta disponibilidade.

2.2 Comparativos entre Tipos de Armazenamento

Yang, Xiong e Ren (2020) compararam File Storage, Block Storage e Object Storage, concluindo que este último é ideal para dados estáticos e não estruturados devido à sua granularidade e custo-benefício, enquanto o Block Storage é mais eficiente para transações de alta velocidade. Esses resultados foram corroborados pelo artigo "File storage, block storage, or object storage?" (2019), que analisou casos de uso corporativos e destacou que a ausência de hierarquia no Object Storage simplifica o gerenciamento de grandes volumes, reduzindo a complexidade operacional.

2.3 Object Storage e Implementações

Kulkarni et al. (2012) investigaram a arquitetura de Object Storage como solução para dados não estruturados, destacando sua capacidade de fragmentar dados em

objetos identificáveis por metadados. Os autores concluíram que essa abordagem facilita a distribuição em ambientes heterogêneos, reduzindo custos operacionais em larga escala. Complementando essa visão, Zheng et al. (2012) desenvolveram o COSBench, uma ferramenta para avaliar desempenho em sistemas de Object Storage, e identificaram variações significativas na latência de acesso dependendo da localização dos nós, o que reforçou a necessidade de replicação inteligente. Já Gracia-Tinedo et al. (2016) focaram em limitações do Object Storage, como a impossibilidade de modificação incremental de objetos, e propuseram o versionamento como estratégia para garantir consistência, mesmo em atualizações completas.

2.4 Tolerância a Falhas e Escalabilidade

Tanenbaum et al. (2008) discutiram técnicas como redundância e checkpointing em sistemas distribuídos, defendendo que a replicação geográfica é crítica para evitar perda de dados em cenários de desastres. Também foram analisadas estratégias de escalabilidade horizontal em Object Storage, como "server pools", e concluiu-se que a adição dinâmica de nós, sem rebalanceamento, é viável para ambientes que exigem crescimento contínuo, uma abordagem adotada pelo object storage (Ju et al, 2011).

2.5 Estudos de Caso Práticos

Drago, Borges Vieira e Couto da Silva (2013) caracterizaram arquivos armazenados no Dropbox, identificando que 85% deles são binários e estáticos—dados que validam a eficácia do Object Storage para esse perfil. Graebin, et al. (2017) compararam sistemas de arquivos distribuídos, como Hadoop HDFS e Ceph, e recomendaram replicação em múltiplas zonas geográficas para garantir disponibilidade, estratégia implementada pelo MinIO via erasure coding.

Considerando as recentes evoluções no armazenamento de dados, tanto na academia quanto na indústria, esse trabalho nasceu da necessidade do autor de compreender e implementar uma solução para armazenamento escalável de dados não estruturados, que seja tolerante a falhas e eficiente do ponto de vista de custo de armazenamento. Esse trabalho está estruturado da seguinte forma, na seção 3 será apresentado o objetivo geral e metodologias, na seção 4 será apresentado uma breve introdução sobre o cenário atual de sistemas de armazenamento para dados não estruturados, na seção 5 realizaremos um breve comparativo de soluções existentes no mercado e na seção 6 apresentamos nosso estudo de caso, e na seção 7 as considerações finais.

3 Objetivos

3.1 Objetivo geral

O presente estudo tem como objetivo compreender como armazenar dados de forma escalável, tolerante a falhas e eficiente do ponto de vista de custo. Através do desenvolvimento de um estudo de caso.

3.2 Objetivo Específico:

Este artigo visa três contribuições principais para o campo de armazenamento de dados. Primeiramente, apresenta o estado da arte em sistemas de armazenamento escaláveis, com ênfase em Object Storage, revisando criticamente as tecnologias emergentes, desafios técnicos e soluções propostas na literatura recente. Em seguida, constrói um guia prático para auxiliar na tomada de decisões arquiteturais, falando sobre critérios como custo, escalabilidade, tolerância a falhas e adequação a dados não estruturados, de modo a orientar na seleção de soluções alinhadas a seus contextos operacionais. Por fim, apresenta um estudo de caso em ambiente on-premise, utilizando o MinIO, para validar empiricamente a viabilidade de implementação escaláveis e resilientes.

4 Metodologia

Etapa 1 - Como ponto de partida desse trabalho foi realizado a leitura de vários artigos sobre o tema e estruturação do caso de uso escolhido para a investigação.

Etapa 2 - Foi realizado levantamento bibliográfico através da leitura de artigos e trabalhos científicos sobre armazenamento de dados não estruturados. Além disso, foi realizado um levantamento do estado da prática através da investigação de soluções já propostas pela indústria.

Etapa 3 – Foi realizada a estruturação de uma fundamentação teórica sobre o tema através da definição dos principais conceitos, dos principais problemas relacionados ao armazenamento de dados e propostas de soluções existentes.

Etapa 4 – Com base nas descobertas da etapa anterior, um estudo de caso foi desenvolvido para validar a implementação de uma solução para o problema investigado

Etapa 5 – Nesta etapa foram realizadas as principais conclusões e considerações finais foram sintetizadas.

5 Fundamentação teórica

Com a finalidade de executar sua bem sua função, é preciso que um data center funcione numa arquitetura versátil que o permita escalar com facilidade e com economia. Com o objetivo de melhorar o fluxo de dados, o armazenamento definido por software surgiu com o intuito de facilitar e agilizar esse processo (“What is software-defined storage?”, 2018).

5.1 Armazenamento definido por software (SDS - Software Defined Storage)

O conceito de algo definido por software (SDx - Software Defined Anything) engloba tecnologías que abstraem funcionalidades do hardware por meio de software, como redes (SDN – Software Defined Network), comunicações (SDR – Software Defined Radio) e armazenamento (SDS – Software Defined Storage). No caso do SDS, uma camada de software gerencia recursos físicos (como discos e

servidores), independentemente do hardware subjacente, seguindo políticas pré-definidas. Essa abordagem é amplamente adotada em ambientes virtualizados, centralizando o controle do armazenamento sob uma perspectiva flexível e programável (DA et al., 2019).

