ipt Instituto Politécnico de Tomar

Escola(s) Superior de Tecnologia de Tomar

Storage Centralizado para Disponibilidade Crítica de Dados

Projeto

Frederico Mac-Mahon de Victória Pereira Marques Carneiro

Mestrado em Engenharia Informática - Internet das Coisas

Tomar. novembro, 2025

Esta página foi intencionalmente deixada em branco.

Instituto Politécnico de Tomar

Escola(s) Superior de Tecnologia de Tomar

Storage Centralizado para Disponibilidade Crítica de Dados

Projeto

Frederico Mac-Mahon de Victória Pereira Marques Carneiro

Orientado por:

Prof.ª Doutora Ana Cristina Lopes - IPT
Prof. Doutor Luís Oliveira - IPT

Projeto apresentado ao Instituto Politécnico de Tomar
para cumprimento dos requisitos necessários à obtenção do grau
de Mestre em Engenharia Informática

Esta página foi intencionalmente deixada em branco.

Informação sobre o Autor, Orientadores e Instituição

Autor:	Frederico Carneiro aluno24391@ipt.pt
Orientadores:	Prof.ª Doutora Ana Cristina Lopes anacris@ipt.pt Prof. Doutor Luís Oliveira loliveira@ipt.pt
Instituição de acolhimento:	Instituto Politécnico de Tomar
Instituição que concede o grau:	Instituto Politécnico de Tomar

REPÚBLICA PORTUGUESA

EDUCAÇÃO, CIÊNCIA E INOVAÇÃO

UNIÃO EUROPEIA

Fundo Social Europeu

Esta página foi intencionalmente deixada em branco.

"Data is the new oil."
— Clive Humby

Esta página foi intencionalmente deixada em branco.

Agradecimentos

A realização deste trabalho representou uma etapa de grande relevância académica e pessoal, cuja concretização só foi possível graças ao contributo, dedicação e apoio de várias pessoas e instituições, às quais manifesto o meu mais sincero reconhecimento. Em primeiro lugar, expresso o meu profundo agradecimento a todos os professores do Mestrado em Engenharia Informática – Internet das Coisas, do Instituto Politécnico de Tomar, pelo rigor científico, pela dedicação constante e pela partilha de conhecimento ao longo de todo o percurso académico. Cada unidade curricular contribuiu de forma significativa para o desenvolvimento das competências técnicas e analíticas que sustentam o trabalho aqui apresentado. Um agradecimento muito especial é devido à Huawei Portugal pela disponibilização dos equipamentos essenciais para a concretização deste projeto, sem os quais a validação experimental das soluções propostas não teria sido possível. Agradeço, em particular, ao engenheiro Rui Fialho pela disponibilidade demonstrada, e de forma ainda mais especial ao engenheiro João Carmona, cujo apoio, orientação e profundo conhecimento técnico foram determinantes para o sucesso desta investigação. Expresso igualmente a minha gratidão aos meus orientadores, Professora Doutora Ana Cristina Lopes e Professor Doutor Luís Oliveira, pela orientação científica, pela confiança depositada e pela constante disponibilidade para o esclarecimento de dúvidas e partilha de sugestões valiosas que contribuíram para o aperfeiçoamento deste trabalho. Agradeço também aos colegas e amigos do curso, pelo espírito de entreajuda, pela troca de ideias e pela motivação mútua que tornaram este percurso mais enriquecedor e colaborativo. O ambiente de partilha e cooperação vivido durante o mestrado foi, sem dúvida, um fator determinante para a consolidação dos conhecimentos adquiridos. Por fim, deixo um sincero agradecimento à minha família, pelo apoio incondicional, paciência e incentivo permanente ao longo de todo este processo. O suporte familiar foi essencial para a superação dos desafios encontrados e para a concretização deste objetivo académico.

A todos os que, de forma direta ou indireta, contribuíram para o desenvolvimento deste projeto, deixo o meu mais genuíno reconhecimento e gratidão.

Esta página foi intencionalmente deixada em branco.

Resumo

A crescente dependência das organizações em infraestruturas digitais resilientes tem impulsionado a necessidade de soluções avançadas de armazenamento centralizado e proteção de dados. Este trabalho investiga a implementação de uma arquitetura ativo-ativo para armazenamento distribuído, combinada com um mecanismo de proteção contra ransomware baseado em armazenamento isolamento físico da rede e deteção de anomalias. A solução propositada garante alta disponibilidade, continuidade de serviço e proteção avançada contra ataques cibernéticos, alinhando-se com as exigências de recuperação de desastres e resiliência organizacional.

A abordagem adotada envolve a replicação síncrona de dados entre dois sistemas de armazenamento operando em modo ativo-ativo, permitindo que ambas as instâncias processem operações de leitura e escrita de forma simultânea. Esta configuração elimina pontos únicos de falha e reduz o impacto de falhas inesperadas. Para garantir a consistência dos dados e evitar situações de split-brain, são implementados mecanismos de arbitragem e decisão automática em caso de falha de comunicação entre os sistemas.

A solução de proteção contra ransomware complementa esta infraestrutura através da deteção inteligente de anomalias e da utilização de um sistema isolado que mantém cópias imutáveis dos dados do ambiente de produção. A integração destas tecnologias permite identificar e mitigar ataques antes que comprometam a integridade da informação, bem como garantir uma recuperação segura e eficiente sem necessidade de pagamento de resgates.

O estudo inclui a implementação prática e testes de desempenho, comparando a resiliência de uma solução de armazenamento com um único equipamento e uma solução ativo-ativo. Os resultados obtidos demonstram que a combinação destas tecnologias aumenta significativamente a fiabilidade da infraestrutura de armazenamento e reduz os riscos associados à perda de dados, contribuindo para a continuidade operacional e segurança da informação empresarial.

Palavras-chave: Armazenamento ativo-ativo, replicação síncrona, recuperação de desastres, resiliência organizacional, proteção contra ransomware, arbitragem de armazenamento, disponibilidade contínua.

Esta página foi intencionalmente deixada em branco.

Abstract

The increasing reliance of organizations on resilient digital infrastructures has driven the need for advanced centralized storage and data protection solutions. This study explores the implementation of an active-active architecture for distributed storage, combined with a ransomware protection mechanism based on network-isolated storage and anomaly detection. The proposed solution ensures high availability, service continuity, and advanced cybersecurity protection, aligning with disaster recovery and organizational resilience requirements.
The adopted approach involves synchronous data replication between two storage systems operating in an active-active mode, allowing both instances to simultaneously handle read and write operations. This configuration eliminates single points of failure and mitigates the impact of unexpected failures. To ensure data consistency and prevent split-brain scenarios, arbitration mechanisms and automatic decision-making are implemented in the event of communication failures between systems.
The ransomware protection solution complements this infrastructure through intelligent anomaly detection and an isolated storage system that maintains immutable copies of production data. The integration of these technologies allows for the early identification and mitigation of attacks, preserving data integrity while ensuring a secure and efficient recovery process without the need to pay ransom demands.
This study includes a practical implementation and performance testing, comparing the resilience of a single-device storage solution with an active-active configuration. The results demonstrate that the combination of these technologies significantly enhances storage infrastructure reliability and reduces the risks associated with data loss, contributing to business continuity and information security.

Keywords: Active-active storage, synchronous replication, disaster recovery, organizational resilience, ransomware protection, storage arbitration, continuous availability.

Índice

Esta página foi intencionalmente deixada em branco.

Lista de Figuras

Lista de Tabelas

Esta página foi intencionalmente deixada em branco.

Lista de Excertos de Código

Esta página foi intencionalmente deixada em branco.

Esta página foi intencionalmente deixada em branco.

Capítulo 1

Introdução

1.1 Enquadramento

A rápida transformação digital verificada nas últimas décadas impulsionou a criação de enormes quantidades de dados em múltiplos setores de atividade, pressionando organizações a adotarem soluções tecnológicas cada vez mais robustas para armazenarem e protegerem informação vital. Neste contexto, os dados assumem um papel central na operação e na tomada de decisões das organizações, sendo um ativo estratégico. A fiabilidade e a disponibilidade da informação são determinantes para o sucesso e a sustentabilidade das empresas, tornando a sua proteção uma prioridade. A perda de dados, seja por falhas técnicas, ataques cibernéticos ou desastres naturais, pode gerar consequências severas, comprometendo a continuidade do negócio, a reputação e a conformidade regulatória das organizações.
Estudos indicam que a maior parte das empresas que sofrem uma perda massiva de dados encerram as suas operações num período de seis meses. Esta realidade impõe a necessidade de estratégias robustas de proteção e recuperação da informação, que assegurem não apenas a integridade dos dados, mas também a resiliência operacional das infraestruturas de armazenamento.
Regulamentações como o Digital Operational Resilience Act (DORA) (Parliament & the Council of the European Union, 2022), aprovado pela União Europeia, reforçam a importância da resiliência digital, impondo requisitos rigorosos de continuidade operacional, recuperação de desastres e cibersegurança, especialmente no setor financeiro.
A evolução dos ataques de ransomware, que visam diretamente infraestruturas críticas de armazenamento, tem vindo a intensificar este cenário. Relatórios recentes apontam que 79% das organizações já foram alvo de ataques de ransomware nos últimos 12 meses, tornando imperativo o desenvolvimento de soluções que combinem alta disponibilidade, replicação de dados e proteção contra ameaças emergentes.
Este trabalho investiga as arquiteturas e tecnologias de armazenamento centralizado, explorando soluções que garantam disponibilidade contínua dos dados, tolerância a falhas e proteção contra ciberameaças.
Ao longo deste estudo, serão analisadas estratégias de replicação, segurança e resiliência digital, com o objetivo de fornecer diretrizes eficazes para a implementação

de infraestruturas de armazenamento adaptadas às exigências atuais das empresas.

1.2 Motivação

O armazenamento centralizado desempenha um papel essencial na proteção dos dados empresariais, funcionando como a última camada de resiliência numa arquitetura de tecnologias de informação, na medida em que é nele que residem as cópias definitivas e as réplicas que permitem recuperar a informação quando todos os outros mecanismos falham. Mais do que um simples repositório, estas infraestruturas constituem a base para garantir a continuidade das operações, assegurando que a informação permanece acessível e íntegra perante desafios como falhas de hardware, erros de configuração, ataques de *ransomware*, incidentes de segurança interna ou interrupções de serviço em componentes intermédios. Apesar disso, muitas organizações continuam a adotar abordagens predominantemente reativas, sem explorar de forma sistemática soluções como replicação síncrona ou assíncrona entre sistemas, *snapshots* frequentes, cópias de segurança imutáveis, reservas de dados isoladas e mecanismos de monitorização e automatização da recuperação, que permitem elevar significativamente o nível de proteção e a eficiência da gestão de dados.

A motivação para este estudo surge da necessidade de democratizar o acesso a tecnologias e metodologias que reforcem a resiliência do armazenamento centralizado. Se por um lado, as grandes corporações investem em infraestruturas sofisticadas, por outro, muitas empresas ainda desconhecem estratégias eficazes que poderiam estar ao seu alcance sem necessidade de investimentos desproporcionados. A evolução das tecnologias de storage tem permitido a adoção de soluções mais flexíveis e escaláveis, adaptáveis a diferentes realidades empresariais, tornando possível implementar mecanismos de proteção avançados com recursos ajustados à dimensão e às necessidades de cada organização.

Neste contexto, torna-se essencial identificar e avaliar soluções que combinem eficiência, escalabilidade e viabilidade económica, permitindo que empresas de diferentes dimensões possam reforçar as suas infraestruturas sem comprometer o desempenho ou os custos operacionais. A integração de tecnologias de replicação avançadas, sistemas de snapshots e estratégias de recuperação inteligentes pode representar a diferença entre um incidente controlado e um colapso operacional.

Assim, torna-se imperativo compreender como as infraestruturas de armazenamento podem evoluir de simples sistemas de retenção de dados para verdadeiras garantias de continuidade operacional, permitindo às empresas operar com maior segurança e previsibilidade no seu crescimento e sustentabilidade digital.

1.3 Objetivos

Este trabalho tem como principal objetivo analisar e testar soluções de armazenamento centralizado, garantindo alta disponibilidade, resiliência e segurança dos dados empresariais, apresentando soluções realistas e exequíveis para a maioria das empresas, independentemente da sua dimensão ou setor de atividade. Para alcançar este propósito, serão realizados diversos testes e implementações, abordando diferentes aspetos da infraestrutura de armazenamento.

Os objetivos específicos deste estudo incluem:

1. **Comparação de arquiteturas de armazenamento centralizado**
   - realizar testes de benchmarking para comparar a performance e a resiliência de duas soluções diferentes, uma solução com um único equipamento (single point of failure), uma solução com dois equipamentos em modo ativo-ativo.

2. **Instalação e configuração de equipamentos de storage**
   - Efetuar a instalação de raiz dos dispositivos de armazenamento, garantindo a correta parametrização para um desempenho otimizado.
   - Configurar a rede SAN (Storage Area Network), assegurando comunicação eficiente e segura entre os servidores e os sistemas de armazenamento.

3. **Testes de redundância e tolerância a falhas**
   - Avaliar a capacidade de recuperação da infraestrutura através de testes de redundância de hardware, simulando falhas em componentes críticos para verificar a resposta do sistema.

4. **Implementação de mecanismos de proteção contra ataques cibernéticos**
   - Instalar e configurar uma solução de airgap, garantindo uma estratégia eficaz de isolamento de dados para prevenção contra ransomware.

5. **Automação da monitorização e manutenção do storage**
   - Desenvolver scripts de ”daily health check”para envio automático de relatórios sobre o estado dos equipamentos, permitindo detetar precocemente falhas e otimizar a gestão da infraestrutura.

Com esta abordagem, o estudo pretende demonstrar como diferentes configurações e estratégias podem influenciar a resiliência, segurança e disponibilidade dos sistemas de armazenamento centralizado, fornecendo diretrizes para a sua implementação em ambientes empresariais.

1.4 Contribuições e limitações

Este estudo propõe-se a aprofundar a análise de arquiteturas de armazenamento centralizado em modo ativo-ativo, uma abordagem que, embora não seja uma inovação recente, permanece subutilizada devido a perceções de complexidade e potenciais impactos na latência.

1.4.1 Contribuições

- **Desmistificação do impacto da tecnologia ativo-ativo**
  A investigação foca-se na implementação prática desta tecnologia, analisando o impacto real na performance, resiliência e segurança dos dados, em comparação com arquiteturas tradicionais baseadas em ponto único de falha. Esta abordagem permitirá demonstrar que a latência associada a infraestruturas ativo-ativo pode ser controlada e otimizada, contrariando a perceção de que este modelo compromete o desempenho.

- **Demonstração da versatilidade da implementação ativo-ativo**
  Este trabalho destaca a flexibilidade das infraestruturas ativo-ativo, que podem ser configuradas tanto no mesmo local, assegurando redundância interna, como em sites geograficamente distintos, aumentando a resiliência contra falhas regionais e desastres naturais. Esta característica permite que as empresas escolham a abordagem mais adequada às suas necessidades, otimizando a relação desempenho, custo e segurança.

- **Distribuição inteligente das cargas de trabalho**
  A capacidade de distribuir dinamicamente as cargas de trabalho entre múltiplos equipamentos reduz a sobrecarga em sistemas individuais, melhora a eficiência do processamento e minimiza o impacto de falhas, garantindo que os dados permanecem sempre acessíveis com latências controladas.

- **Relevância para a Continuidade de Negócio e Recuperação de Desastres**
  A solução proposta alinha-se diretamente com os princípios da Continuidade de Negócio, ao apresentar Objetivo de Ponto de Recuperação (RPO - *Recovery Point Objective*) = 0 (nenhuma perda de dados em caso de falha) e Objetivo de Tempo de Recuperação (RTO - *Recovery Time Objective*) = 0 (recuperação instantânea). Além disso, a sua implementação permite a configuração em modo de Recuperação de Desastres (DR - *Disaster Recovery*), garantindo que, em caso de falha grave num data center, a organização pode continuar a operar sem interrupções. Tendo as infraestruturas bem configuradas, passa a ser possível atingir uma taxa de disponibilidade de ”sete noves”.

1.4.2 Limitações

Apesar das vantagens em termos de maior disponibilidade de serviço e redução do risco de indisponibilidade prolongada, a adoção de uma solução ativo-ativo enfrenta alguns desafios que podem limitar a sua aplicação, sobretudo em empresas de menor dimensão:

- **Investimento financeiro**
  Embora escalável, esta tecnologia requer dois equipamentos de storage, dobrando o custo inicial quando comparado com uma solução de armazenamento único. No entanto, este investimento pode ser justificado pelo custo potencial da perda de dados e pelos valores exigidos em ataques de ransomware.

- **Requisitos de espaço físico**
  A necessidade de dois equipamentos implica um aumento significativo da área ocupada, o que pode ser um fator limitante em infraestruturas empresariais com restrições de espaço nos seus data centers.

- **Gestão e manutenção**
  A complexidade da configuração e da administração de dois sistemas em simultâneo pode representar um desafio adicional, exigindo equipas especializadas e processos de monitorização contínua para garantir o funcionamento correto da infraestrutura.

Esta página foi intencionalmente deixada em branco.

Capítulo 2

Estado da Arte

2.1 Introdução

A disponibilidade contínua de dados em infraestruturas empresariais e operacionais tem assumido um papel central na resiliência das organizações. No contexto da transformação digital, uma parte crescente das atividades de negócio passa a depender de serviços suportados por dados, o que faz com que os sistemas de armazenamento deixem de ser meros repositórios para se tornarem elementos estruturais da continuidade de serviço. Estudos de mercado evidenciam que uma fração significativa dos workloads empresariais é hoje gerida com requisitos de disponibilidade muito elevados, frequentemente expressos em objetivos de serviço próximos de cinco noves, o que traduz a sensibilidade do impacto económico e reputacional de interrupções de acesso aos dados em ambientes de missão crítica (Pearson, 2023).

Em paralelo, a modernização dos centros de dados tem conduzido à adoção generalizada de plataformas de armazenamento de alto desempenho, muitas vezes baseadas em tecnologias all flash e interfaces *Non Volatile Memory Express* (NVMe), capazes de responder a cargas de trabalho com forte sensibilidade à latência. Estas plataformas são desenhadas para consolidar workloads heterogéneos, incluindo aplicações transacionais, sistemas analíticos e serviços em tempo quase real, mantendo níveis elevados de previsibilidade do desempenho. A esta exigência de desempenho e escalabilidade acrescenta se a necessidade de suportar modelos híbridos e multicloud, garantindo uma experiência consistente de disponibilização e proteção de dados independentemente da localização física dos recursos de computação e armazenamento.

A crescente sofisticação das ciberameaças, em particular dos ataques de ransomware que visam diretamente infraestruturas de armazenamento e cópias de segurança, introduz um novo nível de complexidade no desenho de arquiteturas de storage para ambientes críticos. Orientações de organismos de referência como o National Institute of Standards and Technology (National Institute of Standards and Technology, 2024) salientam que a integridade dos dados não pode depender exclusivamente de mecanismos tradicionais de backup e recuperação, impondo a

necessidade de integrar processos de identificação, proteção, deteção, resposta e recuperação alinhados com frameworks de cibersegurança amplamente adotados. Este enquadramento promove a adoção de arquiteturas que incorporam mecanismos de monitorização de integridade, deteção de anomalias e capacidades de recuperação rápida, capazes de suportar objetivos de ponto de recuperação e de tempo de recuperação cada vez mais restritivos.

Em resposta a estas exigências, a indústria de armazenamento tem vindo a incorporar funcionalidades de ciber-resiliência diretamente nos arrays utilizados em ambientes de missão crítica, ou seja, sistemas de armazenamento que suportam aplicações e serviços cuja indisponibilidade tem impacto direto na operação do negócio.. Entre essas funcionalidades destacam-se a replicação síncrona e assíncrona entre sistemas, a possibilidade de operar em configurações ativo ativo entre centros de dados e a introdução de mecanismos específicos de proteção contra alterações destrutivas, como snapshots imutáveis, volumes apenas de leitura e cofres de dados logicamente ou fisicamente isolados (Sinclair & Keane, 2022). Estas capacidades complementam as abordagens clássicas de alta disponibilidade baseadas em redundância de componentes internos e eliminação de pontos únicos de falha.

A literatura e a prática industrial convergem, assim, na ideia de que a disponibilidade contínua dos dados resulta da combinação de duas dimensões complementares. A primeira dimensão corresponde ao desenho de arquiteturas de alta disponibilidade que asseguram continuidade de serviço perante falhas físicas ou lógicas, recorrendo a replicação entre sistemas, caminhos de acesso redundantes e topologias que eliminam pontos únicos de falha. A segunda dimensão corresponde à forma como os dados são protegidos contra perda, corrupção ou cifragem maliciosa, recorrendo a princípios como o modelo de múltiplas cópias, a diversificação de suportes e localizações, a imutabilidade de *snapshots* e a orquestração de processos de recuperação consistentes com as melhores práticas de cibersegurança (Storage Networking Industry Association, 2017a). A Figura 2.1 sintetiza algumas das tendências emergentes no domínio do armazenamento de dados, relacionando a evolução das arquiteturas de storage com as exigências atuais de disponibilidade, desempenho e ciber resiliência.

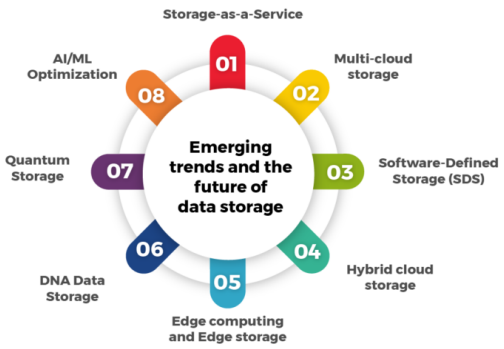


Figura 2.1: Tendências emergentes e o futuro do armazenamento de dados

Neste enquadramento, o presente capítulo procede a uma revisão dos desenvolvimentos mais relevantes no domínio do armazenamento centralizado para ambientes de missão crítica. São analisadas as principais arquiteturas e mecanismos de replicação em uso em sistemas empresariais, bem como as abordagens de proteção e recuperação de dados que procuram mitigar o impacto de falhas e de ciberincidentes. Esta análise estabelece a base conceptual necessária para enquadrar as soluções de armazenamento estudadas nos capítulos seguintes e para interpretar os resultados experimentais obtidos na comparação entre uma arquitetura com um único equipamento e uma arquitetura com dois equipamentos a operar em modo ativo ativo.

2.2 Arquiteturas e Tecnologias

A escolha da arquitetura de armazenamento adequada é um fator crítico na conceção de infraestruturas para dados de missão crítica. O desempenho, a escalabilidade, a resiliência e a capacidade de integração com outras infraestruturas de TI são fatores essenciais na decisão entre diferentes tecnologias. Atualmente, os *storage arrays* evoluíram de soluções convencionais, como *Redundant Array of Independent Disks* (RAID) (Patterson, Gibson, & Katz, 1988), *Storage Area Network* (SAN) (Riabov, 2004) e *Network Attached Storage* (NAS) (Ravi Kumar et al., 2021), para modelos altamente distribuídos, como *Software Defined Storage* (SDS) (Macedo, Paulo, Pereira, & Bessani, 2020) e *Object Storage* (Durner, Leis, & Neumann, 2023), muitas vezes integrados com computação na nuvem para garantir maior flexibilidade e disponibilidade (Figura 2.2).