O SDS opera por meio de interfaces de gerenciamento que automatizam configurações (como alocação de recursos e provisionamento dinâmico) e integram-se a soluções como nuvens (públicas/privadas) e infraestruturas hiper convergentes (HCI – *Hyper-Converged Infrastructure*). Algumas vantagens disso são a flexibilidade de utilização do hardware de armazenamento, a sua alocação eficiente de recursos feita de forma automática, a ativação de funções como desduplicação, replicação e o provisionamento dinâmico utilizando um padrão da indústria, embora esse recurso não seja exclusivo do SDS (CHANDRAMOULI; PINHAS, 2020). Com a utilização desse conceito, que separa o controle do armazenamento do hardware subjacente, é possível obter uma série de vantagens do ponto de vista de flexibilidade, eficiência operacional e escalabilidade.

5.2 Armazenamentos e seus tipos

Enquanto buscamos entender a estrutura do SDS, é importante explorar e compreender os diferentes padrões de armazenamento de dados, cada qual com seu propósito específico. Dentre eles destacam-se o file storage (Armazenamento de arquivos), block storage (armazenamento em blocos) e object storage (armazenamento de objetos). Cada tipo de armazenamento tem seus padrões, políticas e custos que influenciam diretamente em como os dados são acessados, manipulados e escalados.

5.2.1 File Storage

O armazenamento de arquivos, também conhecido como armazenamento em nível de arquivo, utiliza uma metodologia hierárquica: os dados são organizados em pastas e subpastas, armazenados em dispositivos locais (como discos rígidos) ou remotos (como o Network-Attached Storage - NAS). Para localizar um arquivo, o sistema segue o caminho desde o diretório principal até o destino final. Essa estrutura é versátil, suportando desde mídias (fotos, vídeos) até dados de aplicações (bancos de dados, arquivos de configuração).

Sua implementação pode ser local ou em rede, com acesso via protocolos específicos (ARPACI-DUSSEAU, 2018). A facilidade de uso é uma vantagem marcante: usuários navegam pelos arquivos com interfaces intuitivas (como exploradores de arquivos), adicionam pastas sem alterar hardware e podem aplicar medidas de segurança (como senhas). No entanto, o sistema torna-se ineficiente com grandes volumes de dados. A expansão exige upgrades de hardware, elevando custos e complexidade de gerenciamento (YANG; XIONG; REN, 2020).

Apesar das limitações, é uma solução popular pela flexibilidade e capacidade de compartilhamento entre usuários, ideal para cenários que demandam organização simples e acesso centralizado (“What is file storage? | IBM”, 2018).

5.2.2 Armazenamento em blocos - Block storage

Diferente do armazenamento de arquivos, o armazenamento em bloco (block storage) divide os dados em blocos individuais, cada um com um identificador único. Esses blocos são armazenados de forma independente e distribuídos em diferentes sistemas operacionais ou ambientes (como SANs ou nuvem), permitindo otimização de espaço e adaptação às configurações do sistema. Essa abordagem é ideal para cenários que demandam transferência de dados rápida e confiável, como bancos de dados e aplicações empresariais (“File storage, block storage, or object storage?”, 2019).

Sua principal vantagem é a separação entre dados e ambiente do usuário, criando múltiplos caminhos de acesso. Quando um arquivo é solicitado, o sistema remonta os blocos dinamicamente, garantindo eficiência. No entanto, o modelo tem limitações:

- **Metadados básicos:** Armazena apenas atributos simples (ex.: nome, tamanho), exigindo camadas adicionais (como bancos de dados) para informações complexas;
- **Custo elevado:** Requer infraestrutura especializada (ex.: SANs) e profissionais qualificados, tornando-o uma opção cara, especialmente em larga escala (“File storage, block storage, or object storage?”, 2019)."*

5.2.3 Armazenamento de objetos - object Storage

Abordando os armazenamentos e seus tipos é importante reconhecer que a eficácia de qualquer tipo de armazenamento vai além de simplesmente organizar e recuperar dados. Por tanto neste estudo é optado por focar a análise em armazenamento de objetos, uma escolha motivada por sua versatilidade, sua tolerância a falhas, baixo custo e escalabilidade para lidar com grandes volumes de dados não estruturados (KULKARNI et al., 2012). Diferentemente do armazenamento em bloco, Armazenamento de objetos, ou também conhecido como Object Storage, adota uma abordagem mais granular e flexível utilizando uma arquitetura onde os arquivos são fragmentados e distribuídos pelo hardware. Explorando esse tipo de armazenamento, torna-se evidente como esse tipo soluciona alguns problemas encontrados em outros tipos de armazenamento, oferecendo uma solução eficiente para organização de dados em larga escala e não estruturados.

O modelo opera principalmente por meio de APIs HTTP (ZHENG et al., 2012), amplamente suportadas por ferramentas e bibliotecas em diversas linguagens de programação. Essa integração simplificada, aliada à escalabilidade quase ilimitada, torna-o ideal para ambientes de nuvem pública, onde os custos são proporcionais ao uso real de armazenamento. Além disso, sua eficiência no gerenciamento de dados estáticos (como backups e arquivos de mídia) compensa limitações como a

impossibilidade de modificar objetos parcialmente – exigindo a reescrita completa do arquivo (GRACIA-TINEDO et al., 2016).

No entanto, enfrenta desafios significativos. A imutabilidade dos objetos, por exemplo, dificulta a integração com bancos de dados tradicionais, que dependem de atualizações incrementais. A latência em operações de escrita também pode ser um obstáculo para aplicações sensíveis a tempo real. Por fim, a implementação exige adaptações no desenvolvimento de softwares, já que interagir diretamente com APIs requer uma abordagem mais especializada comparada a sistemas de arquivos convencionais. Essas limitações reforçam a importância de avaliar o contexto específico antes de adotar o modelo, garantindo que suas vantagens superem os limites técnicos.

5.3 Tolerância a falhas

A integridade e disponibilidade dos dados são críticas em sistemas de armazenamento, especialmente em contextos sensíveis como saúde ou finanças, onde falhas podem resultar em perdas financeiras ou humanas. A tolerância a falhas, conforme definida por Tanenbaum em "Sistemas Distribuídos: Princípios e Paradigmas", refere-se à capacidade de um sistema se recuperar automaticamente de falhas de hardware ou software, garantindo operação contínua mesmo em cenários adversos. Um exemplo ilustra essa necessidade: a empresa Koingo Software perdeu mais de 6 mil fotos e vídeos após um erro no iCloud, evidenciando os riscos de concentrar dados em um único serviço (NASCIMENTO, 2023).