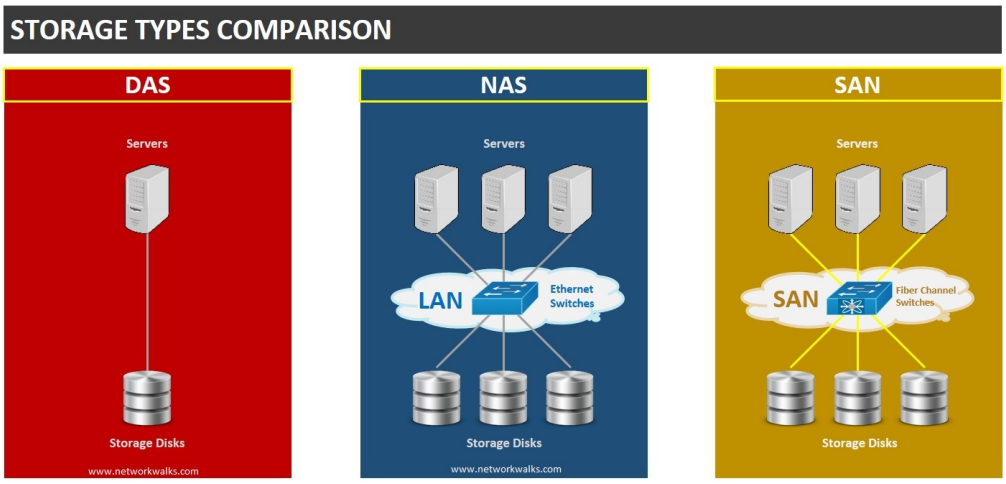


Figura 2.2: Sistemas de armazenamento DAS, NAS e SAN

Nos primórdios das infraestruturas de armazenamento empresarial, a ligação direta dos discos ao servidor, em configurações de *Direct Attached Storage* (DAS), constituía a abordagem dominante. Este modelo apresenta uma arquitetura simples e de baixa latência, mas limita a escalabilidade e a partilha eficiente de capacidade entre vários servidores, conduzindo frequentemente a ilhas de armazenamento

subaproveitado (Riabov, 2004). A introdução de NAS permitiu a disponibilização de dados ao nível de ficheiro, através de protocolos como NFS (*Network File System*) ou SMB (*Server Message Block*), simplificando a partilha de informação entre múltiplos clientes, embora com uma dependência maior da camada de rede e do sistema de ficheiros do servidor de armazenamento (Ravi Kumar et al., 2021). As SAN generalizaram o acesso em bloco através de redes dedicadas, tipicamente baseadas em Fibre Channel ou iSCSI, proporcionando maior previsibilidade de desempenho, caminhos redundantes de acesso e funcionalidades avançadas de gestão de volumes que se tornaram a base de muitas infraestruturas de missão crítica (Riabov, 2004).

Com o crescimento da escala dos centros de dados e da diversidade de workloads, surgiram arquiteturas orientadas ao controlo por software, em que o plano de controlo é desacoplado do plano de dados. As soluções de SDS promovem uma abstração lógica sobre múltiplos dispositivos físicos e permitem agregar, provisionar e gerir recursos de armazenamento de forma programável, independentemente do fabricante ou da topologia subjacente. Os sistemas SDS procuram responder a requisitos de elasticidade, automação e integração com plataformas de orquestração de *cloud*, frequentemente recorrendo a componentes distribuídos e a mecanismos de redundância como replicação e *erasure coding*, em que os dados são divididos em fragmentos e combinados com informação de redundância adicional, permitindo reconstruir a informação original mesmo na presença de falhas de vários discos ou nós, com menor sobrecarga de capacidade do que a replicação tradicional (Darrous, Ibrahim, & Perez, 2019). Este modelo facilita a criação de serviços de armazenamento com características diferenciadas, ajustados a perfis de desempenho e resiliência específicos.

Em paralelo, o *object storage* consolidou-se como a abordagem dominante para grandes volumes de dados não estruturados. Ao invés de expor blocos ou ficheiros, estas plataformas armazenam informação sob a forma de objetos, identificados por chaves e acedidos através de interfaces *RESTful*, muitas vezes compatíveis com a API do Amazon S3. Sistemas como Ceph demonstram a utilização de *object storage* distribuído com gateways compatíveis com S3, suportando multitenancy, versionamento e políticas de ciclo de vida, o que os torna particularmente adequados para cenários de *cloud* pública e privada. O facto dos objetos serem identificados individualmente através de chaves e distribuídos por múltiplos nós de armazenamento permite alcançar elevados níveis de escalabilidade e disponibilidade, embora com uma latência de acesso superior à das soluções de armazenamento em bloco de baixa latência..

A escolha da arquitetura de storage array depende das necessidades específicas de cada organização. Enquanto as SAN e NAS continuam a ser predominantes em ambientes empresariais tradicionais, particularmente em sistemas transacionais e plataformas de virtualização que requerem acesso em bloco de baixa latência, geralmente medidos na ordem de milissegundos, SDS, *Hyper Converged Infrastructure* (HCI) e *Object Storage* estão a emergir como padrões para aplicações distribuídas e *workloads* modernas. As soluções HCI integram computação, armazenamento e rede numa plataforma unificada, controlada por software, simplificando a gestão e

permitindo escalar os recursos de forma modular, o que é especialmente relevante em ambientes que procuram alinhar a infraestrutura *on premises* com modelos de *cloud* híbrida.

A adoção de NVMe e de extensões como *NVMe over Fabrics* reflete a necessidade crescente de desempenho e de acesso paralelo massivo, explorando diretamente a largura de banda de ligações PCIe e de redes de baixa latência. Em paralelo, técnicas de *erasure coding* são cada vez mais utilizadas em sistemas distribuídos para reduzir o overhead de armazenamento face à replicação tradicional, mantendo a capacidade de tolerar falhas múltiplas e, em muitos casos, otimizando o volume de dados transferido aquando do processo de reparação. Em conjunto com mecanismos de replicação síncrona e assíncrona entre data centers, estes desenvolvimentos permitem conceber arquiteturas de armazenamento que combinam elevada densidade, desempenho previsível e capacidade de recuperação rápida, características essenciais para ambientes de missão crítica.

2.2.1 Estratégias de Alta Disponibilidade e Replicação

A alta disponibilidade em sistemas de armazenamento centralizado assenta, em grande medida, na forma como os dados são replicados entre sistemas e locais distintos, de modo a cumprir objetivos de ponto de recuperação (RPO) e de tempo de recuperação (RTO) compatíveis com ambientes de missão crítica (Storage Networking Industry Association, 2017b). A replicação de dados desempenha, por isso, um papel central nestas estratégias, permitindo reduzir o tempo de indisponibilidade e limitar a perda de dados em caso de falha.

Nas plataformas de armazenamento empresarial, a replicação é frequentemente implementada ao nível do próprio array, recorrendo a mecanismos de duplicação remota entre sistemas. Em muitas soluções clássicas baseadas em Fibre Channel, como as que utilizam o software *HP StorageWorks Continuous Access EVA* (Hewlett-Packard, 2008), a replicação remota a nível de bloco é disponibilizada como funcionalidade nativa dos arrays, suportando cenários de continuidade de negócio entre centros de dados distintos . Nestes contextos, a replicação síncrona é normalmente utilizada em distâncias metropolitanas, assegurando que uma operação de escrita apenas é confirmada à aplicação após ser persistentemente gravada em ambos os lados, o que permite atingir RPO próximos de zero à custa de uma maior sensibilidade à latência da ligação.

Em paralelo, soluções mais recentes, como a família Dell PowerStore, incorporam replicação assíncrona nativa para volumes e ficheiros, tirando partido de sessões de replicação configuráveis e grupos de consistência para preservar relações transacionais entre múltiplos LUNs (Dell Technologies, 2022). Neste modelo, as escritas são concluídas localmente e propagadas para o sistema remoto com um desfasamento controlado, o que reduz o impacto da latência de rede sobre o desempenho percebido pela aplicação, mas implica a aceitação de um RPO não nulo em cenário de falha total do sítio primário.

A escolha entre replicação síncrona e assíncrona é, assim, determinada por um

compromisso entre requisitos de perda máxima admissível de dados, latência de acesso, distância entre centros de dados e custos de infraestrutura de rede. Em ambientes com forte sensibilidade à latência e requisitos de perda de dados nula apenas em falhas localizadas, é comum combinar replicação síncrona em curtas distâncias com mecanismos assíncronos para um terceiro local de recuperação, configurando cadeias de proteção em múltiplos níveis.

Para além do modo de replicação, o modelo de alta disponibilidade adotado influencia a forma como os recursos de armazenamento são expostos às aplicações. Em configurações ativo-passivo, um sistema é designado primário e o outro secundário, sendo este último promovido apenas em situação de falha ou manutenção planeada. Por contraste, as arquiteturas ativo-ativo permitem que múltiplas cópias dos dados sejam utilizadas em simultâneo por diferentes instâncias da aplicação ou por diferentes domínios geográficos, recorrendo a replicação bidirecional e a mecanismos de prevenção e resolução de conflitos. Soluções de replicação lógico-transacional, como as baseadas em tecnologias de replicação de bases de dados de terceiros, exploram este modelo para disponibilizar arquiteturas horizontalmente escaláveis com fortes requisitos de disponibilidade (Quest Software Inc., 2016).

A literatura e as boas práticas de fabricantes de bases de dados descrevem diversos padrões de implementação de centros de dados ativo–ativo, onde instâncias distribuídas partilham carga de trabalho e são mantidas em sincronismo através de mecanismos de replicação ao nível da base de dados e do armazenamento (Ray & To, 2008). Em paralelo, fornecedores de infraestruturas de armazenamento propõem soluções de centros de dados ativo–ativo baseadas em replicação síncrona entre arrays, frequentemente complementadas com servidores de quórum para arbitrar situações de partição de rede e evitar condições de *split–brain* (Huawei Technologies Co., Ltd., 2017). Estas arquiteturas permitem que, em caso de falha de um dos sítios, o outro continue a fornecer serviço sem necessidade de operações manuais de promoção ou de reconfiguração de caminhos de I/O.

Para maximizar a eficiência da replicação, as soluções de armazenamento empresarial recorrem a técnicas de otimização de capacidade e de redução de tráfego, entre as quais se destacam compressão, deduplicação de dados e utilização extensiva de snapshots. A deduplicação, em particular, permite eliminar blocos redundantes antes da transferência, reduzindo significativamente o volume de dados a replicar e o consumo de largura de banda, o que é especialmente relevante em cenários de replicação remota sobre ligações com capacidade limitada (Paulo & Pereira, 2014). Nas implementações modernas, estas técnicas são frequentemente aplicadas de forma transparente às aplicações e integradas com políticas de retenção e de proteção de dados.

Os mecanismos de snapshot e de clones instantâneos assentam em técnicas de *copy-on-write* ou *redirect-on-write*, permitindo capturar pontos de restauro consistentes com impacto reduzido no desempenho e sem necessidade de duplicar imediatamente todo o espaço de armazenamento (Bunn, Simpson, Peglar, & Nagle, 2004). Quando combinados com replicação remota, estes snapshots podem ser transmitidos para o sítio secundário como deltas incrementais, constituindo uma base eficiente tanto para recuperação operacional rápida como para estratégias de

recuperação em caso de desastre. Desta forma, as empresas conseguem alinhar os requisitos de alta disponibilidade com políticas de proteção de dados e de ciber-resiliência, reduzindo simultaneamente o risco de perda de informação e o tempo necessário para restabelecer o serviço.

As grandes empresas de storage empresarial adotam, assim, arquiteturas que combinam replicação síncrona e assíncrona, modos ativo–ativo e ativo–passivo, e técnicas avançadas de otimização de capacidade, como compressão, deduplicação (Figura 2.3) e snapshots, com o objetivo de minimizar o impacto na largura de banda, melhorar os tempos de recuperação e garantir níveis elevados de disponibilidade em ambientes de missão crítica.

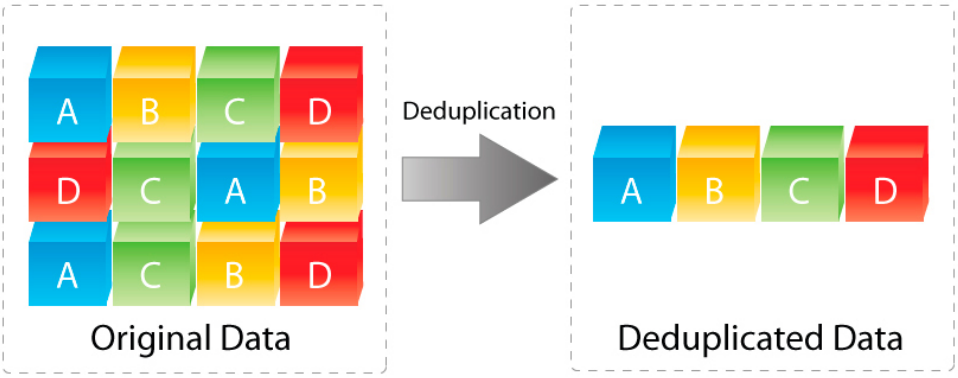


Figura 2.3: Redução de blocos após processo de Deduplicação

2.3 Impacto da Perda de Dados e Resiliência Organizacional

A perda de dados representa uma ameaça significativa para as organizações, afetando de forma direta a continuidade operacional, a reputação e a conformidade regulatória. As causas são diversas, incluindo falhas de hardware, desastres naturais, ataques cibernéticos e erros humanos. Estudos recentes evidenciam que o impacto financeiro médio de um incidente de violação de dados atinge vários milhões de dólares por evento, agregando custos de interrupção das operações, resposta técnica, notificações legais e perda de clientes (IBM Security, 2024).

O impacto financeiro pode ser substancial, abrangendo desde a interrupção das operações até multas por incumprimento de regulamentações como o Regulamento Geral de Proteção de Dados (RGPD), que prevê sanções significativas para violações de segurança que conduzam a perda ou divulgação indevida de dados pessoais (European Parliament and Council of the European Union, 2016). Além disso, a perda de dados sensíveis pode comprometer a confiança dos clientes e parceiros de negócios, resultando em danos reputacionais duradouros. Para mitigar

estes riscos, é essencial que as empresas adotem estratégias robustas de proteção e recuperação de dados, fortalecendo a sua resiliência organizacional.

No caso específico do ransomware, a perda de acesso aos dados pode paralisar por completo a atividade da organização, mesmo quando não ocorre exfiltração de informação sensível. Estudos longitudinais sobre famílias de ransomware mostram a evolução deste tipo de ameaça desde as primeiras campanhas até variantes altamente automatizadas e com cadeias de monetização bem estabelecidas (Kharraz, Robertson, Balzarotti, Bilge, & Kirda, 2015). Estes trabalhos documentam padrões de comportamento comuns, como a cifragem massiva de ficheiros, a destruição de cópias de sombra e a tentativa sistemática de comprometer cópias de segurança acessíveis a partir dos sistemas afetados, o que agrava o impacto operacional e financeiro das infeções.

O enquadramento de boas práticas proposto pelo National Institute of Standards and Technology trata a perda de dados e os ataques de ransomware como problemas de integridade da informação que exigem uma abordagem ao longo de todo o ciclo de vida do incidente. As publicações da série NIST SP 1800 salientam a necessidade de identificar e proteger ativos críticos antes do ataque (Cawthra et al., 2020b), detetar e responder rapidamente a alterações destrutivas (Cawthra et al., 2020a) e recuperar de forma fiável para um estado conhecido e íntegro (Townsend, McBride, Lusty, Sexton, & Ekstrom, 2020). Este enquadramento traduz se, na prática, em requisitos concretos sobre inventário de sistemas, segmentação de redes, monitorização de integridade, registo de eventos e, de forma particularmente relevante, definição e teste regular de estratégias de cópia de segurança.

A literatura recente tem vindo a reforçar a ideia de que a resiliência organizacional face a ataques de ransomware depende não apenas de mecanismos de prevenção, mas também de arquiteturas de armazenamento que tornem economicamente desinteressante o pagamento de resgates. O trabalho de Castiglione e Pavlovic analisa o ransomware como consequência de um desequilíbrio arquitetural entre processos dinâmicos e armazenamento estático e propõe o conceito de *dynamic distributed secure storage*, em que dados cifrados são replicados de forma distribuída por múltiplos nós (Castiglione & Pavlovic, 2020). Nesta perspetiva, a existência de cópias distribuídas e criptograficamente protegidas reduz o valor de chantagem associado à perda de uma instância local, desde que a organização disponha de mecanismos eficazes para reconstituir rapidamente o estado dos sistemas a partir dessas cópias.

Para mitigar o risco de perda de dados em ambientes de missão crítica, as organizações são incentivadas a combinar várias camadas de proteção. Entre estas incluem-se políticas de cópia de segurança que contemplem o princípio de múltiplas cópias em locais distintos, a utilização de snapshots frequentes e, quando tecnicamente viável, a adoção de cópias imutáveis e de reservas de dados logicamente ou fisicamente isoladas da infraestrutura de produção. A eficácia destas medidas depende da sua integração com controlos de acesso robustos, processos de gestão de vulnerabilidades e programas de sensibilização dos colaboradores, de forma a reduzir o risco de erro humano. Testes periódicos de disaster recovery, alinhados com objetivos de tempo e de ponto de recuperação previamente definidos, são fun-

damentais para validar que os procedimentos implementados permitem restaurar os serviços dentro dos limites aceitáveis para o negócio.

A continuidade do negócio depende, em última análise, da capacidade das empresas para proteger, monitorizar e recuperar os seus dados com eficiência, o que torna imperativo o investimento em infraestruturas de armazenamento resilientes e em estratégias de mitigação proativas. A integração de mecanismos de replicação, cópias de segurança robustas e arquiteturas de armazenamento projetadas para resistir a falhas e a ciberataques constitui, assim, um elemento estrutural da resiliência organizacional em ambientes de missão crítica.

Esta página foi intencionalmente deixada em branco.

Capítulo 3

Trabalho Desenvolvido

A tecnologia escolhida é uma solução de armazenamento ativo-ativo que permite que dois sistemas de armazenamento processem serviços simultaneamente, estabelecendo uma relação de backup mútuo para garantir maior continuidade de serviço e melhor utilização dos recursos de armazenamento. Atualmente, existem dois modos principais de funcionamento para infraestruturas de armazenamento ativo-ativo: modo ativo-passivo e modo ativo-ativo. No modo ativo-passivo, parte dos serviços é executada no equipamento A, enquanto o equipamento B atua como backup ativo, e vice-versa, proporcionando um nível aproximado de redundância. No entanto, no modo ativo-ativo, todos os caminhos de Input/Output (I/O) permanecem ativos e podem aceder simultaneamente a uma mesma *Logical Unit Number* (LUN) ativo-ativo, garantindo balanceamento de carga dos serviços e um failover transparente e automático. A tecnologia escolhida baseia-se numa arquitetura ativo-ativo, permitindo a replicação de dados entre dois equipamentos separados até 300 km, conforme indicado na documentação técnica da Huawei disponibilizada no âmbito deste projeto [1]. No caso de serviços de base de dados, a distância recomendada entre os centros de dados é até 100 km, de forma a garantir o menor impacto possível na latência. Esta solução permite *failover* automático e sem impacto nos serviços em caso de falha de um dos equipamentos ou até mesmo de um data center completo. Além disso, assegura um RPO igual a zero e um RTO próximo de zero, sendo que o RTO dependerá do sistema aplicacional e do modo de implementação.

3.1 Arquitetura

A solução de armazenamento implementada assenta numa arquitetura de armazenamento centralizado com dois sistemas all-flash configurados em modo ativo-ativo, interligados através de uma rede SAN Fibre Channel redundante. O objetivo principal desta arquitetura é assegurar a continuidade de serviço perante falhas de componentes individuais e permitir a comparação controlada entre um cená-

[1] Documentação técnica fornecida pela Huawei Portugal ao abrigo de acordo de parceria, não acessível publicamente.

rio com um único equipamento e um cenário com dois equipamentos a operar em simultâneo.

Do ponto de vista lógico, a solução organiza-se em três domínios principais. No domínio de computação, um conjunto de *hosts* com hipervisor suporta as máquinas virtuais responsáveis pela geração de carga e pela recolha de métricas. Estes *hosts* acedem aos volumes de armazenamento através de múltiplos caminhos Fibre Channel, explorando mecanismos de *multipathing* para distribuir o tráfego de I/O e garantir a manutenção de acesso em caso de falha de um adaptador de rede, de um switch SAN ou de uma porta de controladora.

No domínio de armazenamento, cada sistema é composto por duas controladoras em configuração redundante, com todos os volumes apresentados de forma simétrica aos *hosts*. Na configuração ativo-ativo, os volumes pertencem a grupos de consistência que são replicados de forma síncrona entre os dois sistemas, permitindo que operações de leitura e escrita sejam processadas em qualquer uma das controladoras disponíveis. Esta organização elimina pontos únicos de falha ao nível do equipamento de armazenamento e permite avaliar o comportamento da solução em cenários de falha de controladora ou de sistema.

Entre estes dois domínios existe o domínio de interligação SAN, constituído por switches Fibre Channel organizados em dois *fabrics* independentes, isto é, dois conjuntos de *switches* totalmente separados, tanto ao nível físico como lógico, cada um formando uma rede SAN autónoma capaz de encaminhar o tráfego de armazenamento de forma independente. Cada host estabelece ligações a ambos os *fabrics*, e cada controladora de cada sistema de armazenamento é igualmente ligada a ambos, formando uma topologia em malha completa que assegura caminhos redundantes entre qualquer host e qualquer sistema de storage. A definição lógica de zonas nos switches SAN limita a visibilidade recíproca entre portas, garantindo simultaneamente isolamento de tráfego e flexibilidade na gestão de ligações.

A par da rede SAN, a solução integra ainda redes IP dedicadas à gestão, monitorização e arbitragem entre sistemas. A rede de gestão suporta o acesso às consolas dos equipamentos e das plataformas de monitorização, enquanto a rede de quorum e de replicação entre os sistemas de armazenamento é utilizada para troca de informação de estado e coordenação das operações em modo ativo-ativo. Embora fisicamente distintas, estas redes funcionam de forma complementar à SAN Fibre Channel, contribuindo para a resiliência global da solução.

Esta arquitetura modular, baseada em múltiplos caminhos redundantes e na operação cooperativa de dois sistemas de armazenamento (Figura 3.1), estabelece o enquadramento necessário para os cenários de teste descritos nas secções seguintes, onde são comparados o comportamento de uma solução com um único equipamento e de uma solução ativo-ativo sob diferentes tipos de falha e perfis de carga.

3.1.1 Ambiente de Implementação

A infraestrutura utilizada na validação prática da solução foi concebida para reproduzir, de forma controlada, uma arquitetura de armazenamento centralizado representativa de um ambiente empresarial. O sistema experimental integrou dois equipamentos de armazenamento "Huawei Dorado 5000 V6"(Figura 3.2), configurados para operar em modo cooperativo, assegurando redundância e continuidade de serviço. Cada equipamento é constituído por duas controladoras independentes, interligadas entre si e conectadas simultaneamente a cada um dos *fabrics* SAN existentes, formando uma topologia em malha completa (*mesh*) que garante caminhos redundantes entre quaisquer pontos da infraestrutura de armazenamento.