Ainda de acordo com o livro de Tanenbaum, a tolerância a falhas utiliza técnicas para garantir a integridade do sistema mesmo diante a falhas como por exemplo problemas de rede ou problemas com disco, podemos utilizar redundância e replicação, gravando dados em mais de um disco por exemplo, ou havendo mais de um sistema rodando para que em caso de um falhar seja possível utilizar um outro sistema, dados de aplicativos, armazenamento de mídias, compartilhamento assim evitando perdas, o grande objetivo é que o sistema tenha um autocuidado e se recupere. É evidente que uma solução eficaz de tolerância a falhas é essencial para alcançarmos uma melhor disponibilidade, confiabilidade e performance em casos de falhas.

Quando falamos de sistemas de armazenamento, para alcançar resiliência, técnicas como redundância e replicação são empregadas. Na redundância, dados são armazenados em múltiplos discos ou sistemas, enquanto a replicação cria cópias distribuídas geograficamente, permitindo acesso contínuo mesmo se um nó falhar. Metadados registram a localização dessas cópias, direcionando solicitações para réplicas disponíveis quando o original está inacessível. Essa abordagem é complementada por estratégias como o checkpointing, que salva o estado de processos em tempo real em nós secundários. Em caso de falha durante o processamento (como em transações financeiras ou fluxos de dados), o sistema retoma a operação a partir do último estado estável registrado, minimizando interrupções (TANENBAUM et al., 2008).

Ao utilizar replicação no armazenamento de dados, que desempenha um papel fundamental na recuperação de dados pós-desastre, é importante assegurar que os

dados e a sua replicação estejam em locais diferentes. A implementação consiste em um conjunto de recursos de armazenamento e processamento que trabalham com um ou mais servidores de object storage, também conhecidos como ‘nós’, juntos esses nós formam um grande repositório unificado para armazenamento.

5.4 Escalabilidade

O armazenamento baseado em objetos consolidou-se como a solução preferencial para dados não estruturados devido à sua capacidade de escalar de forma flexível e contínua ("Armazenamento de objetos: uma introdução | IBM", 2021). Diferentemente de sistemas hierárquicos (file storage) ou baseados em blocos (block storage), a arquitetura de object storage elimina limites pré-definidos de capacidade, permitindo expansão horizontal para exabytes por meio da adição simples de nós de armazenamento (Ju et al., 2011).

Um exemplo emblemático dessa escalabilidade ocorreu durante as eleições presidenciais brasileiras, quando o Tribunal Superior Eleitoral (TSE) utilizou a infraestrutura da AWS para lidar com 1,5 milhão de solicitações por segundo — três vezes acima do esperado ("Por dentro do uso da nuvem nas eleições", 2022). A combinação de object storage com uma rede de distribuição de conteúdo (CDN), implementada em parceria com a Embratel, garantiu acesso em tempo real aos resultados da apuração em 17 localidades, demonstrando como a escalabilidade dinâmica é crítica para cenários de alta demanda.

Essa flexibilidade é viabilizada por server pools (piscinas de servidores), grupos de nós independentes que operam em conjunto, mas com recursos dedicados. Cada pool pode ser adicionado a um cluster existente sem rebalanceamento complexo, permitindo que novos dados sejam distribuídos automaticamente. Em ambientes multiusuário, cada grupo mantém isolamento lógico, compartilhando apenas a infraestrutura física subjacente (Ju et al., 2011). Essa abordagem simplifica a expansão, tornando-a uma tarefa ágil e previsível, mesmo em sistemas já consolidados.

5.5 Custos para Armazenamento de Dados on-premise.

A infraestrutura on-premise envolve custos significativos, como aquisição de hardware, licenças de software, manutenção predial e treinamento de equipe. Diferentemente da nuvem, onde os gastos são operacionais (OPEX), o modelo local demanda investimentos de capital (CAPEX), incluindo atualizações periódicas de hardware para evitar obsolescência. Estudos como o de Shastri Pothukuchi et al. (2022) destacam que estratégias como automação e virtualização podem reduzir o Total Cost of Ownership (TCO) em até 40%, mitigando despesas com energia, refrigeração e substituição de componentes. No entanto, o controle direto sobre a infraestrutura justifica essa abordagem em cenários que exigem segurança crítica ou dados sensíveis, onde a terceirização para nuvem pública é inviável.

Ao abordarmos os desafios relacionados aos custos do armazenamento de dados, é importante buscar soluções que proporcionem não apenas eficiência financeira, mas também controle direto sobre a infraestrutura. Nesse contexto, a abordagem

on-premise, onde as empresas mantêm seus próprios servidores e sistemas de armazenamento, surge como uma alternativa significativa. Destacando-se nesse cenário, o MinIO surge como uma solução que promete eficiência e escalabilidade para ambientes locais.

5.6. Minio

Explorar essa ferramenta específica que desempenha um papel crucial torna-se essencial nesse contexto, e é aqui que o MinIO se destaca. Esta solução de object storage oferece uma abordagem inovadora para gerenciar dados de maneira eficiente e escalável. No âmbito do nosso estudo de caso, o MinIO não apenas fornece controle de custos, mas também contribui para a robustez e flexibilidade no armazenamento local.

Aprofundando ainda mais, examinamos como o MinIO se torna uma escolha estratégica, não apenas em termos de custos, mas também em desempenho e controle de dados. Em um cenário de gestão de dados em constante evolução, o MinIO destaca-se como uma solução dinâmica e inovadora no armazenamento de objetos. Seu design de código aberto oferece flexibilidade para diversos ambientes, desde nuvens públicas até implementações locais.

A arquitetura distribuída e orientada a objetos do MinIO possibilita que as organizações atinjam níveis excepcionais de desempenho e confiabilidade, especialmente em ambientes on-premise. Esta exploração mais aprofundada nos permite entender como o MinIO se posiciona como um componente essencial em nosso estudo de caso sobre armazenamento escalável com tolerância a falhas em ambientes locais.

5.6.1 Escalabilidade com o Minio

O MinIO adota a simplicidade como princípio central para escalabilidade em armazenamento de objetos. Sua arquitetura horizontal, baseada em Pools de Servidores, permite expandir a capacidade de forma modular: cada pool opera como um conjunto autônomo de nós, com recursos dedicados de computação, rede e armazenamento. Em ambientes multi-inquilino, clusters de pools coexistem na mesma infraestrutura física, mantendo isolamento total entre namespaces. A adição de novos pools é simplificada, exigindo apenas a integração de seus endereços ao cluster existente, sem interromper operações em andamento (“Scaling up MinIO Internal Connectivity”, 2023).

Três pilares sustentam essa escalabilidade:

Eliminação de Rebalanceamento: Dados existentes permanecem em seus pools originais, enquanto novos dados são direcionados a pools recém-adicionados. Essa abordagem evita o custo e a complexidade de redistribuir dados massivos, comum em sistemas tradicionais.

Suporte a Infraestrutura Heterogênea: O MinIO permite a integração de hardware diversificado (servidores, discos, fornecedores) em cada pool, adaptando-se a upgrades graduais sem exigir padronização rígida.

Domínios de Falha Gerenciáveis: A segmentação em pools limita o impacto de falhas a um grupo específico de nós. Atualizações ou falhas em um pool não afetam outros, garantindo resiliência em ambientes compartilhados.

Essa combinação de simplicidade e eficiência posiciona o MinIO como uma solução escalável para cenários que exigem crescimento contínuo sem comprometer a estabilidade operacional.

5.6.2 Tolerância a falhas com o minio

O MinIO combina escalabilidade horizontal e tolerância a falhas de forma integrada, tornando-o uma solução robusta para ambientes de armazenamento críticos. A arquitetura baseada em Pools de Servidores não apenas simplifica a expansão de capacidade, mas também fortalece a resiliência do sistema: cada pool opera de forma isolada, limitando o impacto de falhas a um grupo específico de nós. Essa abordagem garante que atualizações ou interrupções em um pool não afetem a disponibilidade dos demais.

A resiliência é reforçada pelo Erasure Coding, técnica que divide os dados em fragmentos e calcula códigos de correção, permitindo que o sistema recupere informações mesmo após falhas em até metade de suas unidades ("Gracefully handling disk failures in MinIO", 2023). Essa redundância inteligente elimina a necessidade de replicação completa, reduzindo custos de armazenamento enquanto mantém a durabilidade dos dados.

Ao integrar server pools e erasure coding, o MinIO assegura que a escalabilidade não comprometa a confiabilidade. Por exemplo, em um cluster com 16 nós, a perda simultânea de 8 unidades não interrompe o acesso aos dados — uma característica vital para aplicações que exigem disponibilidade contínua, como sistemas eleitorais ou financeiros. Essa dualidade entre crescimento flexível e resistência a falhas posiciona o MinIO como uma solução completa para ambientes dinâmicos e demandantes.

5.7 Object Storage em nuvens públicas

MinIO em Ambientes On-Premise vs. Nuvem Pública

Embora o MinIO se destaque em infraestruturas locais pela escalabilidade, controle e flexibilidade, as nuvens públicas como AWS S3, Google Cloud Storage e Azure Blob Storage oferecem vantagens complementares, como elasticidade sob demanda e redundância geográfica automatizada. Esses serviços replicam dados em múltiplos data centers globalmente, garantindo disponibilidade contínua mesmo em falhas

regionais, além de integração nativa com ecossistemas de análise, machine learning e CDNs (Content Delivery Networks).

Recursos como controle de acesso granular, versionamento de objetos e armazenamento em camadas (ex: acesso frequente vs. arquivamento) são padrão nessas plataformas, reduzindo custos para grandes volumes de dados estáticos. No entanto, diferentemente do MinIO, que permite personalização total em ambientes on-premise, as nuvens públicas operam em modelos pay-as-you-go, onde custos variáveis podem escalar com o tráfego ou requisições.

A escolha entre as abordagens depende de prioridades organizacionais: o MinIO é ideal para dados sensíveis ou ambientes regulatórios restritivos, enquanto soluções em nuvem pública são mais adequadas para escalabilidade global e integração com serviços gerenciados. Para muitas empresas, uma estratégia híbrida — usando MinIO localmente e nuvem para backups ou distribuição — oferece o melhor equilíbrio entre controle, custo e resiliência.

5.8 On-premise e Cloud computing

A escolha entre ambientes on-premise e cloud computing depende de prioridades organizacionais, como controle, custos e escalabilidade. O modelo on-premise, baseado em infraestrutura física própria, oferece controle total sobre dados sensíveis e conformidade regulatória rigorosa, sendo ideal para setores como saúde e financeiro. No entanto, demanda investimentos significativos em hardware (CAPEX), manutenção e atualizações tecnológicas, além de equipe especializada para gestão contínua.

Já o cloud computing elimina a necessidade de infraestrutura física, operando sob um modelo pay-as-you-go (OPEX) que ajusta custos conforme a demanda. Plataformas como AWS, Azure e Google Cloud oferecem escalabilidade instantânea, redundância geográfica e integração com serviços gerenciados (ex: IA, análise de dados). Entretanto, custos variáveis, como taxas de transferência de dados e requisições, podem aumentar imprevisivelmente, exigindo monitoramento rigoroso.

Um estudo do MIT (FISHER, 2018) revela que organizações com capacidade interna robusta podem economizar a longo prazo com on-premise, enquanto a nuvem é mais vantajosa para quem prioriza agilidade e redução de custos iniciais. A terceirização de infraestrutura via nuvem também libera recursos para outras áreas estratégicas, como inovação, ao reduzir a dependência de equipes dedicadas a hardware e redes.

A nuvem híbrida surge como uma alternativa equilibrada, combinando controle local com elasticidade da nuvem pública. Nesse modelo, dados críticos são armazenados on-premise, enquanto cargas de trabalho temporárias ou distribuídas globalmente são hospedadas na nuvem. Essa abordagem otimiza custos e desempenho, permitindo, por exemplo, usar a nuvem para backups automatizados ou distribuição de conteúdo via CDN, enquanto mantém operações sensíveis em servidores locais.