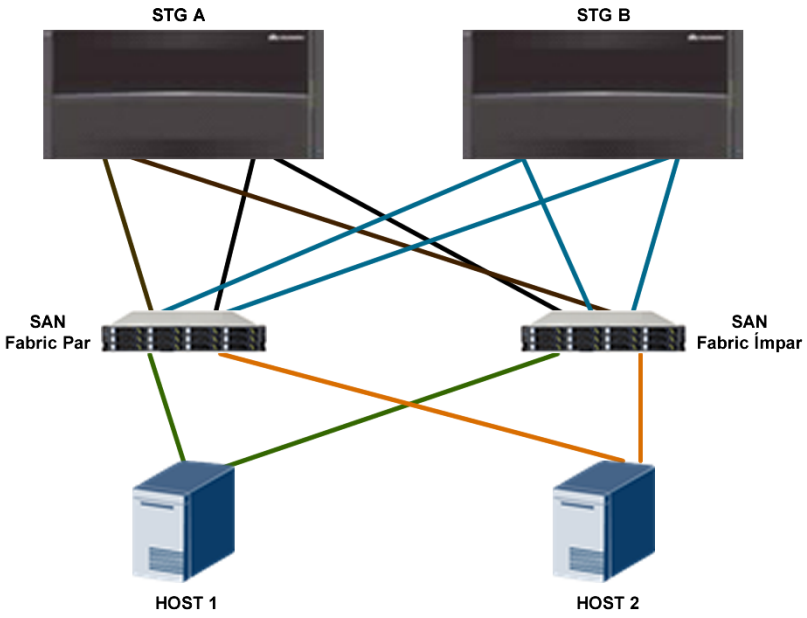


Figura 3.1: Ambiente físico de implementação

De igual modo, cada host físico participante nos testes foi ligado a ambos os fabrics SAN, assegurando a continuidade de acesso aos volumes de armazenamento mesmo em caso de falha de uma ligação ou de um fabric. As ligações entre controladoras, switches e hosts foram configuradas através de Soft Zoning, metodologia que permite a definição lógica das zonas de comunicação dentro dos SAN switches, assegurando isolamento e segurança na comunicação sem necessidade de configuração física rígida.

Esta arquitetura, baseada em múltiplos caminhos redundantes e numa configuração de malha completa, proporciona simultaneamente tolerância a falhas, balanceamento de carga e resiliência a nível de comunicação, refletindo as práticas de implementação utilizadas em ambientes empresariais de elevado desempenho.

A camada de computação foi constituída por dois hosts configurados com o hipervisor, sobre os quais foram implementadas três máquinas virtuais que de-

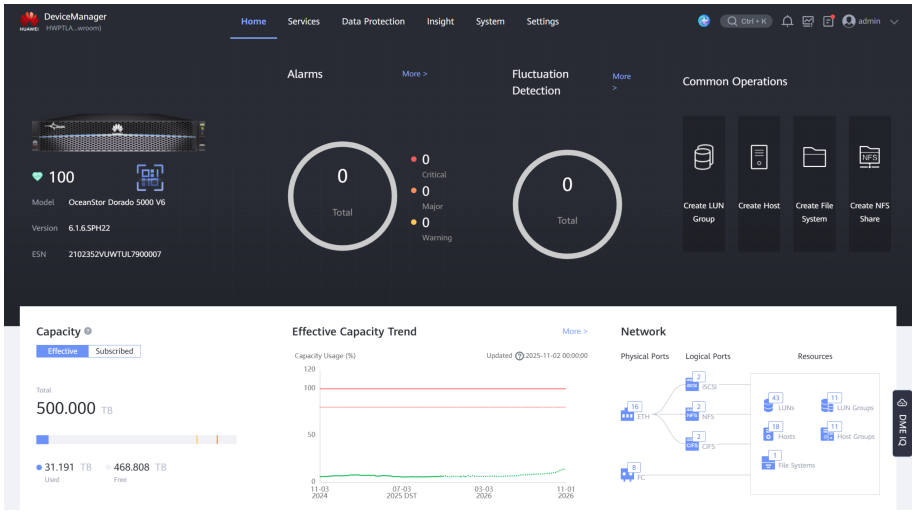


Figura 3.2: Consola de comando de um equipamento Huawei Dorado 5000 V6

sempenharam papéis distintos no contexto dos testes. Estas instâncias virtuais permitiram a execução isolada das cargas de trabalho, a recolha de métricas e o controlo das operações, garantindo a consistência das condições de ensaio. A adoção de virtualização possibilitou a segmentação das tarefas e o isolamento de variáveis externas, assegurando que as diferenças observadas entre os cenários decorrem exclusivamente da arquitetura de armazenamento subjacente.

Para a avaliação quantitativa do desempenho das soluções, recorreu-se ao *VDBench* (Oracle Corporation, 2023), ferramenta amplamente utilizada em estudos de caracterização de storage pela sua capacidade de gerar cargas de entrada e saída controladas e reprodutíveis. O sistema de monitorização e análise foi suportado por duas plataformas complementares: o *Grafana* (Grafana Labs, 2024), utilizado para visualização temporal das métricas, e o *Graphite* (Graphite Project, 2024), responsável pela recolha e armazenamento histórico dos dados de desempenho.

A execução dos ensaios e a gestão das métricas foram automatizadas através de um conjunto de scripts desenvolvidos especificamente para o efeito. Um script em Python assegurou a execução coordenada dos testes e a recolha sistemática das medições, enquanto outro script em PowerShell permitiu verificar o estado dos equipamentos de storage e dos volumes apresentados. Esta automação reduziu a intervenção manual, garantindo a repetibilidade das experiências e a consistência temporal dos resultados.

A infraestrutura descrita estabeleceu a base para os testes comparativos realizados nos capítulos seguintes, assegurando a neutralidade experimental necessária à análise entre a solução single e a solução ativo-ativo. O desenho modular do ambiente, aliado ao controlo rigoroso das variáveis de execução, permitiu garantir que as conclusões obtidas refletem o comportamento intrínseco das arquiteturas estudadas e não fatores externos ao sistema de armazenamento.

3.2 Infraestrutura Experimental

A infraestrutura experimental concebida para este estudo foi desenhada com o propósito de proporcionar um ambiente controlado, estável e integralmente replicável, adequado à avaliação comparativa do desempenho entre duas arquiteturas de armazenamento distintas. O desenho desta infraestrutura assegurou que as variáveis associadas ao hardware, à rede e ao sistema operativo permanecessem constantes em todas as execuções, permitindo que as diferenças observadas fossem exclusivamente atribuídas ao modo de operação do armazenamento.

Todo o ambiente experimental foi implementado sobre uma plataforma de virtualização baseada em hipervisor, utilizando dois servidores físicos configurados como hosts de virtualização. Sobre estes hosts foram criadas as máquinas virtuais necessárias à execução dos diferentes cenários de teste. Duas máquinas virtuais foram dedicadas à realização dos ensaios associados à solução ativo-ativo, enquanto uma terceira foi reservada para os testes da solução *single*. Em complemento, foi criada uma máquina virtual adicional com função de "Controladora", responsável pela orquestração das execuções, pela recolha centralizada de métricas e pelo controlo da sequência experimental.

A utilização de um ambiente virtualizado permitiu o isolamento das cargas de trabalho, a consistência dos recursos atribuídos e a repetibilidade controlada de cada ensaio. O subsistema de armazenamento foi constituído por dois equipamentos de armazenamento centralizado fisicamente independentes, mas logicamente configuráveis em diferentes modos de operação. Para os testes realizados, os equipamentos foram configurados para operar em modo ativo-ativo, permitindo que as LUNs fossem simultaneamente acessíveis a partir de ambos os equipamentos de storage e garantindo sincronização em tempo real e continuidade de serviço em caso de falha. No entanto, a mesma infraestrutura permitiu também a criação de volumes operando em modo *single*, atribuídos a apenas um dos equipamentos, sem qualquer replicação ou sincronização entre eles.

Desta forma, ambas as soluções testadas, *single* e ativo-ativo, foram implementadas sobre os mesmos recursos físicos, distinguindo-se apenas pela configuração lógica das LUNs e respetivos grupos. Para a solução single, foram utilizadas LUNs pertencentes a um grupo configurado exclusivamente num dos equipamentos de armazenamento (designado STG A), sem qualquer ligação ou dependência do segundo equipamento. Já na solução ativo-ativo, foram utilizadas LUNs pertencentes a um grupo configurado em modo ativo-ativo, acessível de forma simétrica a partir de ambos os equipamentos (STG A e STG B) (Figura 3.3). Esta abordagem assegurou condições idênticas de hardware e conectividade, permitindo que as diferenças observadas nos resultados fossem atribuídas unicamente à arquitetura de operação do armazenamento.

A comunicação entre os hosts e os equipamentos de armazenamento foi efetuada através de uma rede de armazenamento dedicada (SAN), suportada por dois switches configurados em redundância total. Esta topologia garantiu múltiplos caminhos ativos entre cada host e o armazenamento, assegurando não apenas

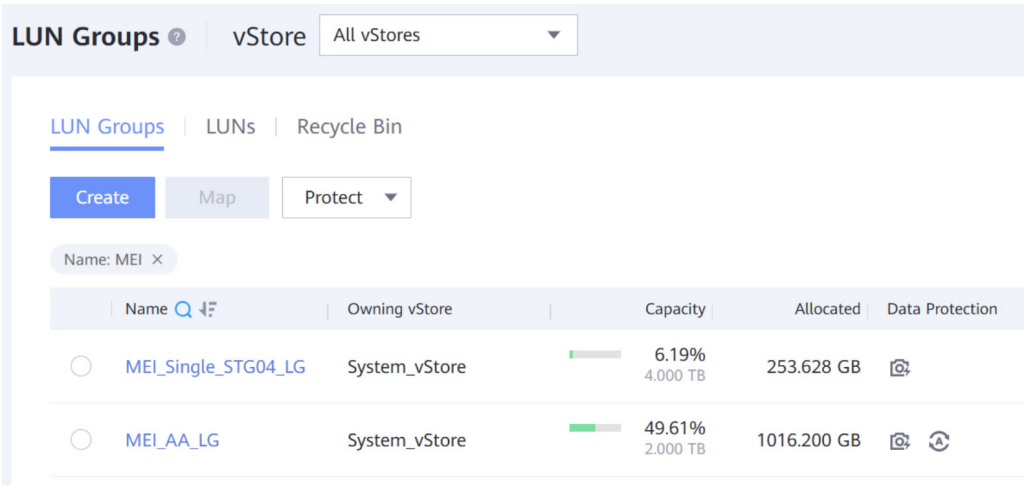


Figura 3.3: LUN Groups configurados para os cenários de teste, distinguidos pelo ícone de proteção de dados que indica sincronização entre equipamentos.

tolerância a falhas, mas também equilíbrio de tráfego e estabilidade de latência durante as execuções.

Para a monitorização e recolha de métricas, foram desenvolvidos scripts VDBench para automatizar as execuções e gerir os intervalos de amostragem, bem como scripts em Python, disponíveis no Apêndice A, para proceder à agregação estatística dos resultados obtidos. A preparação do ambiente experimental envolveu a instalação e parametrização de todos os componentes, incluindo a configuração das interfaces de rede de armazenamento, a criação e mapeamento das LUNs em ambos os cenários e a definição dos caminhos multipath de acesso.

No caso da solução ativo-ativo, foram ainda realizados os procedimentos de sincronização e associação entre os controladores, assegurando coerência de dados e disponibilidade contínua das LUNs. O desenho modular e reprodutível desta infraestrutura garante que o estudo possa ser replicado integralmente em ambientes equivalentes, independentemente do fabricante dos equipamentos, desde que estes suportem replicação síncrona e modos de operação ativo-ativo. Esta independência em relação ao fabricante assegura que os resultados obtidos são comparáveis e válidos, permitindo a reprodutibilidade das experiências em infraestruturas tecnicamente análogas.

3.3 Capacidades de Serviço Ativo-Ativo

A tecnologia escolhida permite a configuração de dois sistemas de armazenamento em modo ativo-ativo, garantindo sincronização em tempo real dos dados armazenados nas LUNs replicadas entre os dois locais (Figura 3.4). Ambos os sistemas processam operações de leitura e escrita simultaneamente, assegurando um acesso contínuo e paralelo por parte dos servidores de aplicação. Em caso de falha de um dos sistemas, a comutação de serviços ocorre de forma transparente e sem interrupções, garantindo a continuidade operacional e a integridade dos dados.

Em comparação com arquiteturas ativo-passivo, esta solução otimiza a utilização de recursos computacionais, reduz a comunicação desnecessária entre os arrays e minimiza os tempos de acesso ao armazenamento.

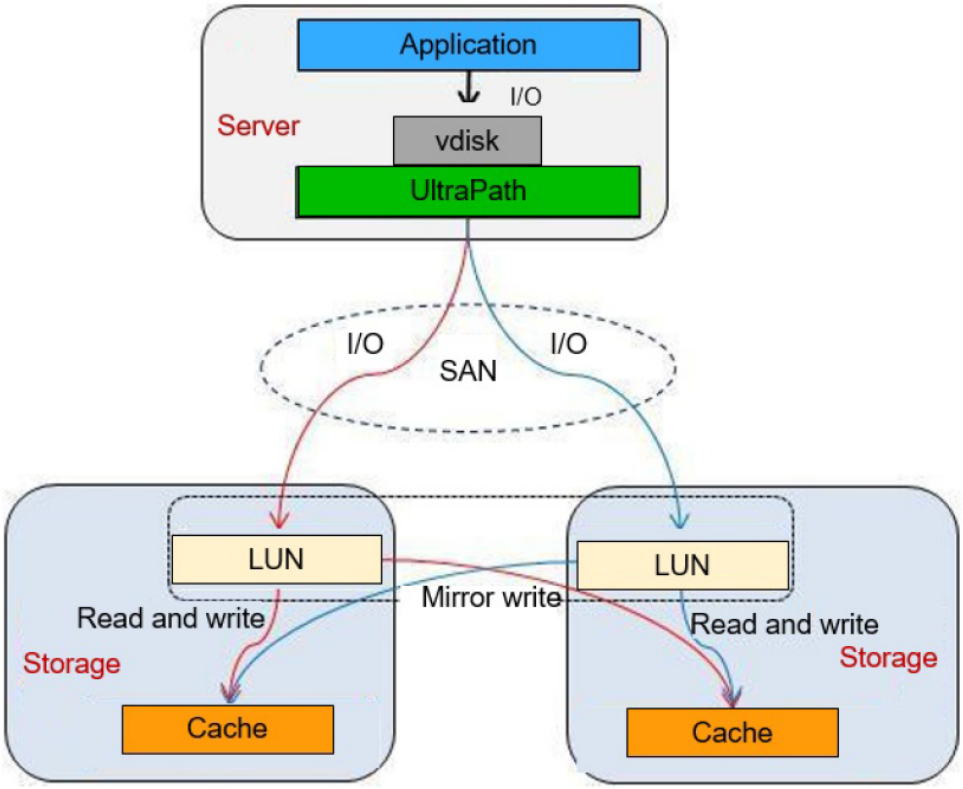


Figura 3.4: Caminhos de I/O ativo-ativo

3.3.1 Topologia e Modo de Operação

A topologia implementada segue o modelo *single-DC deployment*, no qual ambos os sistemas de armazenamento se encontram localizados no mesmo centro de dados físico e interligados através de uma malha de comunicação redundante. Esta configuração minimiza a latência entre controladores e maximiza o desempenho, preservando simultaneamente as propriedades de tolerância a falhas de uma arquitetura ativo-ativo completa. Importa notar que, embora a implementação descrita adote um *single-DC deployment*, o comportamento operacional não se altera em *dual-DC deployment* desde que sejam cumpridas as metas de latência de rede. Em particular, para serviços de base de dados recomenda-se *Round Trip Time* (RTT) ≤ 1 ms com distâncias tipicamente inferiores a 100 km, enquanto para plataformas de virtualização (por exemplo, vSphere) admite-se RTT < 10 ms com distâncias até aproximadamente 300 km, sem impacto significativo no funcionamento normal do *active-active*. (Huawei Technologies Co., Ltd., 2024)[2]

O ambiente opera em modo *cluster uniform*, o que significa que todos os hosts acedem simultaneamente aos dois sistemas de armazenamento através de caminhos de acesso equivalentes. Nesta configuração, cada volume lógico é apresentado de forma simétrica, sendo acessível em paralelo a partir de ambos equipamentos. Esta simetria garante a distribuição equilibrada das operações de I/O, evitando a dependência de caminhos preferenciais e assegurando latências homogéneas entre controladores, independentemente da origem da operação.

O controlo de quorum foi implementado em modo *dual quorum server*, no qual dois servidores externos monitorizam o estado de cada sistema de armazenamento e executam a arbitragem automática em caso de falha de comunicação. Este mecanismo garante que apenas o nó com quorum ativo mantém acesso de escrita aos volumes partilhados, prevenindo a ocorrência de situações de *split-brain* e assegurando a integridade dos dados. A decisão é tomada com base no princípio de maioria, evitando ambiguidades de autoridade entre controladores.

Nos sistemas ativo-ativo, as operações de leitura e escrita são processadas de modo coordenado entre os controladores de ambos os nós. Durante as operações de leitura, o acesso é efetuado localmente pelo nó ao qual o host se encontra ligado. Este comportamento reduz a latência, uma vez que os dados são servidos diretamente a partir do armazenamento local, sem necessidade de encaminhamento para o nó remoto. Em caso de falha ou indisponibilidade temporária do controlador local, o caminho de I/O é automaticamente redirecionado para o controlador remoto, assegurando continuidade de serviço.

As operações de escrita são executadas de forma síncrona. Quando um host emite um pedido de escrita, os dados são transmitidos em simultâneo para ambos os controladores. A operação apenas é confirmada ao host após ambas as escritas terem sido concluídas com sucesso, garantindo consistência total entre os nós. Este mecanismo elimina a possibilidade de divergência de dados e assegura um *Recovery*

[2]Requisitos de rede: *HyperMetro Feature Guide*, secção de planeamento de rede (latência/distância) e *quorum network*.

Point Objective (RPO) de zero.

Durante o processo de replicação, cada nó mantém registos de alterações designados *data change logs* (DCL). Caso ocorra uma falha num dos controladores, o nó ativo continua a aceitar operações de escrita e armazena as diferenças ocorridas. Após a recuperação do nó afetado, o sistema realiza automaticamente a sincronização diferencial com base nos DCL, restaurando a consistência sem interrupção do serviço. Este procedimento assegura um *Recovery Time Objective* (RTO) mínimo, uma vez que apenas os blocos alterados necessitam de ser retransmitidos.

A combinação entre o *single-DC deployment*, o *cluster uniform* e o *dual quorum server mode* proporciona um equilíbrio ótimo entre desempenho e fiabilidade. A arquitetura resultante é capaz de oferecer consistência transacional e disponibilidade contínua com latência reduzida, satisfazendo os requisitos de ambientes de missão crítica onde a integridade dos dados e a continuidade do serviço são essenciais.

3.3.2 Mecanismo de Arbitragem e Prevenção de Cisão de Sistema

Em situações em que a ligação entre os sistemas de armazenamento é interrompida, a replicação síncrona fica temporariamente comprometida. Para evitar riscos de corrupção ou inconsistência de dados, é necessário um mecanismo de arbitragem que determine de forma segura qual dos sistemas mantém o acesso de escrita. Esta decisão é baseada em fatores como o estado da comunicação, a disponibilidade dos nós e a integridade dos dados armazenados, assegurando que nunca existam escritas concorrentes em volumes replicados. A tecnologia ativo-ativo implementa este mecanismo de forma automática, prevenindo a ocorrência de situações de *split-brain* e garantindo a consistência dos dados mesmo durante falhas de rede ou de equipamentos.

O ambiente foi configurado segundo o modo *dual quorum server*, uma abordagem que utiliza dois servidores de quorum externos e independentes, responsáveis por arbitrar o estado do par. Estes servidores comunicam com ambos os sistemas de armazenamento através de endereços IP de gestão dedicados, interligados por uma rede de comunicação plana (*flat VLAN*) especificamente reservada para o serviço de quorum. Esta rede foi planeada garantindo redundância, isolamento de tráfego e tempos de resposta dentro das metas definidas (*Round Trip Time* (RTT) inferior a 50 ms para a rede de quorum). Esta topologia elimina pontos únicos de falha e assegura que, em caso de perda de comunicação entre os dois sistemas de armazenamento, a decisão sobre qual deles mantém acesso de escrita é sempre tomada por uma entidade externa e neutra.

A instalação e configuração dos servidores de quorum seguiram as diretrizes para ambientes de arbitragem baseados em servidores *third-party* com sistema operativo Ubuntu. O processo incluiu a instalação do *Quorum Server Software*, a configuração das interfaces de rede, a importação dos certificados de arbitragem e a definição das chaves de comunicação seguras entre os storages e os servidores de

quorum. Estes certificados são responsáveis por autenticar as ligações e garantir a integridade das comunicações entre os elementos do domínio de arbitragem. Cada servidor de quorum mantém sessões de controlo com ambos os storages (Figura 3.5), registando periodicamente *heartbeats* e atualizando o estado de cada sistema através de um mecanismo de *lease*, o qual é renovado ciclicamente para confirmar a disponibilidade de cada nó.

Após a configuração dos servidores de quorum, foi criado um domínio ativo-ativo de replicação síncrona, associando os dois sistemas de armazenamento e o par de quorum servers. A ligação entre os storages foi estabelecida através de uma rede Fibre Channel dedicada. A criação do domínio garante que as LUNs definidas para operação ativo-ativo são geridas de forma coordenada, permitindo que os processos de replicação, *heartbeat* e arbitragem ocorram de forma automática e transparente.

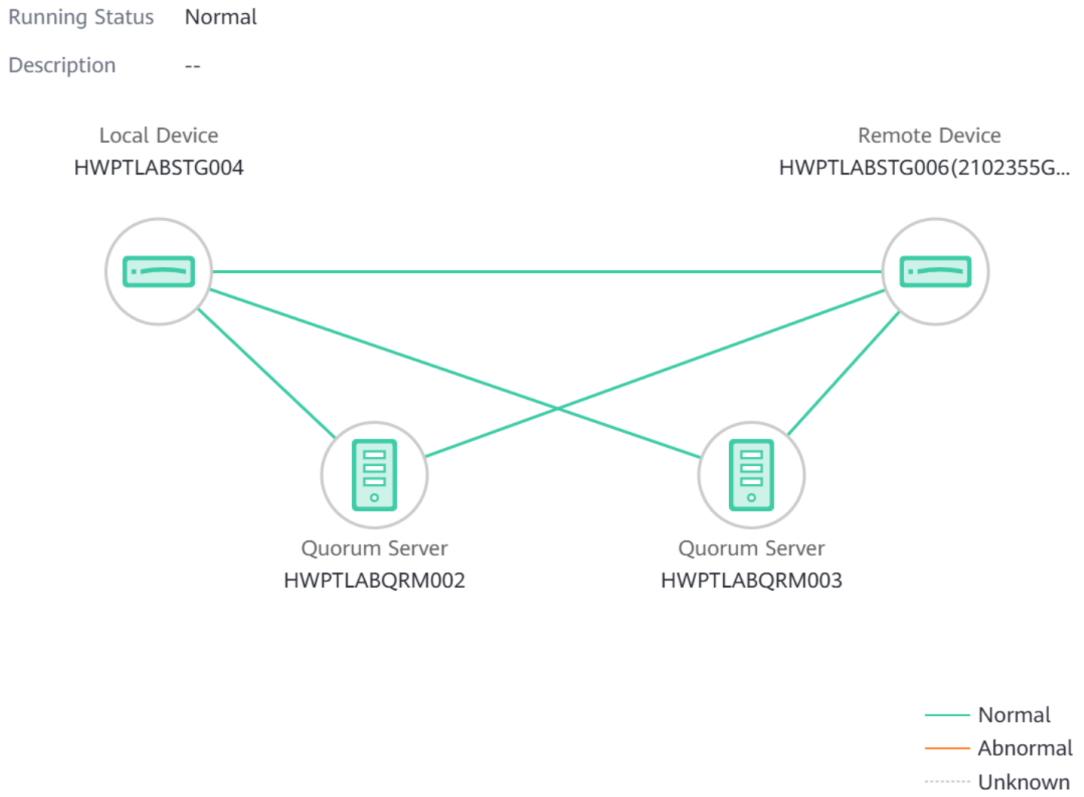


Figura 3.5: Domínio ativo-ativo e respetivos Quorum Servers

O processo de decisão durante falhas ocorre de forma determinística. Em operação normal, ambos os sistemas de armazenamento mantêm comunicações estáveis entre si e com os dois servidores de quorum, sendo as operações de leitura e escrita aceites em simultâneo. Caso ocorra uma falha de ligação entre os sistemas, o nó que mantiver conetividade com pelo menos um servidor de quorum permanece com acesso de escrita, enquanto o outro entra em modo de leitura até que o quorum seja restabelecido. Se um dos servidores de quorum se tornar indisponível, o segundo assume a função de arbitragem sem impacto no serviço. A tomada de posse do quorum em modo de espera ocorre tipicamente num intervalo inferior a um minuto. Este mecanismo assegura que a continuidade de serviço é mantida

e que apenas um lado permanece autorizado a efetuar escritas, preservando a consistência global dos dados.