Em resumo, a decisão deve considerar fatores como sensibilidade dos dados, orçamento, capacidade técnica e objetivos de crescimento. Não há uma solução universal, mas a análise crítica dessas variáveis permite selecionar a arquitetura mais eficiente para cada contexto ("On-premise vs Cloud computing: quais as diferenças?", 2021).

5.9 Comparativo de Soluções de Armazenamento Escalável

Para orientar a escolha de tecnologias alinhadas aos requisitos de ambientes on-premise, é fundamental comparar soluções de armazenamento escalável. A Tabela 1 contrasta características técnicas do MinIO, Ceph, GlusterFS e Amazon S3, destacando arquitetura, escalabilidade, tolerância a falhas e custos. Essa análise subsidia a decisão pela adoção do MinIO no estudo de caso proposto, considerando simplicidade e eficiência em ambientes locais.

Tabela 1: Comparativo entre soluções de armazenamento escalável

Critério	MinIO	Ceph	GlusterFS	Amazon S3
Tipo de Armazenamento	Objeto	Objeto, Bloco, Arquivo	Arquivo Distribuído	Objeto
Arquitetura	Distribuída, Server Pools	Clusterizado	Federado (Bricks e Volumes)	Nuvem Pública (Serviço Gerenciado)
Escalabilidade	Horizontal (adição de nós/pools)	Horizontal (expansão de nós)	Horizontal (adição de bricks)	Elasticidade automática sob demanda
Tolerância a Falhas	Erasure Coding, Replicação	Replicação, Distribuição de Dados	Replicação, Distribuição de Dados	Redundância geográfica automática
Facilidade de Implementação	Configuração simples	Complexa (requer componentes Ceph)	Moderada (configuração de volumes)	Totalmente gerenciado (sem infraestrutura)
Custo	Open-Source (custos fixos com infraestrutura)	Open-Source (custos fixos com infraestrutura)	Open-Source (custos fixos com infraestrutura)	pague conforme o uso (custos variáveis)
Casos de Uso	Dados não estruturados, On-Premise	Nuvens privadas, SDS	Sistemas de arquivos distribuídos	Nuvem pública, dados estáticos
Integração com Cloud	Híbrida (on-premise + nuvem)	Híbrida	Limitada (foco em on-premise)	Nativa

Metadados	Ricos	Depende da interface	Básicos	Ricos
Desempenho	Alta velocidade para objetos	Balanceado (depende da configuração)	Alta para arquivos	baixa latência em requisições
Suporte a APIs	Compatível com S3	APIs S3 e Swift	Protocolos de arquivo (NFS)	API S3

Fonte: *Elaborado pelo autor com base em (BOCCHI et al., 2024), (ABOUZAID, 2025), (DONVITO; MARZULLI; DIACONO, 2014) (*DESHMUKH, S. C.; DESHMUKH, S. S., 2015), (AWS, 2023), (DAHER, Z.; HAJJDIAB, H., 2018).

Conforme evidenciado na Tabela 1, o MinIO destaca-se pela arquitetura simplificada e facilidade de implementação, critérios essenciais para implementações on-premise. Enquanto o Ceph e o GlusterFS demandam configurações complexas, o MinIO oferece compatibilidade nativa com APIs S3 e escalabilidade horizontal sem rebalanceamento, fatores decisivos para sua adoção neste trabalho. Além disso, sua tolerância a falhas via Erasure Coding alinha-se aos requisitos de resiliência explorados no estudo de caso.

6 Estudo de caso

O objetivo deste estudo de caso é demonstrar a notável escalabilidade e efetiva tolerância a falhas proporcionadas pela implementação do MinIO em ambientes de armazenamento de objetos. Pretendemos destacar como essa solução não apenas se adapta dinamicamente ao aumento de demanda, garantindo eficiência operacional, mas também oferece robustez ao lidar com situações imprevistas, reforçando sua confiabilidade.

6.1 Detalhes

Foi desenvolvido um protótipo de Sistema de Gerenciamento de Arquivos que visa fornecer uma solução robusta para armazenamento eficiente de arquivos, com ênfase em escalabilidade e tolerância a falhas. A aplicação é projetada para atender às necessidades de organizações e empresas que lidam com grandes volumes de dados.

6.2 Funcionalidades Principais:

- Upload de Arquivos:
    - Possibilidade de carregar arquivos de vários formatos na plataforma.
- Download de Arquivos:
    - Facilidade de baixar arquivos armazenados no sistema.

- Busca de Arquivos:
    - Sistema eficiente de busca para localizar rapidamente arquivos.
- Exclusão de Arquivos:
    - Remoção segura de arquivos do sistema.

6.3 Arquitetura e Tecnologias Utilizadas:

A aplicação segue uma arquitetura Model-View-Controller (MVC), com o front end desenvolvido em TypeScript e Angular, e o backend em Java com o framework Spring e integração com o MinIO.

O Sistema é implementado em containers Docker, garantindo consistência e facilitando a escalabilidade da aplicação.

Ambiente de Teste do object storage

- **Hardware:**
    - **Inicial:** 4 servidores, cada um com 2 discos.
    - **Expandido:** 8 servidores (16 discos) após teste de escalabilidade.
- **Software:**
    - MinIO (versão RELEASE.2023-11-15), configurado com *Erasure Coding* (esquema 4+2).
    - Docker para orquestração (ALLESSON182, 2023).

6.3 Resultados

Este estudo de caso abrange dois cenários experimentais: o primeiro foca na escalabilidade, enquanto o segundo se concentra em tolerância a falhas.

6.3.1 Cenário 1: Escalabilidade Horizontal

Objetivo: Validar a capacidade de expansão dinâmica do cluster MinIO.