A Tabela 3.1 apresenta uma síntese dos principais cenários de falha e da respetiva decisão de acesso a volumes replicados.

Tabela 3.1: Comportamento do sistema em diferentes cenários de falha no modo *dual quorum server*

Cenário de Falha	**Condição de Rede**	**Nó com Acesso de Escrita**	**Ação do Sistema**
Falha de ligação inter-site	Quorum alcançável apenas por um nó	Nó com quorum ativo	Mantém escrita; outro passa para modo leitura
Falha de um servidor de quorum	Segundo quorum operacional	Ambos os nós	Operação normal; arbitragem mantida
Falha de ambos os servidores de quorum	Sem quorum disponível	Nó com prioridade estática	Mantém escrita conforme política de prioridade
Falha total de um nó de armazenamento	Outro nó + quorum disponíveis	Nó saudável	Mantém escrita; sincronização retomada após recuperação

Com esta configuração, a solução implementada assegura uma arbitragem fiável, determinística e isenta de intervenção manual. O modo *dual quorum server* elimina a dependência de um único ponto de decisão, reduz o risco de *split-brain* e permite uma recuperação rápida e previsível, garantindo $RPO = 0$ e RTO mínimo. Todos os procedimentos e parâmetros seguidos basearam-se na documentação técnica *OceanStor Dorado V6 Series HyperMetro Feature Guide for Block* (Huawei Technologies Co., Ltd., 2024), respeitando as recomendações de planeamento, rede e segurança para implementações ativo-ativo de classe empresarial.

Em situações onde a ligação entre os sistemas de armazenamento é interrompida, a replicação em tempo real fica comprometida. Para evitar riscos de corrupção ou inconsistência de dados, um mecanismo de arbitragem decide qual dos sistemas continua a fornecer os serviços. Esta decisão é tomada com base em fatores como estado da comunicação, número de nós ativos e integridade dos dados armazenados.

A tecnologia utilizada suporta diversos métodos de conectividade entre os sistemas, incluindo ligações diretas, redes Fibre Channel e redes IP, garantindo flexibilidade na implementação conforme as infraestruturas existentes.

3.4 Metodologia Experimental

A metodologia experimental foi concebida para comparar, em condições controladas, o comportamento das duas arquiteturas de armazenamento estudadas, uma baseada num único equipamento e outra numa configuração ativo-ativo, sob cargas representativas de workloads transacionais. A execução dos ensaios iniciou-se com uma fase de validação preliminar da infraestrutura, na qual foram realizadas iterações destinadas a verificar a tolerância a falhas e o correto funcionamento do modelo ativo-ativo. Apenas após essa verificação foram conduzidos os testes formais de desempenho, concebidos para permitir uma comparação direta entre as duas arquiteturas, aplicando a mesma matriz de ensaios em ambos os casos.

Nos testes comparativos, todas as execuções foram realizadas com um único host de geração de carga (vm01), de modo a eliminar o impacto da concorrência multi-host sobre as métricas de latência e *Input/Output Operations Per Second* (IOPS). Cada host acedeu a dois dispositivos de bloco independentes (`/dev/sdb` e `/dev/sdc`) configurados em modo *raw* e abertos com a opção `o_direct`, garantindo que as medições refletiam o comportamento real do subsistema de armazenamento sem influência de *page caching* do sistema operativo.

O perfil de carga adotado consistiu em operações de I/O aleatório com uma relação de 70% de leituras e 30% de escritas, distribuídas uniformemente sobre ambos os dispositivos. Esta composição procura aproximar-se de *workloads* transacionais típicos, em que as operações de leitura representam a maioria do volume de I/O, mas as operações de escrita mantêm um peso relevante. A adoção deste rácio de 70/30 segue a prática habitual em testes de infraestruturas OLTP, em que se procura emular o comportamento de bases de dados transacionais com predominância de operações de leitura sobre conjuntos de dados de dimensão apreciável (Raghunatha & Ramappa, 2009). A utilização deste perfil de I/O assegura, assim, uma aproximação razoável ao comportamento esperado de aplicações empresariais com acesso concorrente a dados estruturados.

Cada ensaio teve duração total de quinze minutos, incluindo um período inicial de estabilização de dois minutos, durante o qual o sistema atingiu o estado estacionário. As métricas foram recolhidas a intervalos de um segundo, permitindo uma análise temporal de elevada resolução.

O paralelismo foi mantido constante, com sessenta e quatro *threads* de I/O por host, assegurando operação em regime de saturação controlada. O parâmetro `iorate=max` foi utilizado para eliminar limitações artificiais e permitir observar o comportamento natural do sistema até ao ponto de máxima eficiência.

A matriz experimental contemplou cinco tamanhos de bloco (4 KiB, 8 KiB, 16 KiB, 32 KiB e 64 KiB) aplicados de forma idêntica em ambas as arquiteturas. A seleção destes tamanhos segue a prática comum em estudos de caracterização de desempenho de subsistemas de armazenamento e de plataformas de virtualização, nos quais blocos em potências de dois nesta gama representam as transferências de dados pequenas e médias mais frequentes em sistemas de ficheiros, bases de dados e ambientes virtualizados. Em particular, tamanhos de bloco entre 4 KiB e 16 KiB

são típicos de páginas de bases de dados e de unidades de alocação de sistemas de ficheiros genéricos, enquanto blocos de 32 KiB e 64 KiB são frequentemente utilizados para representar acessos de maior dimensão e maior débito sustentado. Esta variação permitiu caracterizar a transição entre regimes limitados por IOPS e regimes limitados por largura de banda, evidenciando a escalabilidade e a estabilidade de desempenho das soluções avaliadas.

```
# VDBENCH Parameter File
# Name: teste_aa_4k_70-30random.vdb

hd=vm01,user=root,vdbench=/usr/src/vdbench/

sd=sd1,host=*,lun=/dev/sdb,openflags=o_direct
sd=sd2,host=*,lun=/dev/sdc,openflags=o_direct

wd=wd1,sd=sd*,xfersize=4k,rdpct=70,seekpct=100

rd=run1,wd=wd1,elapsed=900,interval=1,iorate=max,warmup=120,threads=64
```

Listing 3.1: Excerto de configuração de um teste VDBench 4K 70/30 aleatório

As métricas consideradas incluíram a taxa efetiva de IOPS, a latência média e os percentis p50, p95 e p99, bem como o *throughput* agregado em MB/s, calculados a partir dos registos de desempenho produzidos pelo VDBench. A utilização de percentis de latência permite caracterizar não apenas o comportamento médio, mas também a variabilidade e o impacto dos valores de latência mais elevados: o p50 aproxima-se da experiência típica da maioria das operações, enquanto os percentis p95 e p99 evidenciam o comportamento das operações mais lentas, particularmente relevantes em ambientes sensíveis à latência. Apenas os dados recolhidos após o período de *warm-up* foram analisados, garantindo que os resultados correspondiam ao comportamento em regime estacionário. A observação das séries temporais permitiu ainda identificar variações transitórias associadas a processos internos de *garbage collection*, isto é, operações em que o sistema reorganiza e liberta blocos de memória ou de flash para reutilização, e a políticas de *destage de cache*, em que dados temporariamente mantidos em memória rápida são gravados de forma definitiva nos discos ou dispositivos flash subjacentes, comportamentos típicos em cargas mistas de leitura e escrita.

O controlo rigoroso das variáveis experimentais e a aplicação da mesma matriz de ensaios às duas arquiteturas asseguraram a comparabilidade dos resultados obtidos, permitindo que as diferenças observadas em latência, throughput ou estabilidade temporal fossem atribuídas ao modo de operação do armazenamento e não a fatores externos ao ambiente de teste.

3.5 Proteção contra Ransomware

A conceção inicial da estratégia de proteção contra ransomware previa a utilização de um equipamento dedicado de proteção de dados que introduziria uma camada adicional de isolamento de dados, frequentemente designada por *air-gap*. Esta camada permitiria a replicação periódica de cópias de dados para um domínio logicamente isolado, apenas acessível durante as janelas de sincronização planeadas. O objetivo era criar uma barreira física e lógica entre os dados de produção e as cópias de segurança, mitigando a possibilidade de propagação de software malicioso ou de encriptação em cadeia.

A arquitetura proposta incluía ainda uma componente de deteção comportamental baseada em análise de metadados e variações de conteúdo, suportada por algoritmos de aprendizagem automática, capaz de identificar padrões de encriptação maliciosa e atividades anómalas nos volumes de armazenamento. Esta abordagem permitiria uma resposta proativa, combinando deteção antecipada de ataques com isolamento automático das cópias protegidas.

No entanto, o equipamento previsto para a implementação desta camada de isolamento físico ficou indisponível, e a aquisição de um novo sistema dentro do prazo do projeto revelou-se inviável. Perante esta limitação, foi necessário adaptar a solução, mantendo o foco na imutabilidade e na recuperação fiável dos dados. Assim, foi implementado um mecanismo de *Secure Snapshots* do tipo WORM (*Write-Once, Read-Many*), que assegura a proteção lógica contra modificações indevidas ou encriptação dos dados armazenados. Ao contrário dos *snapshots* convencionais, que podem ser eliminados ou alterados antecipadamente por um utilizador com privilégios administrativos, os *Secure Snapshots* permanecem imutáveis durante um período de retenção previamente definido, impedindo a sua alteração ou remoção e garantindo a existência de pontos de recuperação consistentes mesmo em cenário de compromisso das credenciais de administração.

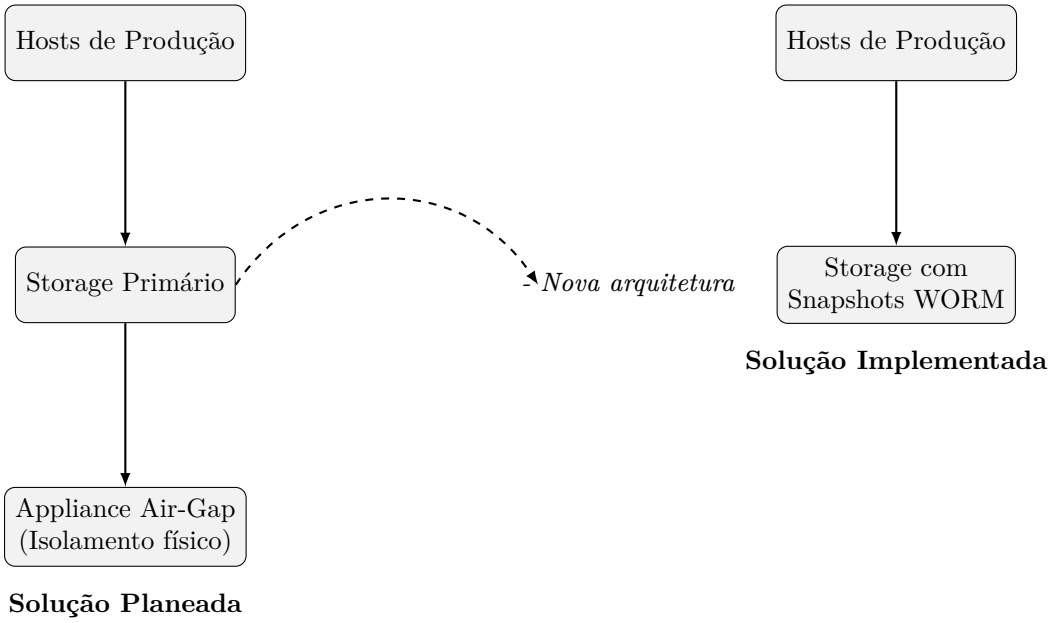


Figura 3.6: Evolução da arquitetura de proteção contra ransomware: da solução *air-gap* planeada à solução com *Secure Snapshots WORM*.

A estratégia de proteção adotada assenta, portanto, na criação periódica de *snapshots* imutáveis, cuja principal característica é a impossibilidade de alteração ou eliminação durante o período de retenção configurado. Esta abordagem impede que software malicioso ou ações administrativas indevidas comprometam as cópias de restauro, garantindo a disponibilidade de pontos de recuperação consistentes e verificáveis.

Do ponto de vista funcional, cada *snapshot* WORM representa um ponto no tempo obtido por mecanismos de redirecionamento de escrita a nível de bloco, preservando o estado lógico dos volumes de produção no instante da sua criação. Durante o período de retenção, o *snapshot* permanece protegido contra modificações, sendo a expiração controlada pelo sistema de acordo com a política definida. A criação é efetuada de forma não intrusiva para as aplicações, e a retenção é escalonada temporalmente para equilibrar granularidade e consumo de capacidade.

O processo de recuperação consiste na seleção de um *snapshot* anterior ao incidente e na apresentação controlada desse ponto aos *hosts* de produção, permitindo restaurar o serviço a um estado consistente. Quando tecnicamente suportado, a operação de restauro do volume a partir de um *snapshot* permite repor diretamente o estado anterior sem necessidade de cópia integral dos dados. Em alternativa, a recuperação pode ser realizada através da clonagem do *snapshot* para um novo volume, preservando o original para referência e eventual análise.

A adoção de *snapshots* WORM complementa as capacidades de alta disponibilidade descritas nas secções anteriores, fornecendo uma camada adicional de resiliência lógica orientada à integridade dos dados. Enquanto a arquitetura ativo-ativo assegura continuidade de serviço perante falhas físicas, a imutabilidade dos *snapshots* garante proteção contra corrupção lógica e ataques de encriptação, redu-

zindo significativamente o risco operacional e o tempo necessário para restabelecer o ambiente produtivo a um estado limpo e validado.

Em síntese, a substituição da abordagem *air-gap* planeada pelos *Secure Snapshots* WORM revelou-se eficaz no cumprimento dos objetivos de proteção e recuperação, permitindo validar no ambiente experimental a criação automática de cópias imutáveis, o bloqueio de modificações durante a retenção e a restauração operacional sem impacto nos *snapshots* remanescentes. Esta solução, embora logicamente distinta da proposta inicial, mantém a robustez e os princípios de defesa em profundidade originalmente previstos no projeto.

Capítulo 4

Resultados e Análise

Este capítulo apresenta e analisa os resultados obtidos durante a fase experimental do estudo, avaliando o comportamento das arquiteturas de armazenamento centralizado configuradas em modo *single* e em modo ativo-ativo. Os ensaios realizados permitiram caracterizar o desempenho, a resiliência e a capacidade de recuperação das soluções implementadas, bem como validar a eficácia dos mecanismos de automação desenvolvidos para monitorização e manutenção do sistema. A análise baseia-se em métricas objetivas recolhidas através de testes de carga controlados e reprodutíveis, complementadas por observações decorrentes dos testes de tolerância a falhas e dos procedimentos de verificação automatizada.

A estrutura do capítulo encontra-se organizada em três secções principais, resultados de desempenho e comparação entre as arquiteturas, testes de redundância e recuperação, automação e verificação do estado dos equipamentos.

4.1 Arquitetura *Single*

A primeira fase dos ensaios teve como objetivo avaliar o desempenho da arquitetura single, representando uma configuração convencional de armazenamento centralizado baseada num único sistema de serviço. Esta abordagem foi utilizada como referência de base para a análise comparativa posterior, permitindo quantificar o impacto da introdução de redundância total na arquitetura ativo-ativo. Todos os testes foram conduzidos garantindo a reprodutibilidade e a neutralidade das medições.

Os resultados obtidos para a arquitetura *single* evidenciam um comportamento estável e previsível em todas as execuções, sem variações significativas entre as repetições de teste. As métricas apresentadas nos gráficos do repositório de resultados indicam que, para tamanhos de bloco de 4 KiB, o sistema alcançou a maior taxa de operações por segundo, situando-se no regime dominado por I/O de pequena granularidade. À medida que o tamanho de bloco aumentou (8 KiB, 16 KiB, 32 KiB e 64 KiB), observou-se uma diminuição gradual no número de IOPS, acompanhada por um aumento proporcional do *throughput* agregado, o que confirma

a transição para um regime limitado por largura de banda (Figuras 4.1 e 4.2). Este comportamento é típico de infraestruturas de armazenamento empresarial, nas quais blocos pequenos favorecem o paralelismo e blocos maiores privilegiam o débito total de transferência.

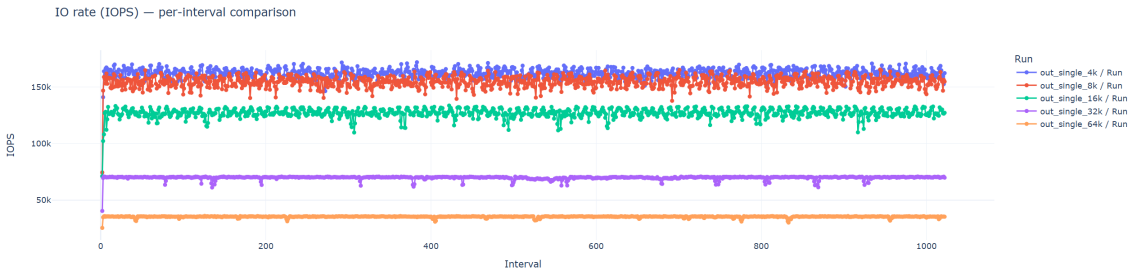


Figura 4.1: Evolução do número de IOPS em função do tamanho de bloco para a arquitetura single.

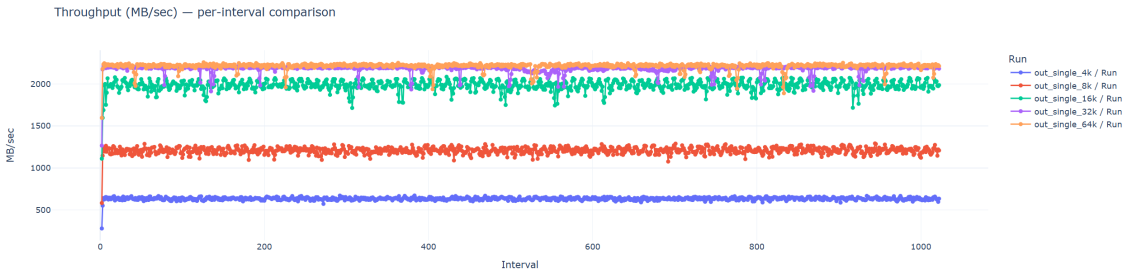


Figura 4.2: Evolução do throughput agregado em função do tamanho de bloco para a arquitetura single.

A análise de latências baseou-se nos gráficos de latência média e de percentis apresentados nas Figuras B1 a B10, no Apêndice B, onde se representa a evolução temporal das respostas para os diferentes tamanhos de bloco e perfis de carga. Esses resultados demonstram valores médios crescentes à medida que o tamanho de bloco aumenta, evidenciando o impacto previsível da quantidade de dados processados por operação. Os gráficos de latência média e de percentis confirmam a consistência temporal das respostas, sem ocorrência de picos anómalos associados a processos de *cache* ou mecanismos internos de *destage*. A distribuição das latências, avaliada através dos percentis p50, p95 e p99, revelou uma diferença marginal entre os valores médios e os extremos superiores, indicando um comportamento estável e previsível ao longo de todo o ensaio. A proximidade entre os percentis demonstra que a maior parte das operações de I/O foram concluídas dentro de tempos de resposta consistentes, sem variações significativas nos 5% ou 1% de operações mais lentas, o que confirma a ausência de degradação transitória e a fiabilidade do subsistema de armazenamento. As medições de CPU revelam níveis de utilização inferiores a 60% em todos os testes, indicando que o sistema não atingiu saturação e que o desempenho observado foi limitado pelo subsistema de armazenamento e não pela camada de computação.

Em síntese, os resultados obtidos demonstram que a arquitetura single proporciona um desempenho estável e linear sob cargas controladas, com boa previ-

sibilidade e sem degradações súbitas de resposta. Contudo, a ausência de redundância estrutural torna esta solução vulnerável a falhas do equipamento principal, conduzindo à interrupção total do serviço em caso de avaria. Esta configuração estabelece, portanto, a linha de base necessária para a avaliação subsequente da arquitetura ativo-ativo, cuja análise visa determinar em que medida a replicação síncrona e a distribuição de carga podem aumentar a disponibilidade e a resiliência sem comprometer o desempenho global do sistema.

4.2 Arquitetura Ativo-Ativo

A avaliação da arquitetura ativo-ativo teve como objetivo analisar o impacto do paralelismo e da replicação síncrona no desempenho global do sistema de armazenamento. Os testes foram realizados sob as mesmas condições experimentais aplicadas à arquitetura single, garantindo a comparabilidade direta entre ambas soluções. Foram testados tamanhos de bloco de 4 KiB, 8 KiB, 16 KiB, 32 KiB e 64 KiB, recorrendo a um perfil misto de 70% de operações de leitura e 30% de escrita. A duração total de cada ensaio foi de quinze minutos, com um período inicial de *warm-up* de dois minutos para estabilização das métricas.

Os resultados obtidos revelam um comportamento globalmente estável em todos os tamanhos de bloco testados, com pequenas oscilações associadas a variações transitórias de carga. A tendência geral confirma a relação inversa entre o número de IOPS e o tamanho de bloco, acompanhada por um aumento quase linear do *throughput* até ao limite imposto pela largura de banda do sistema. Este padrão é visível nos gráficos de desempenho apresentados nas Figuras 4.3 e 4.4, que evidenciam a diminuição progressiva das operações por segundo e o aumento sustentado do débito agregado até à estabilização para blocos de 32 KiB e 64 KiB.

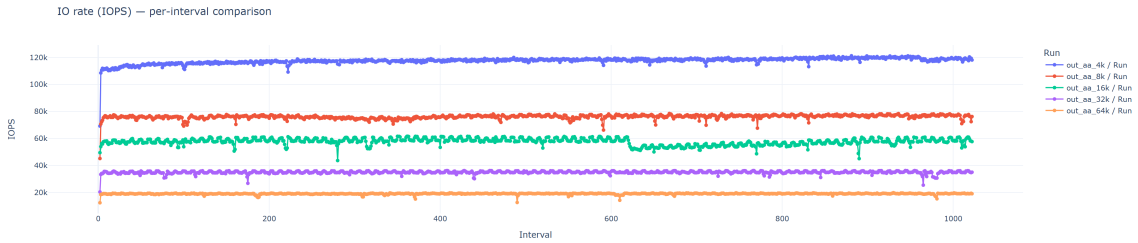


Figura 4.3: Evolução do número de IOPS em função do tamanho de bloco para a arquitetura ativo-ativo.

Para blocos de 4 KiB, a arquitetura ativo-ativo atingiu aproximadamente 120000 IOPS, com latência média inferior a 1,2 ms e variação mínima entre os percentis p50, p95 e p99, indicando elevada consistência temporal nas respostas. À medida que o tamanho de bloco aumentou, o número de IOPS reduziu-se para cerca de 75000 em 8 KiB, 60000 em 16 KiB, 35000 em 32 KiB e 19000 em 64 KiB. Em contrapartida, o *throughput* evoluiu de aproximadamente 450 MB/s para 1,2 GB/s, estabilizando nos blocos de maior dimensão. Esta relação confirma a transição do regime limitado por operações para um regime limitado por

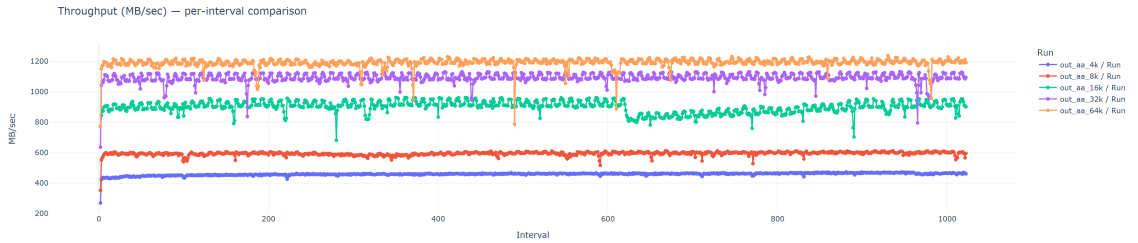


Figura 4.4: Evolução do throughput agregado em função do tamanho de bloco para a arquitetura ativo-ativo.

largura de banda, característico de sistemas de armazenamento otimizados para carga paralela.

A análise de latências demonstrou um crescimento gradual e previsível à medida que o tamanho de bloco aumentou: valores médios próximos de 1,1 ms em 4 KiB, 1,8 ms em 8 KiB, 2,2 ms em 16 KiB, 3,6 ms em 32 KiB e 6,5 ms em 64 KiB. Apesar do aumento progressivo, as diferenças entre os percentis p50, p95 e p99 mantiveram-se reduzidas em todas as execuções, o que demonstra estabilidade e ausência de eventos de latência anómala.

A utilização média de CPU manteve-se em níveis reduzidos e decrescentes com o aumento do tamanho de bloco, variando de cerca de 25 % em 4 KiB para menos de 12 % em 64 KiB. Este comportamento confirma que o subsistema de processamento não constitui fator limitativo e que o desempenho observado é dominado pelas operações de I/O e pela largura de banda disponível. A estabilidade das métricas de CPU e de *throughput* ao longo do tempo reforça ainda a eficiência da distribuição de caminhos de I/O entre controladores e a ausência de desequilíbrios significativos de carga.

Por fim, a percentagem de operações de leitura manteve-se próxima de 70 % em todos os testes, com variações inferiores a 1 %, o que comprova a fidelidade do perfil 70/30 definido na metodologia e garante a validade comparativa dos resultados. No conjunto, a arquitetura ativo-ativo demonstrou um desempenho consistente, previsível e escalável, atingindo níveis elevados de *throughput* sem evidência de saturação ou degradação de latência significativa, o que confirma a sua adequação a ambientes que requerem simultaneamente redundância e elevada disponibilidade operacional.

4.3 Comparação Single vs Ativo-Ativo

Esta secção apresenta a análise comparativa entre as arquiteturas single e ativo-ativo, com base nos resultados experimentais obtidos para os diferentes tamanhos de bloco, mantendo inalteradas as condições metodológicas descritas. O objetivo é caracterizar, de forma técnica e quantitativa, o impacto das diferenças arquiteturais no desempenho global do sistema de armazenamento, considerando métricas de IOPS, throughput, latência e utilização de CPU.

Nos testes realizados, a arquitetura single apresentou, de forma consistente, um número superior de IOPS em todos os tamanhos de bloco. Esta diferença é particularmente visível nos testes com blocos de 4 KiB e 8 KiB, em que o desempenho da solução single é cerca de duas vezes superior ao da arquitetura ativo-ativo. A Figura 4.5 ilustra este comportamento, mostrando a estabilidade das séries temporais e a menor taxa de operações registada na ativo-ativo. À medida que o tamanho de bloco aumenta, observa-se uma redução do número de operações por segundo em ambas as soluções (Figura 4.6), mas a relação relativa entre elas mantém-se praticamente constante.

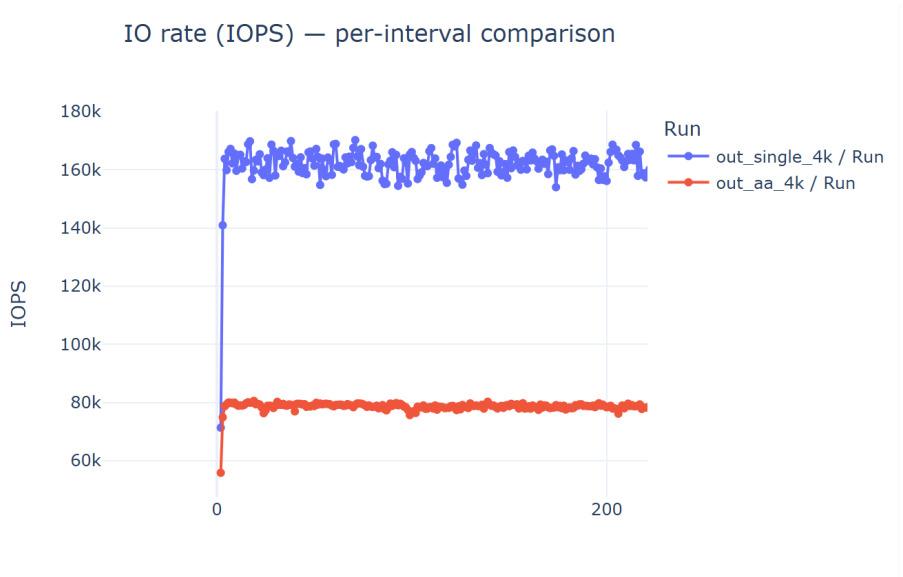


Figura 4.5: Comparação de IOPS (4 KiB) entre as arquiteturas single e ativo-ativo (séries por intervalo).

Figura 4.6: Comparação de IOPS (64 KiB) entre as arquiteturas single e ativo-ativo (séries por intervalo).

O comportamento do *throughput* segue uma tendência inversa à dos IOPS. Em

ambos os casos, o valor de débito agregado cresce com o aumento do tamanho de bloco, atingindo a estabilização nos testes de 64 KiB. A Figura 4.7 demonstra que, apesar da arquitetura single continuar a apresentar valores ligeiramente superiores, a diferença entre as duas soluções se torna progressivamente menos significativa à medida que o volume de dados transferido por operação aumenta. Este comportamento confirma que, em blocos grandes, o desempenho global passa a ser limitado pela largura de banda disponível e não pela capacidade de processamento de I/O.

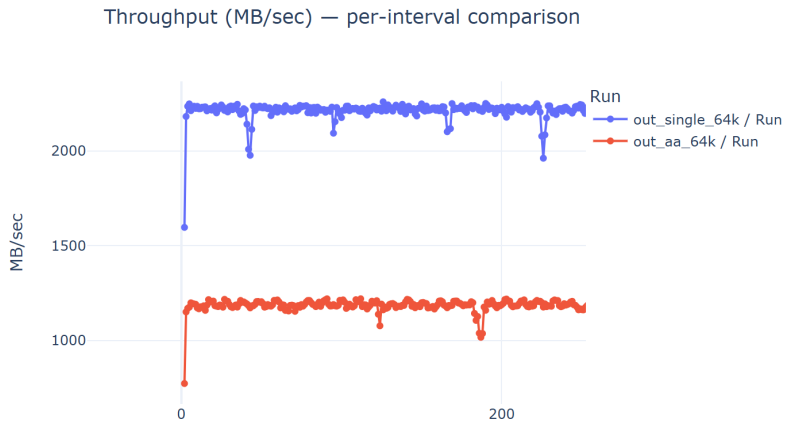


Figura 4.7: Comparação de *throughput* (64 KiB) entre as arquiteturas single e ativo-ativo (séries por intervalo).

No que respeita à latência, verificou-se um aumento gradual com o crescimento do tamanho de bloco em ambas as arquiteturas. A solução single manteve tempos de resposta inferiores em todos os ensaios, com valores médios próximos de 0,8 ms no teste de 4 KiB e de 3,5 ms no de 64 KiB. Na arquitetura ativo-ativo, a latência média foi consistentemente superior, situando-se entre 1,6 ms e 6,8 ms para os mesmos tamanhos de bloco. Esta diferença deve-se ao funcionamento intrínseco do modo ativo-ativo, que requer confirmação síncrona da operação em múltiplos controladores antes de concluir a transação, o que introduz uma pequena penalização temporal, visível nos gráficos da Figura 4.8.

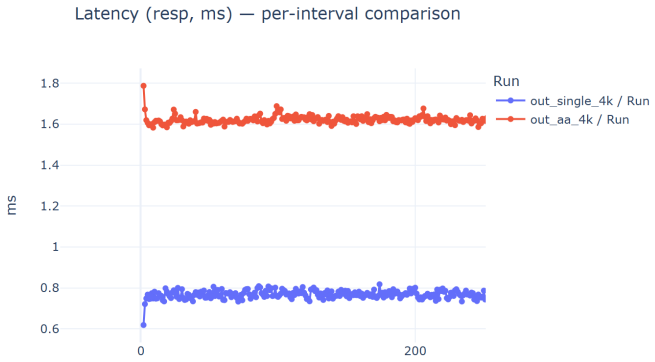


Figura 4.8: Comparação de latência de resposta (4 KiB) entre as arquiteturas single e ativo-ativo (séries por intervalo).

A análise temporal das métricas revela ainda que ambas as arquiteturas exibem um comportamento estável ao longo da duração dos testes, sem flutuações abruptas ou picos de latência significativos. No entanto, a arquitetura ativo-ativo demonstra uma dispersão ligeiramente superior nos percentis p95 e p99, o que indica um impacto marginal do mecanismo de replicação síncrona na consistência temporal das respostas (Figura 4.9). Apesar disso, os valores mantêm-se dentro de intervalos considerados aceitáveis para sistemas de armazenamento de elevado desempenho.

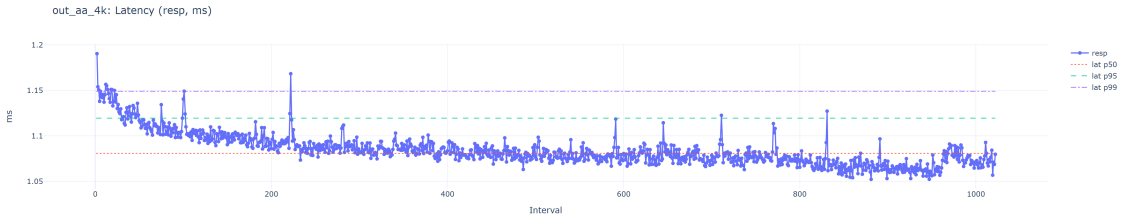


Figura 4.9: Comparação de latência de resposta (4 KiB) entre as arquiteturas single e ativo-ativo (séries por intervalo).

Relativamente à utilização de CPU, observou-se que a arquitetura single apresenta níveis de ocupação mais elevados, diretamente proporcionais à maior carga de operações processadas por segundo. Nos testes com blocos de 4 KiB, a utilização média de CPU atingiu cerca de 60 %, enquanto a arquitetura ativo-ativo se manteve em torno dos 30 %. Esta diferença reduz-se gradualmente à medida que o tamanho de bloco aumenta, aproximando-se dos 20 % e 10 % respetivamente nos testes com blocos de 64 KiB. O comportamento observado demonstra que o impacto da carga de I/O é mais pronunciado na arquitetura single, enquanto a ativo-ativo distribui de forma mais equilibrada o processamento associado às operações de leitura e escrita.

Em síntese, os resultados experimentais evidenciam que a arquitetura single privilegia o desempenho em termos de latência e número de operações por segundo, enquanto a ativo-ativo apresenta um perfil de desempenho mais estável, com menor consumo de CPU e ligeira penalização de latência decorrente do processo de replicação síncrona.

4.4 Testes de Redundância e Recuperação

As experiências de resiliência foram realizadas sob uma carga de trabalho controlada, com monitorização contínua dos parâmetros de desempenho tanto ao nível do sistema de armazenamento como da camada de virtualização, de forma a observar o comportamento do sistema perante falhas físicas e lógicas simuladas. O objetivo foi verificar a continuidade de serviço, a transição de caminhos I/O e a evolução temporal dos indicadores de desempenho durante e após cada perturbação. As experiências foram repetidas nas duas arquiteturas sob análise, variando apenas o tipo de indisponibilidade induzida.

A caracterização do estado do hardware nos equipamentos de armazenamento

foi efetuada com base na consola de gestão, que regista a remoção de módulos e de controladoras, bem como a perda de ligações Fibre Channel. A título ilustrativo, a remoção de um módulo de interfaces fica refletida de imediato no painel de alarmes com a indicação do componente afetado e da severidade do evento, acompanhado de recomendações do fabricante para recuperação (Figuras 4.10 e 4.11). Em paralelo, a vista de inventário mostra a ausência do módulo ou controladora correspondente, o que valida a correlação entre o evento físico e o estado lógico do equipamento.

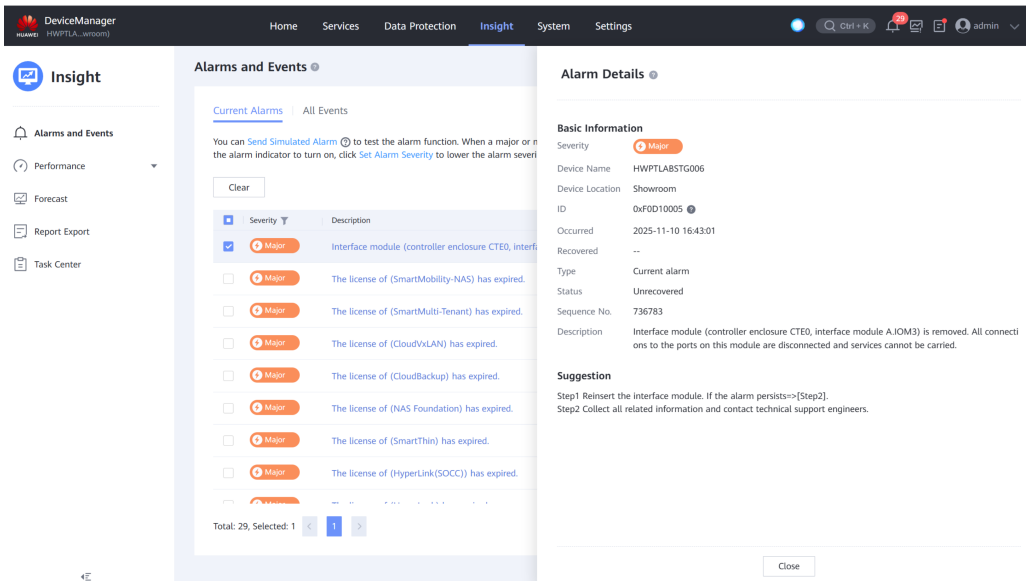


Figura 4.10: Alarme relativo à remoção de módulo de interface

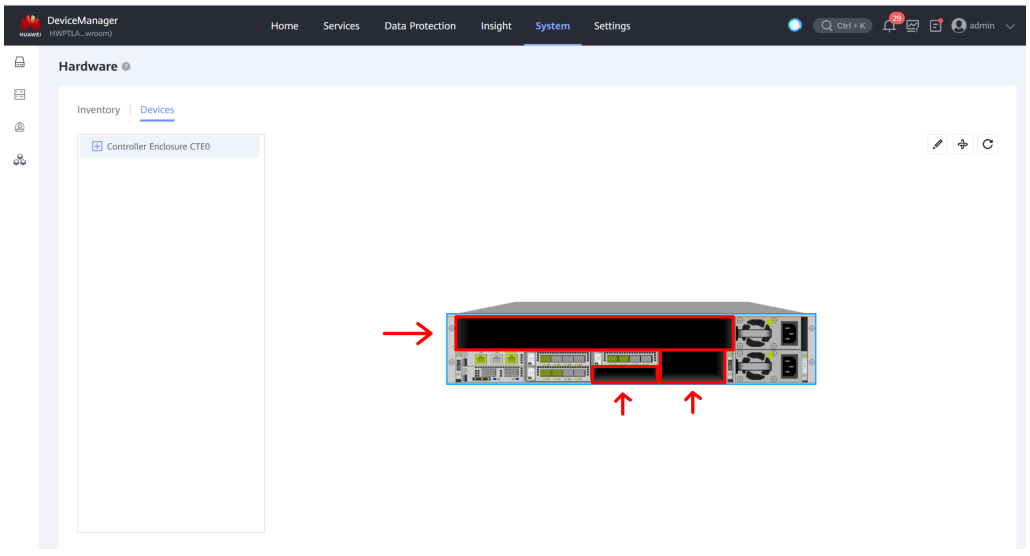


Figura 4.11: Indisponibilidade de uma das controladoras no sistema STG

No plano dos *hosts*, a observabilidade centrou-se no ecrã de *Storage Devices* do vSphere, com ênfase na aba *Paths*. Em operação normal, todos os caminhos configurados surgem como *Active (I/O)* ou *Active*, confirmando a presença de múltiplos percursos redundantes por fabric (Figura 4.12).

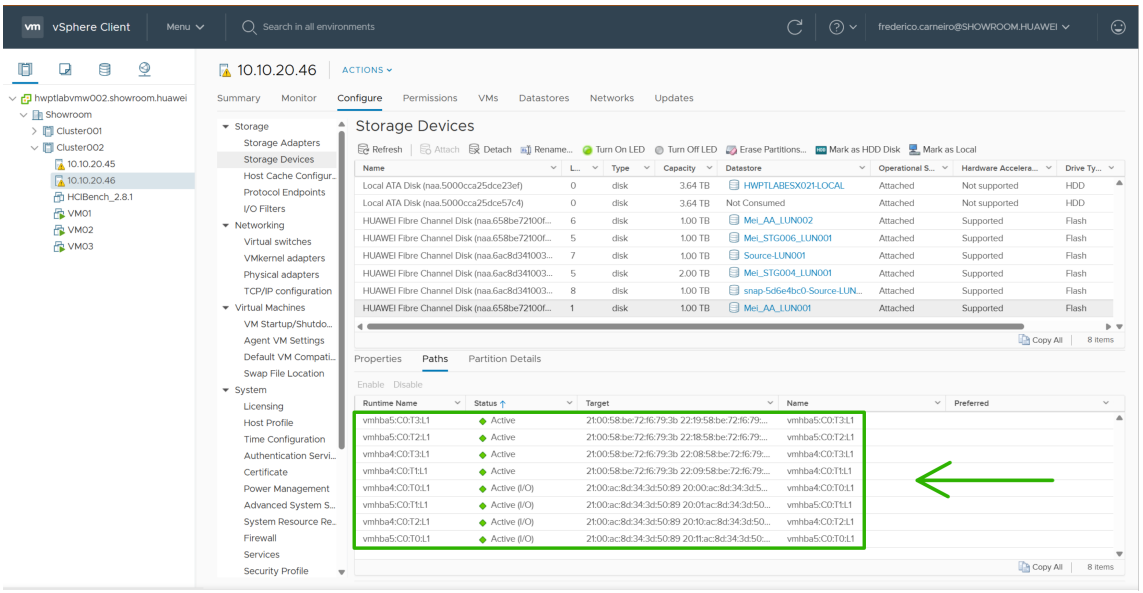


Figura 4.12: Estado normal dos caminhos de I/O no hipervisor VMware

Sempre que se provocou a perda parcial de um *fabric* SAN, desligando um conjunto de portas de um dos lados, os caminhos desse domínio transitaram para *Dead*, mantendo-se o tráfego pelos percursos pertencentes ao *fabric* saudável (Figuras 4.13 e 4.14). A recuperação do *fabric* restabeleceu automaticamente os caminhos para o estado *Active (I/O)*, sem necessidade de intervenção manual e sem perda de acessibilidade aos LUNs.

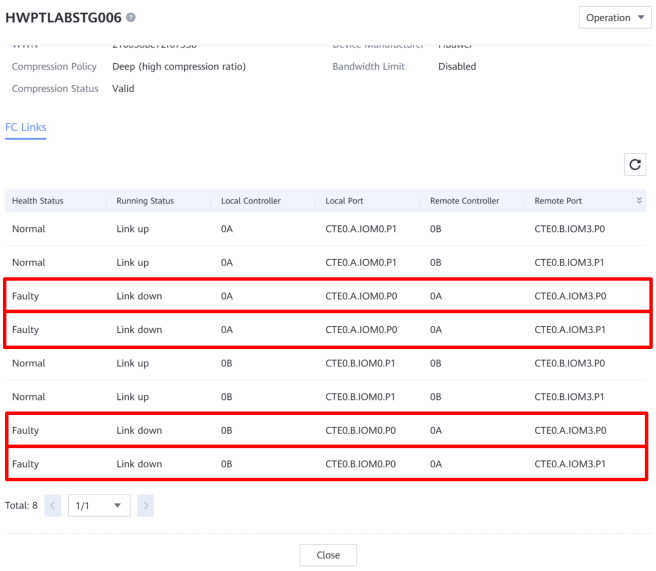


Figura 4.13: Perda parcial de ligações Fibre Channel num dos *fabrics* SAN

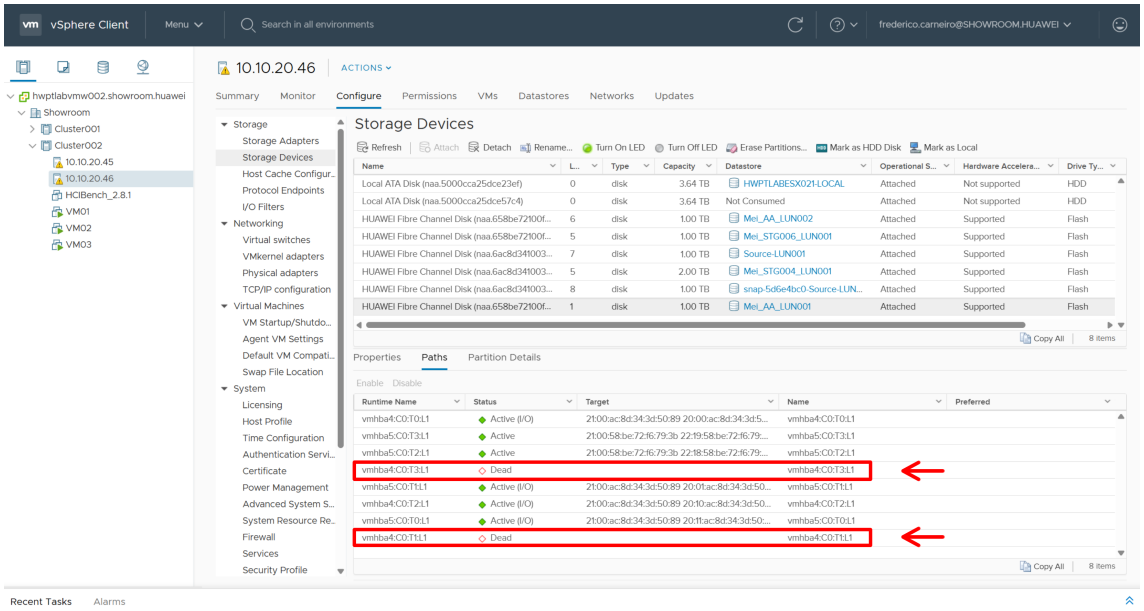
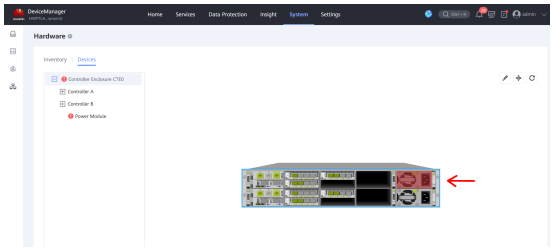
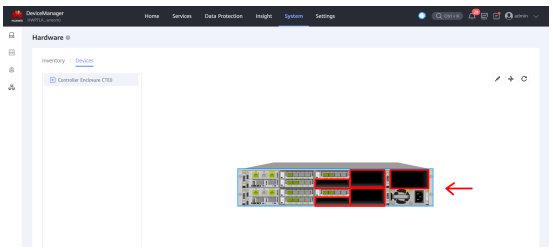


Figura 4.14: Comportamento dos caminhos de I/O após falha de um dos *fabrics*

Os testes de falha de alimentação incidiram sobre a extração de um cabo e a colocação em falha de uma fonte de alimentação, confirmando a operação do armazenamento com a fonte remanescente e o registo do alarme correspondente (Figuras 4.15a e 4.15b). De forma análoga, a remoção de um módulo de interface induziu a perda de ligações FC específicas, devidamente refletida na lista de *FC Links* com estados *Link down* para as portas afetadas e *Link up* nas restantes, preservando a conetividade por caminhos alternativos.