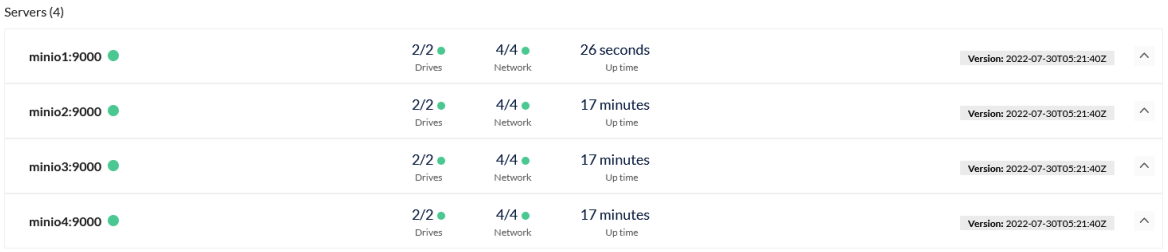
Etapas:

1. **Fase inicial (4 servidores):**
    - Upload de variados arquivos.
2. **Expansão do Cluster (8 servidores):**
    - Adição de 4 novos servidores via atualização do docker-compose.yml.
3. **Testes de Carga Contínua:**
    - Simulação de usuários realizando uploads e downloads de arquivos para validar distribuição automática e funcionamento do armazenamento.

A aplicação foi inicialmente implantada com o sistema possuindo 4 servidores, cada um com 2 drivers totalizando 8. Posteriormente, foi realizado um teste manual

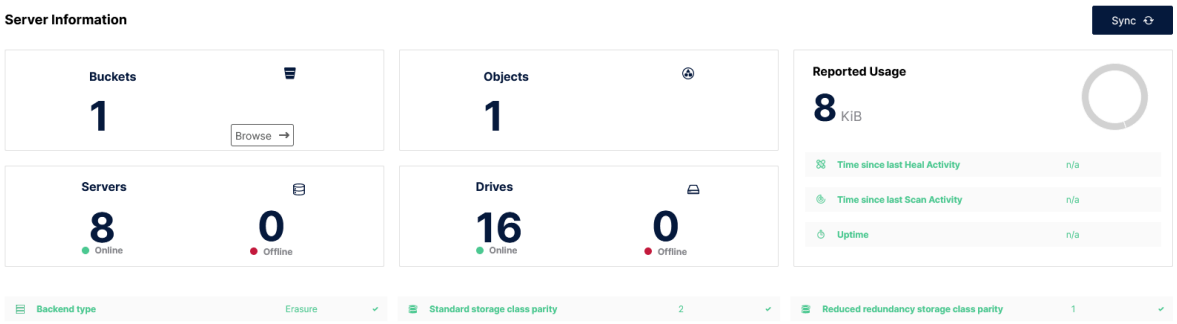
simulando usuários realizando o upload de vários arquivos, então foi observado que o sistema replicou automaticamente os objetos, gerando redundância automática para os arquivos enviados.

Imagem 1 – Cluster com 4 Servidores e cada um com 2 drivers



Ao adicionar mais espaço aos servidores, o docker compose com as configuração dos servidores foi atualizado para que fosse adicionado um novo conjunto de servidores, adicionando mais 1 um conjunto com 4 servidores, e em cada um deles, 2 drivers, totalizando 16, a replicação automática expandiu-se para esses nós adicionados.

Imagem 2 – Informações do servidor com mais armazenamento.



6.3.1.1 Conclusão do cenário 1

Constatou-se que foi possível expandir a infraestrutura de servidores mediante a aplicação de escalabilidade horizontal, com o sistema implementando automaticamente mecanismos de redundância. No entanto, ao empregar tecnologias de orquestração de containers Docker, observou-se que o sistema não consegue realizar expansões sem interrupções operacionais. Durante a expansão por meio do Docker Compose, verificou-se a necessidade de reinicialização do sistema para que as alterações fossem efetivadas. Essa limitação configura um potencial risco operacional em ambientes que exigem armazenamento contínuo de arquivos críticos ou em contextos onde a indisponibilidade temporária do serviço é inaceitável.

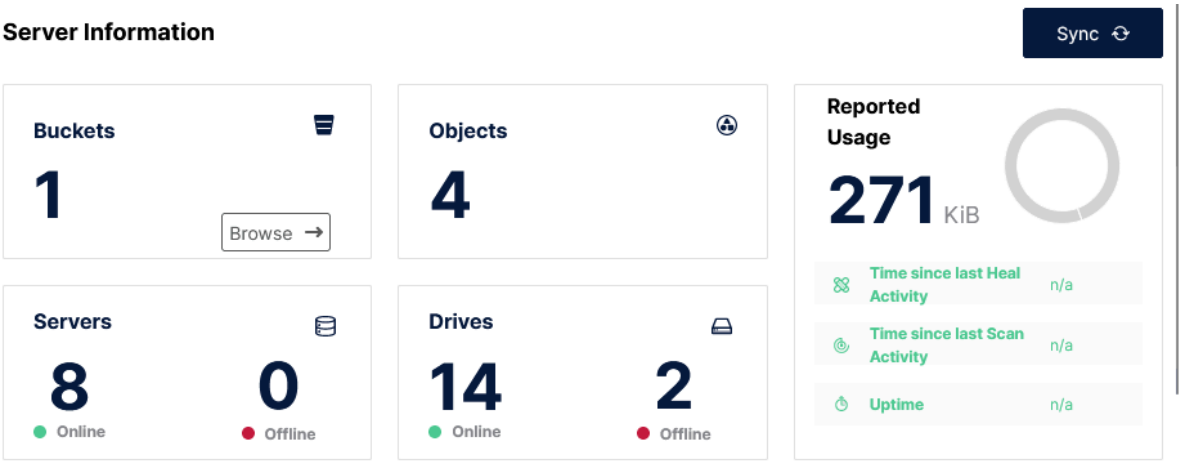
6.3.2 Cenário 2: Tolerância a Falhas

Objetivo: Avaliar a resiliência do sistema em falhas críticas.
Etapas:

1. **Falha Controlada:**
    - Remoção abrupta de 2 discos durante uploads e downloads.
2. **Teste de Disponibilidade:**
    - Download dos arquivos durante a falha.