(a) Teste de falha de alimentação de uma das fontes



(b) Alarme gerado após falha de módulo de alimentação

Figura 4.15: Comportamento do sistema perante falha de alimentação

No cenário de degradação ao nível da controladora, foi efetuada a remoção de uma das controladoras do storage, deixando o subsistema a operar em modo unicamente com a controladora remanescente. A consola de gestão assinalou o evento como *Controller removed* com severidade *Major*, enquanto, do lado do ESXi, os caminhos mapeados para a controladora indisponível passaram a *Dead*, mantendo-se o I/O pelos caminhos ainda ativos (Figura 4.16). Durante todo o período de teste não se registou indisponibilidade do *datastore*, a coerência dos caminhos ativos foi suficiente para sustentar a carga.

Foi também ensaiado um evento de falha total de um dos storages na arqui-

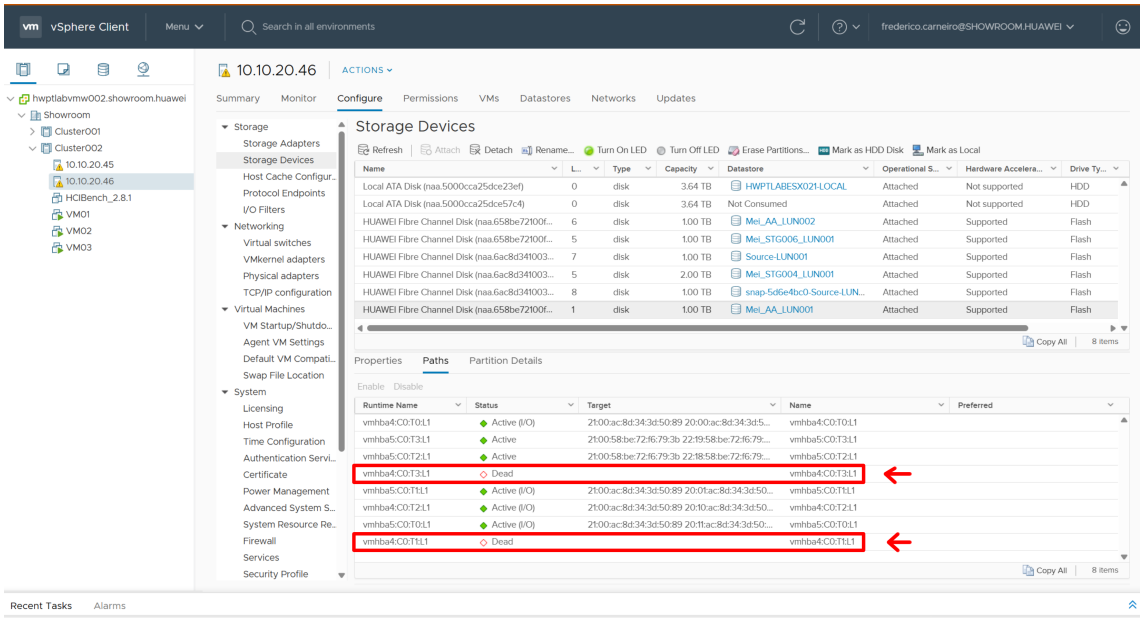


Figura 4.16: Remoção de uma controladora e manutenção de I/O ativo

tetura ativo-ativo. A indisponibilidade do equipamento STG B ocorreu aproximadamente ao segundo 253 do ensaio com bloco de 4 KB. A sequência temporal nos gráficos mostra, de forma consistente, uma degradação momentânea nos indicadores seguida de recuperação para o patamar pré-falha. No gráfico de IOPS observa-se um degrau descendente muito curto, com reestabilização rápida do débito (Figura 4.17). O throughput replica o mesmo padrão de queda breve e recomposição (Figura 4.18).

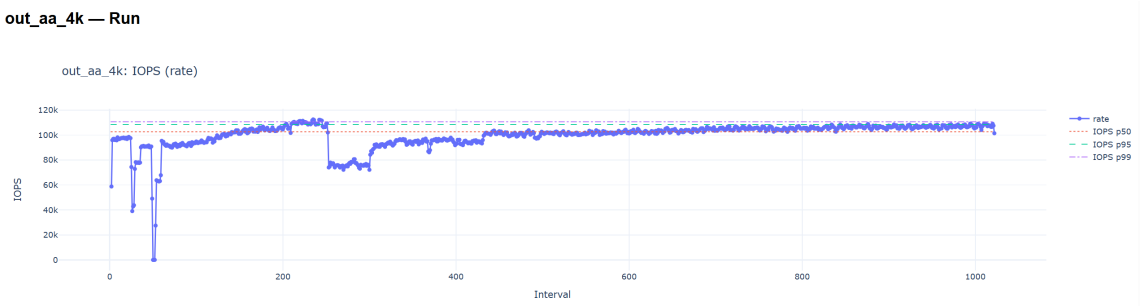


Figura 4.17: Evolução dos IOPS durante a falha total do storage STG B

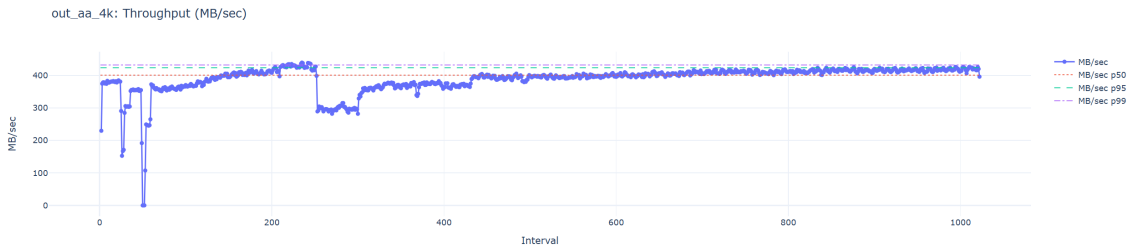


Figura 4.18: Evolução do throughput durante a falha total do storage STG B

A latência apresenta um pico transitório associado à convergência de caminhos e à reeleição de percursos pelo multipath, regressando de seguida aos valores anteriores (Figuras 4.19, 4.20 e 4.21).

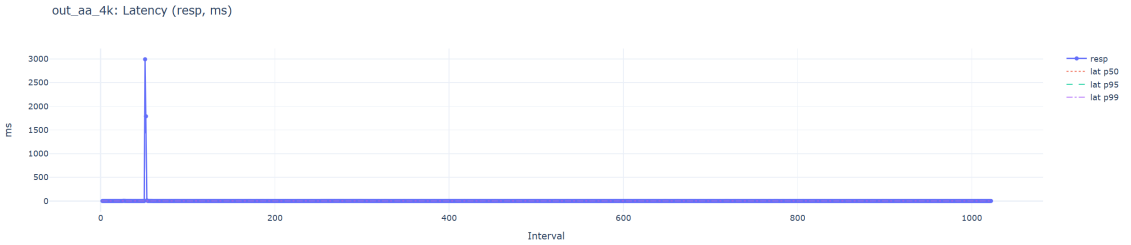


Figura 4.19: Latência agregada durante a falha total do storage STG B

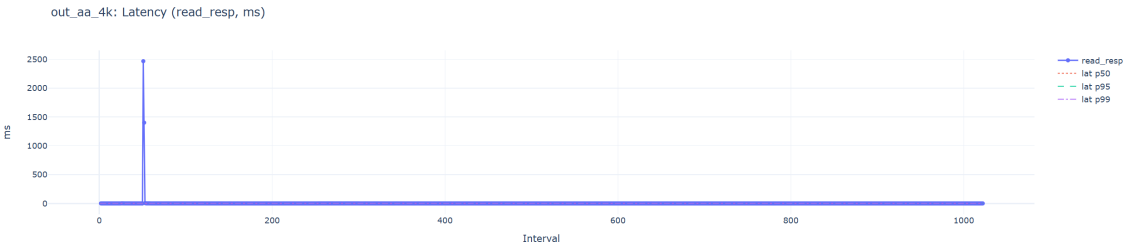


Figura 4.20: Latência de leitura durante a falha total do storage STG B

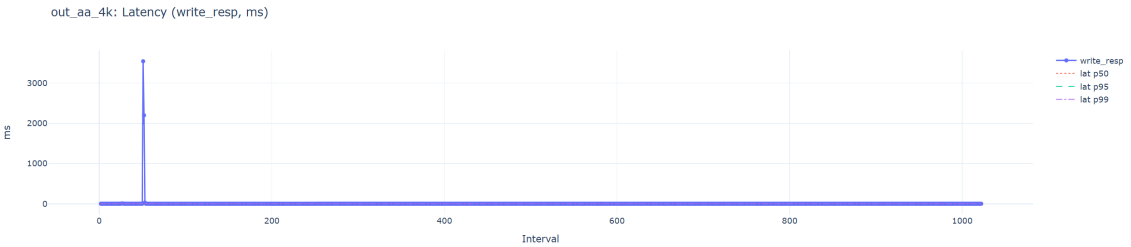


Figura 4.21: Latência de escrita durante a falha total do storage STG B

O perfil de I/O manteve-se estável, sem alteração do rácio de leitura (Figura 4.22), indicando manutenção do regime de trabalho da carga. Este comportamento confirma a capacidade da solução ativo-ativo de absorver a perda integral de um nó com impacto apenas transitório no desempenho e continuidade integral do serviço.

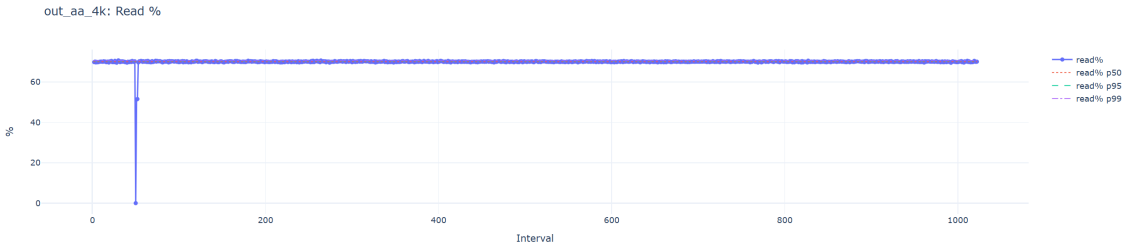


Figura 4.22: Percentagem de operações de leitura durante a falha total do storage STG B

Por contraste, na arquitetura single, a indisponibilidade integral do equipamento resultou na perda de acessibilidade ao conjunto de LUNs expostos por esse storage, com transição de todos os caminhos para *Dead* e quebra efetiva de serviço até à reposição do sistema. A evidência no vSphere mostra a tabela de caminhos sem alternativas ativas e a impossibilidade de I/O durante o período de falha total, comportamento alinhado com o desenho monocêntrico da arquitetura.

Para suporte documental dos eventos de falha total, foram recolhidas capturas adicionais da consola que evidenciam a perda simultânea de ligações FC e o estado global do dispositivo remoto no domínio de proteção, incluindo o diagrama de replicação e a condição *Link down* transversal às portas do storage afetado (Figuras 4.23, 4.24). Estas capturas enquadram a cronologia da falha e a sua propagação ao plano de dados, reforçando a leitura dos gráficos de desempenho.

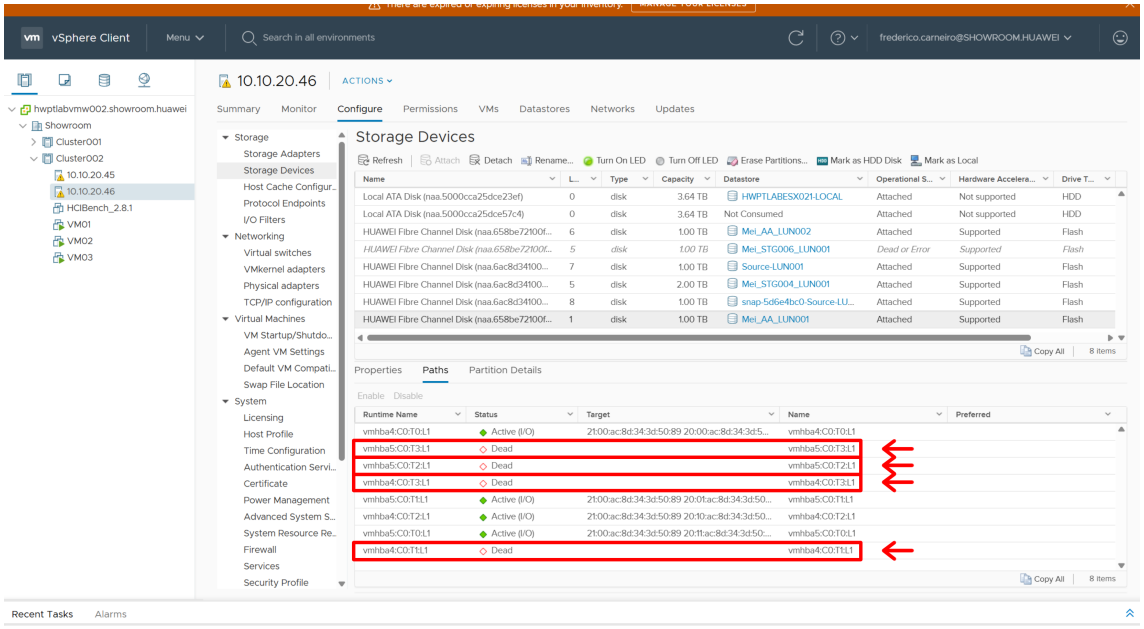


Figura 4.23: Falha total do storage STG B

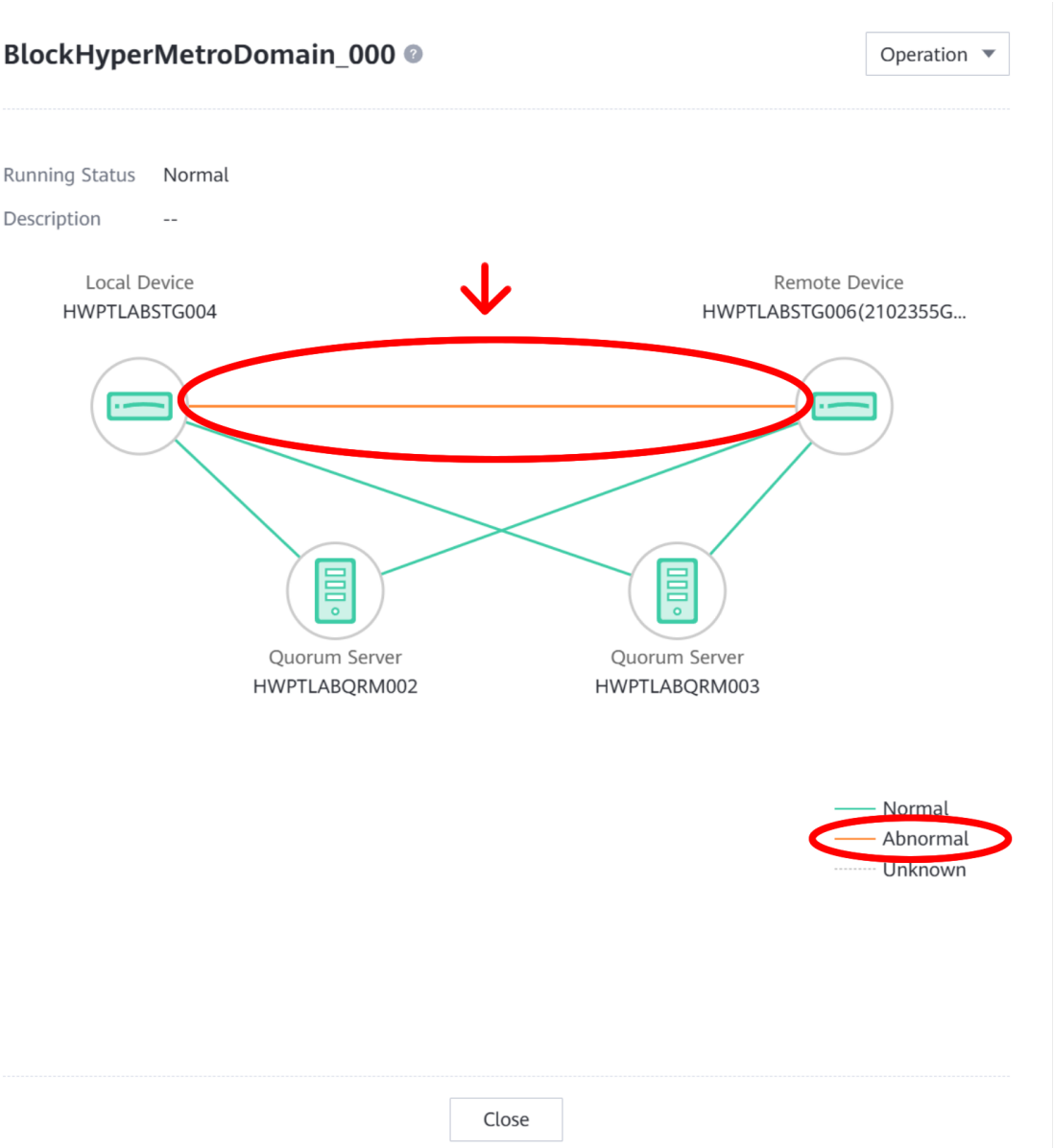


Figura 4.24: Perda de ligações FC durante falha total do storage STG B

No que respeita às ferramentas de monitorização de desempenho, os gráficos de CPU do controlador, IOPS, latência e throughput permitem seguir, segundo a segundo, a dinâmica de transição. No conjunto ativo-ativo com falha do STG B, o gráfico de utilização de CPU mostra apenas uma ligeira alteração na amplitude dos valores observados, sem sinais de saturação, o que sugere capacidade sobrante para absorver o *failover* (Figura 4.25). Já nos gráficos agregados do ensaio *out_aa_4k*, é possível identificar de forma precisa o instante da falha através de pequenas quebras nos traçados, imediatamente seguidos de recomposição. O alinhamento temporal destes sinais com o registo de alarme na consola confirma a correlação causa-efeito.

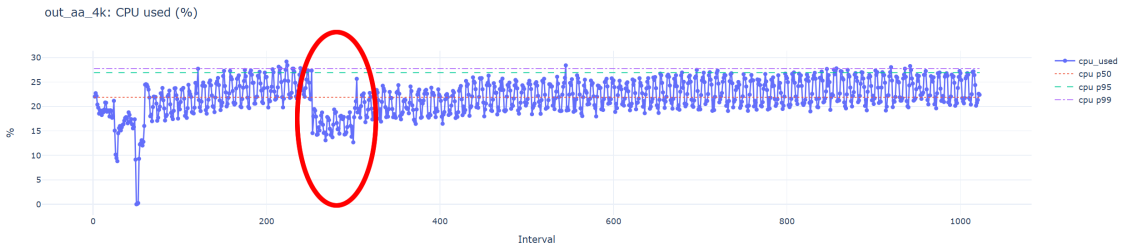


Figura 4.25: Utilização de CPU do controlador durante a falha total do storage STG B

Em síntese descritiva, as experiências de falha localizada, alimentação, módulo de interface, controladora e *fabric* SAN, demonstraram continuidade de acesso aos volumes graças ao *multipathing* e à redundância de ligações. A perda total de um storage produziu dois comportamentos distintos: indisponibilidade do serviço na arquitetura single por ausência de caminhos e operação contínua na arquitetura ativo-ativo com apenas uma oscilação pontual nos indicadores de desempenho aquando da comutação. Todas as observações estão sustentadas pelas evidências capturadas tanto na camada de gestão dos storages como na camada do hipervisor, e pelos registos de telemetria de I/O recolhidos durante os ensaios.

4.5 Automação e Verificação do Estado dos Equipamentos

A crescente complexidade das infraestruturas de armazenamento e a criticidade associada à disponibilidade de dados motivaram o desenvolvimento de mecanismos de monitorização que operem de forma contínua e independente da intervenção humana. Nesta secção apresenta-se o sistema de automação concebido e implementado para a verificação do estado operacional dos storages, baseado em Python e na API REST dos dispositivos de storage centralizado. O objetivo foi criar um processo inteiramente autónomo de recolha, análise e distribuição de informação, assegurando monitorização permanente e notificação proativa de anomalias.

4.5.1 Arquitetura do Sistema

A solução desenvolvida segue uma arquitetura modular composta por três componentes principais: recolha de dados, processamento e geração de relatórios, e distribuição automática de notificações. A comunicação com os storages é estabelecida através da interface REST disponibilizada pelos equipamentos, utilizando o protocolo HTTPS na porta 8088. Cada ciclo de execução inicia-se com o processo de autenticação, no qual o sistema solicita um *token* temporário de acesso (denominado *iBaseToken*), utilizado subsequentemente em todos os pedidos HTTP. A gestão do ciclo de vida do *token* foi automatizada, garantindo a persistência da sessão e a reautenticação em caso de expiração.

O sistema foi integralmente desenvolvido em Python 3, recorrendo à biblioteca `requests` para comunicação HTTP e à biblioteca `smtplib` para a distribuição de relatórios por email. A autenticação e o envio de notificações utilizam credenciais armazenadas em ficheiros externos com permissões restritivas, eliminando qualquer risco de exposição de dados sensíveis no código-fonte. A execução é orquestrada por uma tarefa cron configurada com a expressão `0 8 * * *`, que garante a execução diária às 08h00, assegurando monitorização contínua e geração regular de relatórios, conforme ilustrado na Listagem 4.1.

Listing 4.1: Cron job para health check

```
0 8 * * * cd /usr/src/vdbench && \
/usr/bin/python3 /usr/src/vdbench/daily_health_check.py >>
    ↪ \
/usr/src/vdbench/cron.log 2>&1
```

4.5.2 Processo de Implementação

A recolha de dados inicia-se com a consulta ao *endpoint* principal da API, de onde são obtidas as métricas globais de capacidade e estado de cada equipamento. Quando determinadas versões de firmware não disponibilizam o *endpoint* `/system/`, o script implementa um mecanismo de *fallback* sequencial, recorrendo a `/device/`, `/dashboard/` e `/overview/` até obter resposta válida. Esta abordagem assegura compatibilidade retroativa com diferentes gerações de firmware e evita falhas de recolha por variação de nomenclaturas.

Os cálculos de capacidade utilizam o mesmo algoritmo presente nas ferramentas oficiais do fabricante, baseado num fator de conversão setorial, garantindo consistência entre os valores reportados pelo sistema automatizado e os apresentados na interface nativa de administração. Durante o processo de monitorização, os dados recolhidos são estruturados em objetos JSON, permitindo a posterior serialização e o seu armazenamento em estruturas de dados Python.

A deteção e classificação de alarmes é efetuada através dos *endpoints* `/alarm/currentalarm/count` e `/alarm/currentalarm`, responsáveis, respetivamente, pela contagem e pelo detalhe de cada evento ativo. A informação obtida é normalizada por algoritmos que extraem os campos essenciais (nome, nível de severidade e descrição) e sintetizam os resultados em texto legível. O número de alarmes e a respetiva gravidade são destacados visualmente no relatório final, permitindo a identificação imediata de dispositivos em estado crítico (4.2).