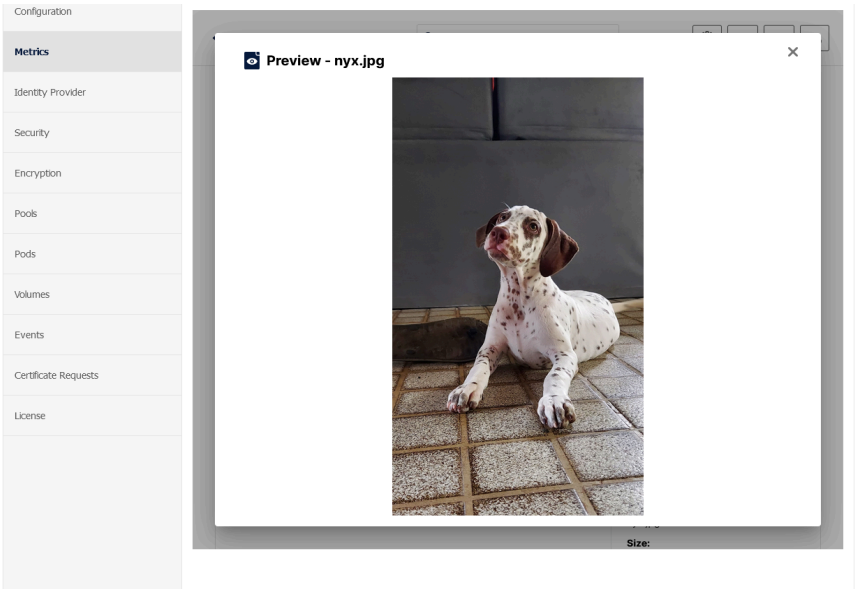
No cenário de tolerância a falhas, uma quantidade de 2 nós é manualmente removida para simular uma falha no sistema.

Imagem 3 – Informações do cluster com drivers falhos.



Após a remoção manual de 2 drivers, 1 em 2 diferentes servidores, para simular uma falha, é realizado um teste de disponibilidade de dados na aplicação, onde foi feito um teste simulando usuários realizando manualmente vários downloads dos arquivos previamente enviados aos servidores, e foi observado que o sistema continuou operando corretamente e não houve perda de nenhum dado.

Imagem 4 – Dados disponíveis após falha em alguns drivers.



6.3.2.1 Conclusão do cenário 2

Verificou-se que o sistema mantém a operacionalidade mesmo em cenários de falha simultânea de duas unidades de armazenamento em uma configuração distribuída de 8 servidores, cada um dotado de 2 discos. Entretanto, ressalta-se a necessidade crítica de alinhar a implementação técnica aos limites de redundância suportados pela infraestrutura. A excedência da taxa máxima de falhas toleráveis pelo esquema de redundância configurado pode resultar em comprometimento da disponibilidade dos dados, caracterizando um cenário de risco à integridade do armazenamento. Portanto, destaca-se a importância de monitoramento contínuo da capacidade de resiliência do sistema para mitigar potenciais interrupções não planejadas.

7 Considerações finais

Em um cenário marcado pelo incessante aumento do volume de dados diários, a seleção da estratégia de armazenamento emerge como um fator de grande importância para o sucesso operacional das organizações. Nesse contexto, o presente artigo ofereceu uma exploração abrangente, abordando os desafios inerentes ao armazenamento de dados não estruturados e fornecendo soluções para essas complexidades.

Desde a concepção do Armazenamento Definido por Software (SDS) até a discussão sobre escalabilidade, ressalta-se a importância fundamental da Tolerância a Falhas e da Escalabilidade como elementos-chave nesse panorama. O MinIO destacou-se como uma solução notável, apresentando-se não apenas como um sistema de escalabilidade dinâmica, mas também robusto.

As contribuições deste estudo para o campo de armazenamento de dados são notáveis, especialmente no que tange a sistemas escaláveis e tolerantes a falhas. O MinIO figura como uma ferramenta essencial para a gestão eficiente de recursos em ambientes de armazenamento de objetos.

Ao reconhecer as limitações inerentes ao estudo, abrimos espaço para futuros trabalhos. Sugerimos extensões no sistema, otimizações adicionais e a exploração de áreas como criptografia e comparações entre soluções de armazenamento de objetos. Esses direcionamentos prometem enriquecer a aplicabilidade do MinIO em variados cenários, consolidando sua posição como uma solução versátil e robusta.

O código-fonte desenvolvido para este sistema está disponível publicamente no

GitHub e pode ser acessado pelo link: https://github.com/allesson182/FileIsland.

REFERÊNCIAS

ABOUZAID, A. et al. **Building a modern data platform based on the data**

lakehouse architecture and cloud-native ecosystem. fev. 2025.

AKASHDUBEY-MS. **Optimize costs for Blob storage with reserved capacity - Azure Storage**. Disponível em: <https://learn.microsoft.com/en-us/azure/storage/blobs/storage-blob-reserved-capacity>. Acesso em: 29 jan. 2024.

ALLESSON182. **FileIsland/compose.yaml at master · allesson182/FileIsland**. Disponível em: <https://github.com/allesson182/FileIsland/blob/master/compose.yaml>. Acesso em: 2 mar. 2025.

AMARAVADI, C. The Grand Challenges in Information Systems. **Journal of Software Engineering and Applications**, v. 15, n. 04, p. 103–115, 2022.

ARPACI-DUSSEAU, R. H. **OPERATING SYSTEMS : three easy pieces.** S.L.: Createspace, 2018.

Armazenamento de objetos: uma introdução | IBM. Disponível em: <https://www.ibm.com/br-pt/topics/object-storage>. Acesso em: 4 fev. 2024.

Armazenamento definido por software: o que é e para que serve. Disponível em: <https://blog.eveo.com.br/armazenamento-definido-por-software>.

AWS. **What is Amazon S3? - Amazon Simple Storage Service**. Disponível em: <https://docs.aws.amazon.com/AmazonS3/latest/userguide/Welcome.html>.

DRAGO, I.; BORGES VIEIRA, A.; COUTO DA SILVA, **A. Caracterização dos Arquivos Armazenados no Dropbox.** 2013. Disponível em: <http://sbrc2013.unb.br/files/anais/wp2p/artigos/artigo-11.pdf>. Acesso em: 23 fev. 2024.

BARAL, SUMATI. **Popularity of Amazon S3-Cloud Storage**. Disponível em: <https://www.academia.edu/42846642/Popularity_of_Amazon_S3_Cloud_Storage>. Acesso em: 1 mar. 2025.

BOCCHI, E. et al. Enabling Storage Business Continuity and Disaster Recovery with Ceph distributed storage. **EPJ Web of Conferences**, v. 295, p. 01021–01021, 1 jan. 2024.

CHANDRAMOULI, R.; PINHAS, D. Security Guidelines for Storage Infrastructure. **NIST Special Publication 800-209**, 26 out. 2020.

DA, R. et al. **Utilização de Armazenamento Definido por Software em uma Arquitetura Hiperconvergente de Containers**. [s.l: s.n.]. Disponível em: <http://icts.unb.br/jspui/bitstream/10482/36759/1/2019_RodrigodaSilvaLeiteMorais.pdf>. Acesso em: 9 dez. 2023.

DAHER, Z.; HAJJDIAB, H. **Cloud storage comparative analysis amazon simple storage vs. Microsoft azure blob storage**. *International Journal of Machine Learning and Computing*, v. 8, n. 1, p. 85-89, 2018.

DESHMUKH, S. C.; DESHMUKH, S. S. **Simple Application of GlusterFs: Distributed file system for Academics**. **International Journal of Computer Science and Information Technologies (JUCSIT)**, Pune, India, v. 6, n. 3, p. 2972-2974, 2015.

DONVITO, G.; MARZULLI, G.; DIACONO, D. Testing of several distributed file-systems (HDFS, Ceph and GlusterFS) for supporting the HEP experiments analysis. **Journal of Physics: Conference Series**, 2014.

FISHER, C. Cloud versus On-Premise Computing. **American Journal of Industrial and Business Management**, v. 08, n. 09, p. 1991–2006, 2018.

File storage, block storage, or object storage? Disponível em: <https://www.redhat.com/en/topics/data-storage/file-block-object-storage>.
GRAEBIN, L.; HAMBURGO, N.; DE, J. **ESTUDO COMPARATIVO DE SISTEMAS DE ARQUIVOS DISTRIBUÍDOS**. [s.l: s.n.]. Disponível em: <https://tconline.feevale.br/tc/files/0001_1043.pdf>. Acesso em: 8 dez. 2023.
GOOGLE. Cloud Storage. Disponível em: <https://cloud.google.com/storage>.Acesso em: 12 fev. 2024.

Gracefully handling disk failures in MinIO. Disponível em: <https://blog.min.io/troubleshooting-disk-failures/>. Acesso em: 11 fev. 2024.

GUPTA, B.; MITTAL, P.; MUFTI, T. A Review on Amazon Web Service (AWS), Microsoft Azure & Google Cloud Platform (GCP) Services. **Proceedings of the 2nd International Conference on ICT for Digital, Smart, and Sustainable Development, ICIDSSD 2020, 27-28 February 2020, Jamia Hamdard, New Delhi, India**, 2021.

JU, J. et al. A Survey on Cloud Storage. **Journal of Computers**, v. 6, n. 8, 1 ago. 2011.

LITCHFIELD, A.; ALTHOUSE, J. **A Systematic Review of Cloud Computing, Big Data and Databases on the Cloud**. [s.l: s.n.]. Disponível em: <https://web.archive.org/web/20200323121749id_/https://aisel.aisnet.org/cgi/viewcontent.cgi?referer=&httpsredir=1&article=1215&context=amcis2014>. Acesso em: 8 dez. 2023.
MACEDO, R. et al. A Survey and Classification of Software-Defined Storage Systems. **ACM Computing Surveys**, v. 53, n. 3, p. 1–38, 31 maio 2021.
NASCIMENTO, D. **Empresa é paralisada após perda total de fotos do iCloud [atualizado]**. Disponível em: <https://macmagazine.com.br/post/2023/03/22/empresa-e-paralisada-apos-perda-total-de-fotos-do-icloud/>. Acesso em: 7 fev. 2024.

Por dentro do uso da nuvem nas eleições. Disponível em: <https://www.baguete.com.br/noticias/02/12/2022/por-dentro-do-uso-da-nuvem-nas-eleicoes>. Acesso em: 11 fev. 2024.

Preço Amazon S3 - AWS. Disponível em: <https://aws.amazon.com/pt/s3/pricing/>. Acesso em: 29 jan. 2024.
SAEED, I.; BARAS, S.; HAJJDIAB, H. **Security and Privacy of AWS S3 and Azure Blob Storage Services**. Disponível em: <https://ieeexplore.ieee.org/abstract/document/8821735/>. Acesso em: 18 jul. 2022.

Scaling up MinIO Internal Connectivity. Disponível em: <https://blog.min.io/scaling-up-minio-internal-connectivity/>. Acesso em: 9 fev. 2024.

SHASTRI POTHUKUCHI, A.; VASUDA KOTA, L.; MALLIKARJUNARADHYA, V. A CRITICAL ANALYSIS OF THE CHALLENGES AND OPPORTUNITIES TO OPTIMIZE STORAGE COSTS FOR BIG DATA IN THE CLOUD. **Asian Journal of Multidisciplinary Research & Review (AJMRR) ISSN 2582**, v. 8088, n. 1, 2022.

Stocator: Providing High Performance and Fault Tolerance for Apache Spark Over Object Storage | IEEE Conference Publication | IEEE Xplore. Disponível em: <https://ieeexplore.ieee.org/abstract/document/8411062>. Acesso em: 10 dez. 2023.

TANENBAUM, A. S. et al. **Sistemas distribuidos : principios y paradigmas**. México: Pearson Educación, Cop, 2008.

Temos mais dados do que nunca. Como usá-los a nosso favor? Disponível em: <https://exame.com/carreira/dados-uso-favor/>.

What is file storage? | IBM. Disponível em: <https://www.ibm.com/topics/file-storage>.

What is software-defined storage? Disponível em: <https://www.redhat.com/en/topics/data-storage/software-defined-storage>.

YANG, P.; XIONG, N.; REN, J. Data Security and Privacy Protection for Cloud Storage: A Survey. **IEEE Access**, v. 8, p. 131723–131740, 2020.

ZHENG, Q. et al. COSBench: A Benchmark Tool for Cloud Object Storage Services. 1 jun. 2012.