Listing 4.2: Recolha e normalização de alarmes ativos

```
# 2. OBTER CONTAGEM DE ALARMES
alarm_count_url = f"{base_url}/{device_id}/alarm/currentalarm/count"
alarm_count_response = session.get(alarm_count_url, headers=auth_headers, timeout=30)
if alarm_count_response.status_code == 200:
    alarm_count_data = alarm_count_response.json()
    if 'data' in alarm_count_data and alarm_count_data['data']:
        alarm_count = alarm_count_data['data'].get('COUNT', 0)
        storage_info['Alarm_Count'] = int(alarm_count)
        storage_info['Alarmes'] = "ERROR" if storage_info['Alarm_Count'] > 0 else "OK"

# 3. SE HOUVER ALARMES, OBTER DETALHES
if storage_info['Alarm_Count'] > 0:
    alarm_detail_url = f"{base_url}/{device_id}/alarm/currentalarm"
    alarm_detail_response = session.get(alarm_detail_url, headers=auth_headers, timeout=30)
    if alarm_detail_response.status_code == 200:
        alarm_detail_data = alarm_detail_response.json()
        if 'data' in alarm_detail_data and alarm_detail_data['data']:
            alarms = alarm_detail_data['data']
            alarm_descriptions = []
            for alarm in alarms[:3]: # limitar a 3 alarmes
                name = alarm.get('name', 'Sem nome')
                level = alarm.get('level', 'N/A')
                alarm_descriptions.append(f"{name} (Nível: {level})")
            storage_info['Alarm_Details'] = " | ".join(alarm_descriptions)
else:
    storage_info['Alarm_Details'] = "Nenhum alarme ativo"
```

4.5.3 Elaboração de Relatórios e Distribuição

Os relatórios são produzidos em formato HTML e incluem um resumo estatístico global, tabelas detalhadas com capacidades, estado de saúde e alarmes, e uma legenda interpretativa. A formatação aplica código de cores normalizado para representar diferentes níveis de severidade, assegurando uma leitura rápida e intuitiva. Cada relatório contém ainda um carimbo temporal e o número total de storages analisados, servindo como histórico de auditoria. O excerto de código apresentado na Listagem 4.3 ilustra a secção responsável pela criação do relatório HTML e pela gravação local do ficheiro resultante:

Listing 4.3: Geração do relatório HTML

```
html_content = generate_html_report_final(all_storage_data)
with open("storage_report_final.html", "w",
          encoding="utf-8") as f:
    f.write(html_content)
```

A distribuição dos relatórios é efetuada automaticamente via SMTP sobre SSL, com envio simultâneo do conteúdo HTML no corpo da mensagem e em anexo. O servidor utilizado opera na porta 465, e as credenciais de autenticação são lidas a partir de um ficheiro externo. A mensagem é formatada como `multipart`, integrando o relatório e os metadados de execução. O processo de envio é totalmente automatizado, sem necessidade de intervenção manual, e inclui registo de logs e verificação de sucesso.

4.5.4 Desafios Técnicos e Soluções Implementadas

Durante o desenvolvimento, foram identificados diversos desafios associados à heterogeneidade dos equipamentos e versões de firmware. A variação de *endpoints* entre versões foi mitigada através de mecanismos de deteção dinâmica e seleção automática do método de recolha mais adequado. O comportamento volátil do *token* de autenticação foi resolvido com a utilização de sessões HTTP persistentes (Listagem 4.4), evitando a necessidade de reautenticação constante. Adicionalmente, foi implementado um tratamento robusto de exceções para garantir tolerância a falhas momentâneas de rede ou indisponibilidade de determinados serviços.

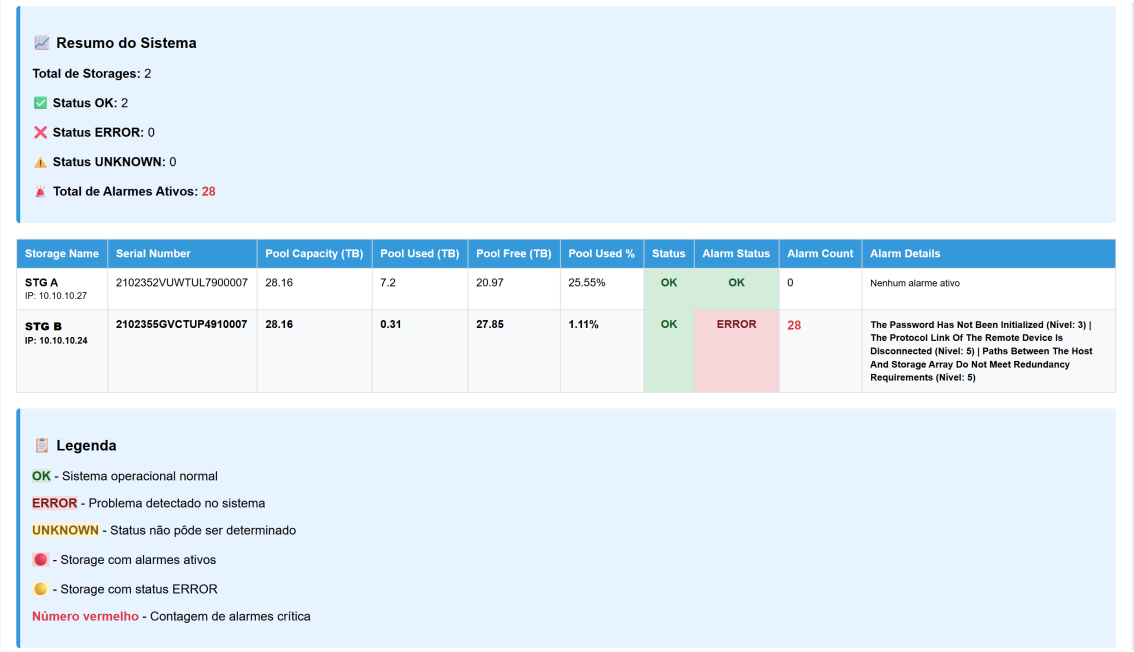
Listing 4.4: Sessão HTTP persistente e obtenção do iBaseToken via REST

```
session = requests.Session()
session.verify = False

response = session.post(session_url, headers=headers, json=body, timeout=30)
if response.status_code == 200:
    login_data = response.json()
    if 'data' in login_data and login_data['data']:
        device_id = login_data['data']['deviceid']
        iBaseToken = login_data['data']['iBaseToken']

        auth_headers = {
            "Content-Type": "application/json",
            "Accept": "application/json",
            "iBaseToken": iBaseToken
        }
```

A geração dos relatórios exigiu também uma abordagem cuidadosa de compatibilidade com diferentes clientes de email. Para tal, o CSS foi integrado diretamente no HTML, evitando dependências externas e assegurando que a renderização se mantém uniforme em ambientes heterogéneos. O código do sistema foi estruturado de modo modular, permitindo extensões futuras e integração com plataformas de monitorização mais complexas.

4.5.5 Validação e Resultados

O sistema foi validado em ambiente de produção com dois storages em operação, designados STG A e STG B. O teste de validação identificou corretamente os vinte e oito alarmes ativos no STG B, abrangendo avisos de licenças expiradas, falhas de conectividade de protocolo e alertas de redundância (Figura 4.26). Os relatórios gerados apresentaram total correspondência com os alarmes visíveis na consola administrativa dos equipamentos, confirmando a precisão da recolha e do processamento de dados.



Resumo do Sistema

Total de Storages: 2

Status OK: 2

Status ERROR: 0

Status UNKNOWN: 0

Total de Alarmes Ativos: 28

Storage Name	Serial Number	Pool Capacity (TB)	Pool Used (TB)	Pool Free (TB)	Pool Used %	Status	Alarm Status	Alarm Count	Alarm Details
STG A IP: 10.10.10.27	2102352VUWTUL7900007	28.16	7.2	20.97	25.55%	OK	OK	0	Nenhum alarme ativo
STG B IP: 10.10.10.24	2102355GVCTUP4910007	28.16	0.31	27.85	1.11%	OK	ERROR	28	The Password Has Not Been Initialized (Nível: 3) | The Protocol Link Of The Remote Device Is Disconnected (Nível: 5) | Paths Between The Host And Storage Array Do Not Meet Redundancy Requirements (Nível: 5)

Legenda

OK - Sistema operacional normal

ERROR - Problema detectado no sistema

UNKNOWN - Status não pôde ser determinado

- Storage com alarmes ativos

- Storage com status ERROR

Número vermelho - Contagem de alarmes crítica

Figura 4.26: Relatório do Daily Check recebido por e-mail

O sistema demonstrou ainda a capacidade de executar autonomamente de forma repetitiva, com geração diária de relatórios e envio automático por email, sem qualquer intervenção manual. A estabilidade observada ao longo de várias semanas de operação confirma a fiabilidade da solução e o seu potencial para integração em infraestruturas de armazenamento críticas.

Em síntese, a automação da verificação de estado dos storages permitiu transformar um processo manual e moroso numa operação totalmente automatizada, garantindo monitorização contínua, uniformidade de resultados e redução significativa do tempo de deteção de incidentes. O código integral da solução encontra-se documentado no Apêndice A.

Esta página foi intencionalmente deixada em branco.

Capítulo 5

Conclusões

A crescente dependência das organizações em infraestruturas digitais críticas tem vindo a acentuar a importância da disponibilidade contínua e da integridade dos dados como elementos estruturais da resiliência empresarial. No contexto atual, em que as operações dependem fortemente de sistemas de informação e em que os ataques cibernéticos representam uma ameaça constante, a continuidade do negócio deixou de ser uma mera vantagem competitiva para se tornar uma exigência operacional. O presente trabalho contribui para este enquadramento ao demonstrar, de forma prática e fundamentada, a viabilidade técnica da implementação de uma solução de armazenamento centralizado em modo ativo ativo com replicação síncrona e proteção avançada contra ransomware, combinando desempenho, redundância e segurança num modelo aplicável a infraestruturas empresariais de diferentes dimensões.

Ambas as arquiteturas estudadas, a solução Single e a solução Ativo Ativo, foram implementadas com redundância completa em todos os componentes críticos, incluindo fontes de alimentação duplas, múltiplas controladoras, ligações SAN redundantes e em Fabric Par e Fabric Ímpar. Esta abordagem assegura a continuidade de serviço perante falhas parciais de hardware, evitando interrupções mesmo em cenários de falha de controladora, fonte de alimentação ou outro componente. Nos testes realizados, todas as redundâncias foram colocadas à prova, verificando-se que a infraestrutura reagiu conforme esperado, sem perda de dados e sem interrupção dos serviços, embora com uma degradação temporária do desempenho em situações de falha de controladora. Estes resultados validam a robustez da arquitetura subjacente e a maturidade do modelo de redundância adotado.

A análise comparativa entre as duas arquiteturas demonstrou que a solução Single apresenta maior desempenho bruto em termos de IOPS e latência, consequência direta da ausência de replicação síncrona entre controladores. No entanto, essa vantagem de desempenho vem acompanhada de um risco estrutural resultante da existência de um ponto único de falha. Já a arquitetura Ativo Ativo, embora apresente uma ligeira penalização de latência decorrente da confirmação simultânea de escrita em ambos os nós, garante um nível de disponibilidade substancialmente superior, permitindo que a infraestrutura continue a operar mesmo em caso de falha total de um dos equipamentos ou de um centro de dados. Esta característica

traduz-se numa proteção efetiva contra incidentes graves, incluindo desastres naturais, incêndios ou falhas elétricas severas, assegurando que a produção permanece operacional e que o impacto no negócio é praticamente nulo.

Os testes de redundância e recuperação efetuados demonstraram que a infraestrutura é capaz de lidar com falhas críticas de forma previsível e controlada. A replicação síncrona entre sistemas garante um Recovery Point Objective (RPO) igual a zero, assegurando a inexistência de perda de dados em caso de falha, enquanto o Recovery Time Objective (RTO) se mantém próximo de zero, dado que o comutador de serviço ocorre automaticamente e sem intervenção manual. A camada de proteção adicional, baseada em Secure Snapshots do tipo WORM (Write Once, Read Many), provou ser eficaz na recuperação de dados corrompidos ou encriptados, permitindo restaurar volumes afetados a partir de cópias imutáveis e verificáveis. Este mecanismo representa uma defesa robusta contra ransomware, garantindo que, mesmo após um ataque, a recuperação é possível sem dependência de terceiros e sem risco de reinfeção dos dados restaurados.

Outro contributo relevante deste trabalho reside na demonstração prática de que soluções tradicionalmente associadas a grandes organizações, com forte investimento em disaster recovery e requisitos normativos rigorosos como o DORA ou a NIS2, podem ser adaptadas e implementadas em contextos de pequenas e médias empresas. A solução proposta alia robustez técnica e exequibilidade financeira, mostrando que é possível alcançar elevados níveis de disponibilidade e proteção de dados sem recorrer a infraestruturas excessivamente complexas ou dispendiosas. Esta abordagem democratiza o acesso a tecnologias de armazenamento de classe empresarial, permitindo que organizações de menor dimensão alcancem maturidade tecnológica compatível com as exigências de segurança e continuidade atuais.

Do ponto de vista prático, a execução deste projeto apresentou desafios significativos, sobretudo pela natureza avançada das tecnologias envolvidas e pela necessidade de integração de múltiplos domínios como storage, redes, automação e cibersegurança. A implementação e validação da arquitetura ativo ativo, associadas à criação de um sistema de automação para verificação de estado dos equipamentos, exigiram uma abordagem iterativa e uma forte componente de experimentação técnica. Contudo, essas dificuldades foram ultrapassadas, resultando numa solução funcional, fiável e replicável em diferentes contextos empresariais.

A consolidação destes resultados reforça a ideia de que, num panorama cada vez mais digital e interligado, a questão já não reside em "se" uma organização será alvo de ataque ou falha, mas em "quando" isso acontecerá e qual a sua capacidade de resposta. A arquitetura proposta e validada neste trabalho constitui uma resposta concreta e tecnicamente fundamentada a essa realidade, demonstrando que a resiliência digital é alcançável através de soluções bem concebidas, sustentadas em replicação, redundância e imutabilidade dos dados. Este projeto, ao conjugar teoria, implementação prática e análise comparativa rigorosa, contribui de forma efetiva para o avanço do conhecimento técnico na área de armazenamento centralizado e continuidade de negócio, servindo como referência aplicável a futuras implementações em ambientes empresariais críticos.

Referências

Bunn, F., Simpson, N., Peglar, R., & Nagle, G. (2004). *Storage virtualization.* Storage Networking Industry Association (SNIA). Retrieved from https://www.snia.org/sites/default/files/sniavirt.pdf

Castiglione, J., & Pavlovic, D. (2020). Dynamic distributed secure storage against ransomware. *IEEE Transactions on Computational Social Systems*, *7*(6), 1469–1475. Retrieved from https://doi.org/10.1109/TCSS.2019.2924650 doi: 10.1109/TCSS.2019.2924650

Cawthra, J., et al. (2020a). *Data integrity: Detecting and responding to ransomware and other destructive events* (Tech. Rep. No. NIST Special Publication 1800-26). National Institute of Standards and Technology. Retrieved from https://doi.org/10.6028/NIST.SP.1800-26 doi: 10.6028/NIST.SP.1800-26

Cawthra, J., et al. (2020b). *Data integrity: Identifying and protecting assets against ransomware and other destructive events* (Tech. Rep. No. NIST Special Publication 1800-25). National Institute of Standards and Technology. Retrieved from https://doi.org/10.6028/NIST.SP.1800-25 doi: 10.6028/NIST.SP.1800-25

Darrous, J., Ibrahim, S., & Perez, C. (2019). Is it time to revisit erasure coding in data-intensive clusters? In *2019 ieee 27th international symposium on modeling, analysis, and simulation of computer and telecommunication systems (mascots)* (p. 165-178). doi: 10.1109/MASCOTS.2019.00026

Dell Technologies. (2022). *Dell powerstore: Replication technologies* (Tech. Rep.). Dell Technologies. Retrieved from https://www.delltechnologies.com/asset/en-us/products/storage/industry-market/h18153-dell-powerstore-replication-technologies.pdf

Durner, D., Leis, V., & Neumann, T. (2023, 08). Exploiting cloud object storage for high-performance analytics. *Proceedings of the VLDB Endowment*, *16*, 2769-2782. doi: 10.14778/3611479.3611486

European Parliament and Council of the European Union. (2016). *Regulation (eu) 2016/679 of 27 april 2016 on the protection of natural persons with regard to the processing of personal data and on the free movement of such data (general data protection regulation).* Official Journal of the European Union L 119. Retrieved from https://eur-lex.europa.eu/eli/reg/2016/679/oj (Accessed December 2025)

Grafana Labs. (2024). *Grafana: Open-source analytics and monitoring platform.* `https://grafana.com`. (Version 11.0. Grafana Labs. Retrieved from https://grafana.com)

Graphite Project. (2024). *Graphite: Scalable time-series monitoring and graphing system.* `https://graphiteapp.org`. (Version 1.3.0. Graphite Project. Retrieved from https://graphiteapp.org)

Hewlett-Packard. (2008). *Hp storageworks arrays family guide* (Tech. Rep.). Hewlett-Packard Development Company, L.P. Retrieved from `https://media.zones.com/images/pdf/ss_hplh_ds_02.pdf`

Huawei Technologies Co., Ltd. (2017). *Active-active data center solution technical white paper* (Tech. Rep.). Huawei Technologies Co., Ltd. Retrieved from `https://actfornet.com/ueditor/php/upload/file/20190103/1546486958172149.pdf`

Huawei Technologies Co., Ltd. (2024). *Oceanstor dorado v6 series hypermetro feature guide for block.* (Internal technical documentation consulted for validation of latency and distance parameters.)

IBM Security. (2024). *Cost of a data breach report 2024* (Technical Report). IBM Corporation. Retrieved from `https://www.ibm.com/reports/data-breach`

Kharraz, A., Robertson, W., Balzarotti, D., Bilge, L., & Kirda, E. (2015). Cutting the gordian knot: A look under the hood of ransomware attacks. In *Detection of intrusions and malware, and vulnerability assessment (dimva 2015)* (Vol. 9148, pp. 3–24). Springer. Retrieved from `https://doi.org/10.1007/978-3-319-20550-2_1` doi: 10.1007/978-3-319-20550-2_1

Macedo, R., Paulo, J., Pereira, J., & Bessani, A. (2020, 05). A survey and classification of software-defined storage systems. *ACM Computing Surveys*, *53*, 1-38. doi: 10.1145/3385896

National Institute of Standards and Technology. (2024). *The nist cybersecurity framework (csf) 2.0* (Tech. Rep. No. NIST CSWP 29). NIST. Retrieved from `https://doi.org/10.6028/NIST.CSWP.29` doi: 10.6028/NIST.CSWP.29

Oracle Corporation. (2023). *Vdbench: Disk i/o workload generator for storage performance and data integrity validation.* `https://www.oracle.com/downloads/server-storage/vdbench-downloads.html`. (Version 5.04.06. Oracle Corporation. Retrieved from https://www.oracle.com/downloads/server-storage/vdbench-downloads.html)

Parliament, T. E., & the Council of the European Union. (2022, dez). *Regulation (eu) 2022/2554 of the european parliament and of the council.* online on: https://eur-lex.europa.eu/legal-content/EN/TXT/PDF/?uri=CELEX:32022R2554.

Patterson, D. A., Gibson, G., & Katz, R. H. (1988, June). A case for redundant arrays of inexpensive disks (raid). *SIGMOD Rec.*, *17*(3), 109–116. Retrieved from `https://doi.org/10.1145/971701.50214` doi: 10.1145/971701.50214

Paulo, J., & Pereira, J. (2014). A survey and classification of storage deduplication systems. *ACM Computing Surveys*, *47*(1), 11:1–11:30. Retrieved from https://doi.org/10.1145/2611778 doi: 10.1145/2611778

Pearson, D. (2023). *High availability critical for storage in the digital business era* (Tech. Rep. No. US51241123). IDC. Retrieved from https://www.netapp.com/ja/media/65047-WP-IDC-High-Availability-Req-Digital-Enterprises.pdf (White Paper sponsored by NetApp)

Quest Software Inc. (2016). *Active-active replication and considerations for high availability* (Tech. Rep.). Quest Software. Retrieved from https://www.quest.com/whitepaper/activeactive-replication-and-considerations-for-high-availability/

Raghunatha, M., & Ramappa, R. (2009). *Dell reference configuration for large oracle oltp deployments on dell equallogic storage* (White Paper). Dell Inc. Retrieved from https://i.dell.com/sites/csdocuments/Business_solutions_engineering-Docs_Documents/fr/fr/large-oracle-deployments_fr.pdf

Ravi Kumar, M., et al. (2021). Network-attached storage: Data storage applications. *Turkish Journal of Computer and Mathematics Education (TURCOMAT)*, *12*(12), 2385–2396.

Ray, A., & To, L. (2008). *Deploying active-active data centers using oracle database solutions* (Tech. Rep.). Oracle Corporation. Retrieved from https://www.oracle.com/technetwork/database/features/availability/300460-132393.pdf

Riabov, V. V. (2004). Storage area networks (sans). In *The internet encyclopedia.* John Wiley & Sons, Ltd. Retrieved from https://onlinelibrary.wiley.com/doi/abs/10.1002/047148296X.tie164 doi: https://doi.org/10.1002/047148296X.tie164

Sinclair, S., & Keane, M. (2022). *Storage's role in addressing the challenges of ensuring cyber resilience* (Tech. Rep.). Enterprise Strategy Group. Retrieved from https://horizoncomputersolutionsinc.kinsta.cloud/wp-content/uploads/2023/04/ESG-Storages-Role-in-the-Challenges-of-Ensuring-Cyber-Resilience.pdf (White paper commissioned by IBM)

Storage Networking Industry Association. (2017a). *Data protection best practices* (White Paper). SNIA. Retrieved from https://www.snia.org/sites/default/files/technical-work/whitepapers/SNIA-Data-Protection-Best-Practices-White%20Paper.pdf (Accessed December 2025)

Storage Networking Industry Association. (2017b). *Data protection best practices* (Technical White Paper). SNIA. Retrieved from https://www.snia.org/sites/default/files/technical-work/whitepapers/SNIA-Data-Protection-Best-Practices-White%20Paper.pdf

Townsend, A., McBride, T., Lusty, L., Sexton, J., & Ekstrom, M. (2020). *Data integrity: Recovering from ransomware and other destructive events* (NIST Special Publication No. 1800-11). National Institute of Standards and Technology. Retrieved from `https://doi.org/10.6028/NIST.SP.1800-11` doi: 10.6028/NIST.SP.1800-11

Apêndice A

Repositório de Código

Este apêndice reúne o código fonte utilizado no projeto, organizado por grupos funcionais. Os scripts são apresentados na íntegra, mantendo a formatação original, e acompanhados por uma legenda identificativa.

A.1 Scripts para Análise das Soluções single e ativo-ativo

Os ficheiros `.vdb` definem os perfis de carga utilizados nos ensaios com o VDBench (tamanhos de bloco de 4 KiB a 64 KiB, perfil 70/30 leitura/escrita), garantindo reprodutibilidade entre execuções. Os restantes scripts automatizam a execução, consolidação e visualização dos resultados.

Perfis VDBench (arquitetura ativo-ativo)

```
###########
# VDBENCH Parameter File
# Name: teste_aa_4k_70-30random.vdb
# Author: Frederico Carneiro
# Date: 10/07/2025
###########

# --- Hosts (Define slaves) ---
hd=vm01,user=root,vdbench=/usr/src/vdbench/
hd=vm02,user=root,vdbench=/usr/src/vdbench/

# --- Storage Definition (raw + o_direct + detail) ---
sd=sd1,host=*,lun=/dev/sdb,openflags=o_direct
sd=sd2,host=*,lun=/dev/sdc,openflags=o_direct

# --- Workload Definition (4k, 70/30, random) ---
wd=wd1,sd=sd*,xfersize=4k,rdpct=70,seekpct=100
```

```
18
19 # --- Run Definition (p50/p95/p99 via histogram; sem format) ---
20 rd=run1,wd=wd1,elapsed=900,interval=1,iorate=max,warmup=120,threads=64
```

Listing A.1: Perfil VDBench – ativo-ativo (4 KiB, 70/30, random)

```
1  ############
2  # VDBENCH Parameter File
3  # Name: teste_aa_8k_70-30random.vdb
4  # Author: Frederico Carneiro
5  # Email:
6  # Date: 10/07/2025
7  ##################
8
9  # --- Hosts (Define slaves) ---
10 # --- 1.13Host Definition parameter detail --
11 hd=vm01,user=root,vdbench=/usr/src/vdbench/     # slave 1
      ↪ (hostname ou FQDN resolvível por SSH)
12 #hd=vm02,user=root,vdbench=/usr/src/vdbench/    # slave 2
      ↪ (hostname ou FQDN resolvível por SSH)
13
14 # -- Storage Definition (set devices,  run with raw device
      ↪ using o_direct)--
15 # -- 1.14 Storage Definition parameter detail --
16 sd=sd1,host=*,lun=/dev/sdb,openflags=o_direct
17 sd=sd2,host=*,lun=/dev/sdc,openflags=o_direct
18
19 # -- Workload Defition (set 4k Block Size, read-write 70-30,
      ↪ 100% random) --
20 # -- 1.15 Workload Definition parameter detail --
21 wd=wd1,sd=sd*,xfersize=8k,rdpct=70,seekpct=100
22
23 # -- Run Definition (set 2m, report 1s, iops no limit, warmup
      ↪ 1m, with 64 Threads) --
24 # -- 1.16Run Definition for raw I/O parameter detail --
25 rd=run1,wd=wd1,elapsed=900,interval=1,iorate=max,warmup=120,threads=64
```

Listing A.2: Perfil VDBench – ativo-ativo (8 KiB, 70/30, random)

```
1  ############
2  # VDBENCH Parameter File
3  # Name: teste_aa_16k_70-30random.vdb
4  # Author: Frederico Carneiro
5  # Email:
6  # Date: 10/07/2025
7  ##################
8
9  # --- Hosts (Define slaves) ---
10 # --- 1.13Host Definition parameter detail --
11 hd=vm01,user=root,vdbench=/usr/src/vdbench/     # slave 1
      ↪ (hostname ou FQDN resolvível por SSH)
```

```
#hd=vm02,user=root,vdbench=/usr/src/vdbench/    # slave 2 (hostname ou FQDN resolvível por SSH)

# -- Storage Definition (set devices,  run with raw device using o_direct)--
# -- 1.14 Storage Definition parameter detail --
sd=sd1,host=*,lun=/dev/sdb,openflags=o_direct
sd=sd2,host=*,lun=/dev/sdc,openflags=o_direct

# -- Workload Defition (set 4k Block Size, read-write 70-30, 100% random) --
# -- 1.15 Workload Definition parameter detail --
wd=wd1,sd=sd*,xfersize=16k,rdpct=70,seekpct=100

# -- Run Definition (set 2m, report 1s, iops no limit, warmup 1m, with 64 Threads) --
# -- 1.16Run Definition for raw I/O parameter detail --
rd=run1,wd=wd1,elapsed=900,interval=1,iorate=max,warmup=120,threads=64
```

Listing A.3: Perfil VDBench – ativo-ativo (16 KiB, 70/30, random)

```
############
# VDBENCH Parameter File
# Name: teste_aa_32k_70-30random.vdb
# Author: Frederico Carneiro
# Email:
# Date: 10/07/2025
##################

# --- Hosts (Define slaves) ---
# --- 1.13Host Definition parameter detail --
hd=vm01,user=root,vdbench=/usr/src/vdbench/     # slave 1 (hostname ou FQDN resolvível por SSH)
#hd=vm02,user=root,vdbench=/usr/src/vdbench/    # slave 2 (hostname ou FQDN resolvível por SSH)

# -- Storage Definition (set devices,  run with raw device using o_direct)--
# -- 1.14 Storage Definition parameter detail --
sd=sd1,host=*,lun=/dev/sdb,openflags=o_direct
sd=sd2,host=*,lun=/dev/sdc,openflags=o_direct

# -- Workload Defition (set 4k Block Size, read-write 70-30, 100% random) --
# -- 1.15 Workload Definition parameter detail --
wd=wd1,sd=sd*,xfersize=32k,rdpct=70,seekpct=100

# -- Run Definition (set 2m, report 1s, iops no limit, warmup 1m, with 64 Threads) --
# -- 1.16Run Definition for raw I/O parameter detail --
rd=run1,wd=wd1,elapsed=900,interval=1,iorate=max,warmup=120,threads=64
```

Listing A.4: Perfil VDBench – ativo-ativo (32 KiB, 70/30, random)

```
############
# VDBENCH Parameter File
# Name: teste_aa_64k_70-30random.vdb
# Author: Frederico Carneiro
# Email:
# Date: 10/07/2025
#################

# --- Hosts (Define slaves) ---
# --- 1.13Host Definition parameter detail --
hd=vm01,user=root,vdbench=/usr/src/vdbench/     # slave 1
   ↪ (hostname ou FQDN resolvível por SSH)
#hd=vm02,user=root,vdbench=/usr/src/vdbench/    # slave 2
   ↪ (hostname ou FQDN resolvível por SSH)

# -- Storage Definition (set devices,  run with raw device
   ↪ using o_direct)--
# -- 1.14 Storage Definition parameter detail --
sd=sd1,host=*,lun=/dev/sdb,openflags=o_direct
sd=sd2,host=*,lun=/dev/sdc,openflags=o_direct

# -- Workload Defition (set 4k Block Size, read-write 70-30,
   ↪ 100% random) --
# -- 1.15 Workload Definition parameter detail --
wd=wd1,sd=sd*,xfersize=64k,rdpct=70,seekpct=100

# -- Run Definition (set 2m, report 1s, iops no limit, warmup
   ↪ 1m, with 64 Threads) --
# -- 1.16Run Definition for raw I/O parameter detail --
rd=run1,wd=wd1,elapsed=900,interval=1,iorate=max,warmup=120,threads=64
```

Listing A.5: Perfil VDBench – ativo-ativo (64 KiB, 70/30, random)

Perfis VDBench (arquitetura single)

```
############
# VDBENCH Parameter File
# Name: teste_single-30random.vdb
# Author: Frederico Carneiro
# Email:
# Date: 10/07/2025
#################

# --- Hosts (Define slaves) ---
# --- 1.13Host Definition parameter detail --
```

```
hd=vm03,user=root,vdbench=/usr/src/vdbench/     # slave 1 (hostname ou FQDN resolvível por SSH)

# -- Storage Definition (set devices,  run with raw device using o_direct)--
# -- 1.14 Storage Definition parameter detail --
sd=sd1,host=*,lun=/dev/sdb,openflags=o_direct
sd=sd2,host=*,lun=/dev/sdc,openflags=o_direct

# -- Workload Defition (set 4k Block Size, read-write 70-30, 100% random) --
# -- 1.15 Workload Definition parameter detail --
wd=wd1,sd=sd*,xfersize=4k,rdpct=70,seekpct=100

# -- Run Definition (set 2m, report 1s, iops no limit, warmup 1m, with 64 Threads) --
# -- 1.16Run Definition for raw I/O parameter detail --
rd=run1,wd=wd1,elapsed=900,interval=1,iorate=max,warmup=120,threads=64
```

Listing A.6: Perfil VDBench – single (4 KiB, 70/30, random)

```
############
# VDBENCH Parameter File
# Name: teste_single-30random.vdb
# Author: Frederico Carneiro
# Email:
# Date: 10/07/2025
##################

# --- Hosts (Define slaves) ---
# --- 1.13Host Definition parameter detail --
hd=vm03,user=root,vdbench=/usr/src/vdbench/     # slave 1 (hostname ou FQDN resolvível por SSH)

# -- Storage Definition (set devices,  run with raw device using o_direct)--
# -- 1.14 Storage Definition parameter detail --
sd=sd1,host=*,lun=/dev/sdb,openflags=o_direct
sd=sd2,host=*,lun=/dev/sdc,openflags=o_direct

# -- Workload Defition (set 8k Block Size, read-write 70-30, 100% random) --
# -- 1.15 Workload Definition parameter detail --
wd=wd1,sd=sd*,xfersize=8k,rdpct=70,seekpct=100

# -- Run Definition (set 2m, report 1s, iops no limit, warmup 1m, with 64 Threads) --
# -- 1.16Run Definition for raw I/O parameter detail --
rd=run1,wd=wd1,elapsed=900,interval=1,iorate=max,warmup=120,threads=64
```

Listing A.7: Perfil VDBench – single (8 KiB, 70/30, random)

```
###########
# VDBENCH Parameter File
# Name: teste_single-30random.vdb
# Author: Frederico Carneiro
# Email:
# Date: 10/07/2025
##################

# --- Hosts (Define slaves) ---
# --- 1.13Host Definition parameter detail --
hd=vm03,user=root,vdbench=/usr/src/vdbench/     # slave 1
    ↪ (hostname ou FQDN resolvível por SSH)

# -- Storage Definition (set devices,  run with raw device
    ↪ using o_direct)--
# -- 1.14 Storage Definition parameter detail --
sd=sd1,host=*,lun=/dev/sdb,openflags=o_direct
sd=sd2,host=*,lun=/dev/sdc,openflags=o_direct

# -- Workload Defition (set 4k Block Size, read-write 70-30,
    ↪ 100% random) --
# -- 1.15 Workload Definition parameter detail --
wd=wd1,sd=sd*,xfersize=16k,rdpct=70,seekpct=100

# -- Run Definition (set 2m, report 1s, iops no limit, warmup
    ↪ 1m, with 64 Threads) --
# -- 1.16Run Definition for raw I/O parameter detail --
rd=run1,wd=wd1,elapsed=900,interval=1,iorate=max,warmup=120,threads=64
```

Listing A.8: Perfil VDBench – single (16 KiB, 70/30, random)

```
###########
# VDBENCH Parameter File
# Name: teste_single-30random.vdb
# Author: Frederico Carneiro
# Email:
# Date: 10/07/2025
##################

# --- Hosts (Define slaves) ---
# --- 1.13Host Definition parameter detail --
hd=vm03,user=root,vdbench=/usr/src/vdbench/     # slave 1
    ↪ (hostname ou FQDN resolvível por SSH)

# -- Storage Definition (set devices,  run with raw device
    ↪ using o_direct)--
# -- 1.14 Storage Definition parameter detail --
sd=sd1,host=*,lun=/dev/sdb,openflags=o_direct
sd=sd2,host=*,lun=/dev/sdc,openflags=o_direct

```

```
# -- Workload Defition (set 4k Block Size, read-write 70-30, 100% random) --
# -- 1.15 Workload Definition parameter detail --
wd=wd1,sd=sd*,xfersize=32k,rdpct=70,seekpct=100

# -- Run Definition (set 2m, report 1s, iops no limit, warmup 1m, with 64 Threads) --
# -- 1.16Run Definition for raw I/O parameter detail --
rd=run1,wd=wd1,elapsed=900,interval=1,iorate=max,warmup=120,threads=64
```

Listing A.9: Perfil VDBench – single (32 KiB, 70/30, random)

```
############
# VDBENCH Parameter File
# Name: teste_single-30random.vdb
# Author: Frederico Carneiro
# Email:
# Date: 10/07/2025
##################

# --- Hosts (Define slaves) ---
# --- 1.13Host Definition parameter detail --
hd=vm03,user=root,vdbench=/usr/src/vdbench/     # slave 1 (hostname ou FQDN resolvível por SSH)

# -- Storage Definition (set devices,  run with raw device using o_direct)--
# -- 1.14 Storage Definition parameter detail --
sd=sd1,host=*,lun=/dev/sdb,openflags=o_direct
sd=sd2,host=*,lun=/dev/sdc,openflags=o_direct

# -- Workload Defition (set 4k Block Size, read-write 70-30, 100% random) --
# -- 1.15 Workload Definition parameter detail --
wd=wd1,sd=sd*,xfersize=64k,rdpct=70,seekpct=100

# -- Run Definition (set 2m, report 1s, iops no limit, warmup 1m, with 64 Threads) --
# -- 1.16Run Definition for raw I/O parameter detail --
rd=run1,wd=wd1,elapsed=900,interval=1,iorate=max,warmup=120,threads=64
```

Listing A.10: Perfil VDBench – single (64 KiB, 70/30, random)

Automação de execução e análise de resultados

Listing A.11: Consolidação e análise de execuções VDBench

```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-

"""
```

```
VDbench run visualizer:
- Parses VDbench flatfile.html (text/ASCII with columns, sometimes line-wrapped)
- Builds per-run time-series charts (IOPS/rate, MB/sec, resp, read_resp, write_resp, CPU used, Read%)
- Draws horizontal percentile bands (p50/p95/p99)
- White background (plotly_white)
- Outputs a single HTML report: vdbench_runs_report.html
Usage:
   python3 vdbench_summarize.py out4k out8k ...
"""

import os
import sys
import re
import numpy as np
import pandas as pd
import plotly.graph_objects as go

# -------- Robust flatfile parser (for typical VDbench ASCII "flatfile.html") --------

# Common leading columns in many VDbench builds.
# We'll keep only the ones we care about later; extras are ignored safely.
EXPECTED_COLS = [
    "tod", "date_zone", "run", "interval",
    "reqrate", "rate", "MB/sec", "bytes/io",
    "read%", "resp", "read_resp", "write_resp",
    "resp_max", "read_max", "write_max",
    "resp_std", "read_std", "write_std",
    "xfersize", "threads", "rdpct", "rhpct", "whpct", "seekpct",
    "lunsize_gb", "version", "compratio", "dedupratio",
    "queue_depth", "cpu_used", "cpu_user", "cpu_kernel",
    # some builds add more after this; we simply ignore if present
]

def _is_new_record(line: str) -> bool:
    """Detect the start of a new data row by timestamp like 'HH:MM:SS.xxx' at line start."""
    return bool(re.match(r"^\d{2}:\d{2}:\d{2}\.\d{3}", line))

def read_flatfile_text(out_dir: str) -> pd.DataFrame:
    """Read VDbench flatfile.html that is actually plain text with columns."""
    ff = os.path.join(out_dir, "flatfile.html")
    if not os.path.exists(ff):
        raise FileNotFoundError(f"{ff} does not exist")

    # 1) Strip comments (*) and HTML tags (<...>)
    raw = []
    with open(ff, "r", errors="ignore") as f:
        for ln in f:
            s = ln.rstrip("\n")
            if not s.strip():
                continue
            s_stripped = s.lstrip()
            if s_stripped.startswith("*") or s_stripped.startswith("<"):
                continue
            raw.append(s.strip())

    if not raw:
        raise ValueError(f"{ff}: empty after removing comments/tags")

    # 2) Join wrapped lines: if a line does NOT start with timestamp, append to previous line
    joined = []
    for ln in raw:
        if _is_new_record(ln) or not joined:
            joined.append(ln)
        else:
            joined[-1] = (joined[-1] + " " + ln).strip()

    # 3) Tokenize each joined line by whitespace and map to columns
    rows = []
    for ln in joined:
        toks = ln.split()
        # tolerate very short (skip)
        if len(toks) < 10:
```

```
            continue
        # keep as many tokens as EXPECTED_COLS, ignore overflow
        take = toks[:len(EXPECTED_COLS)]
        # pad if somehow shorter
        if len(take) < len(EXPECTED_COLS):
            take = take + [""] * (len(EXPECTED_COLS) - len(take))
        rows.append(take)

    if not rows:
        raise ValueError(f"{ff}: no valid data rows parsed")

    df = pd.DataFrame(rows, columns=EXPECTED_COLS)

    # convert numeric columns where applicable
    numeric_cols = [
        "interval","reqrate","rate","MB/sec","bytes/io","read%","resp","read_resp","write_resp",
        "resp_max","read_max","write_max","resp_std","read_std","write_std","xfersize","threads",
        "rdpct","rhpct","whpct","seekpct","lunsize_gb","compratio","dedupratio","queue_depth",
        "cpu_used","cpu_user","cpu_kernel"
    ]
    for c in numeric_cols:
        if c in df.columns:
            df[c] = pd.to_numeric(df[c], errors="coerce")

    # Normalize interval (1..N) if missing or unparseable
    if "interval" in df.columns and df["interval"].notna().any():
        try:
            df["interval"] = df["interval"].astype(int)
        except Exception:
            df["interval"] = np.arange(1, len(df) + 1)
    else:
        df["interval"] = np.arange(1, len(df) + 1)

    # Ensure run label exists
    if "run" not in df.columns or df["run"].isna().all():
        df["run"] = "run"

    return df

# -------- Plot helpers --------

def add_percentile_bands(fig: go.Figure, series: pd.Series, name_prefix: str, x):
    """Draw horizontal p50/p95/p99 lines over a given series."""
    s = pd.to_numeric(series, errors="coerce").dropna()
    if s.empty:
        return
    pvals = {
        "p50": float(np.percentile(s, 50)),
        "p95": float(np.percentile(s, 95)),
        "p99": float(np.percentile(s, 99)),
    }
    # x-domain (works for numeric intervals)
    try:
        x_numeric = pd.to_numeric(x, errors="coerce")
        x0 = float(x_numeric.min())
        x1 = float(x_numeric.max())
    except Exception:
        try:
            x0, x1 = (min(x), max(x))
        except Exception:
            x0, x1 = (0, len(s))

    dash_map = {"p50": "dot", "p95": "dash", "p99": "dashdot"}
    for lbl, val in pvals.items():
        fig.add_trace(
            go.Scatter(
                x=[x0, x1], y=[val, val],
                mode="lines",
                name=f"{name_prefix} {lbl}",
                line=dict(width=1.2, dash=dash_map[lbl]),
                hovertemplate=f"{name_prefix} {lbl}: {val:.4f}<extra></extra>",
            )
```

```
        )

def time_series_fig(x, y, title, y_title, series_name):
    fig = go.Figure()
    fig.add_trace(go.Scatter(
        x=x, y=y, mode="lines+markers", name=series_name,
        hovertemplate="%{x}<br>%{y:.4f}<extra></extra>"
    ))
    fig.update_layout(
        template="plotly_white",
        title=title,
        xaxis_title="Interval",
        yaxis_title=y_title,
        legend_title="",
        height=420
    )
    return fig

def write_or_append(html_path: str, html_snippet: str, first: bool):
    if first:
        with open(html_path, "w", encoding="utf-8") as f:
            f.write("<!doctype html><html><head><meta charset='utf-8'>"
                    "<title>VDbench Run Report</title></head><body>\n")
            f.write("<h1 style='font-family:sans-serif'>VDbench Run Report</h1>\n")
            f.write(html_snippet)
    else:
        with open(html_path, "a", encoding="utf-8") as f:
            f.write(html_snippet)

# -------- Main --------

def main():
    if len(sys.argv) < 2:
        print("Usage: python3 vdbench_summarize.py <out_dir1> [out_dir2 ...]")
        sys.exit(1)

    output_html = "vdbench_runs_report.html"
    first_section = True
    processed_any = False

    for out_dir in sys.argv[1:]:
        print(f"[INFO] Processing {out_dir} ...")
        try:
            df = read_flatfile_text(out_dir)
        except Exception as e:
            print(f"[WARN] {out_dir}: {e}")
            continue

        processed_any = True
        run_name = str(df["run"].iloc[0]) if "run" in df.columns else "run"

        section = f"<hr><h2 style='font-family:sans-serif'>{out_dir} -{run_name}</h2>\n"
        x = df["interval"]

        # IOPS (rate)
        if "rate" in df.columns and df["rate"].notna().any():
            fig = time_series_fig(x, df["rate"], f"{out_dir}: IOPS (rate)", "IOPS", "rate")
            add_percentile_bands(fig, df["rate"], "IOPS", x)
            section += fig.to_html(full_html=False, include_plotlyjs="cdn" if first_section else False)

        # Throughput (MB/sec)
        if "MB/sec" in df.columns and df["MB/sec"].notna().any():
            fig = time_series_fig(x, df["MB/sec"], f"{out_dir}: Throughput (MB/sec)", "MB/sec",
                ↪ "MB/sec")
            add_percentile_bands(fig, df["MB/sec"], "MB/sec", x)
            section += fig.to_html(full_html=False, include_plotlyjs=False)

        # Latency charts
        for col, label in [("resp", "Latency (resp, ms)"),
                           ("read_resp", "Latency (read_resp, ms)"),
                           ("write_resp", "Latency (write_resp, ms)")]:
            if col in df.columns and df[col].notna().any():
```

```
            fig = time_series_fig(x, df[col], f"{out_dir}: {label}", "ms", col)
            add_percentile_bands(fig, df[col], "lat", x)
            section += fig.to_html(full_html=False, include_plotlyjs=False)

        # CPU used (%)
        if "cpu_used" in df.columns and df["cpu_used"].notna().any():
            fig = time_series_fig(x, df["cpu_used"], f"{out_dir}: CPU used (%)", "%", "cpu_used")
            add_percentile_bands(fig, df["cpu_used"], "cpu", x)
            section += fig.to_html(full_html=False, include_plotlyjs=False)

        # Read %
        if "read%" in df.columns and df["read%"].notna().any():
            fig = time_series_fig(x, df["read%"], f"{out_dir}: Read %", "%", "read%")
            add_percentile_bands(fig, df["read%"], "read%", x)
            section += fig.to_html(full_html=False, include_plotlyjs=False)

        write_or_append(output_html, section, first_section)
        first_section = False

    if not processed_any:
        print("No valid output directories processed.")
        sys.exit(2)

    with open(output_html, "a", encoding="utf-8") as f:
        f.write("</body></html>\n")

    print(f"[OK] Report generated: {output_html}")

if __name__ == "__main__":
    main()
```

Listing A.12: Comparação gráfica de execuções (single vs ativo-ativo)

```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-

"""
VDbench compare runs:
- Parses VDbench flatfile.html (ASCII with columns; handles wrapped lines)
- Builds a single HTML report with OVERLAID time-series comparing multiple runs
- Metrics: IO rate (rate), Latency (resp), CPU used (%), Throughput (MB/sec)
- White background (plotly_white)

Usage:
    python3 vdbench_compare_runs.py out4k out8k ...
"""

import os
import sys
import re
import numpy as np
import pandas as pd
import plotly.graph_objects as go

# Columns commonly present in VDbench flatfile (we keep only the ones we need later)
EXPECTED_COLS = [
    "tod", "date_zone", "run", "interval",
    "reqrate", "rate", "MB/sec", "bytes/io",
    "read%", "resp", "read_resp", "write_resp",
    "resp_max", "read_max", "write_max",
    "resp_std", "read_std", "write_std",
    "xfersize", "threads", "rdpct", "rhpct", "whpct", "seekpct",
    "lunsize_gb", "version", "compratio", "dedupratio",
    "queue_depth", "cpu_used", "cpu_user", "cpu_kernel",
]

def _is_new_record(line: str) -> bool:
    """Detect new data row by timestamp 'HH:MM:SS.xxx' at start of line."""
    return bool(re.match(r"^\d{2}:\d{2}:\d{2}\.\d{3}", line))

def parse_flatfile(out_dir: str) -> pd.DataFrame:
```

```
    """Read and normalize a VDbench flatfile.html as a per-interval DataFrame."""
    ff = os.path.join(out_dir, "flatfile.html")
    if not os.path.exists(ff):
        raise FileNotFoundError(f"{ff} not found")

    # 1) Strip comments and HTML tags
    raw = []
    with open(ff, "r", errors="ignore") as f:
        for ln in f:
            s = ln.rstrip("\n")
            if not s.strip():
                continue
            s2 = s.lstrip()
            if s2.startswith("*") or s2.startswith("<"):
                continue
            raw.append(s.strip())
    if not raw:
        raise ValueError(f"{ff}: no content after removing comments/tags")

    # 2) Join wrapped lines (VDbench can wrap long numeric rows)
    joined = []
    for ln in raw:
        if _is_new_record(ln) or not joined:
            joined.append(ln)
        else:
            joined[-1] = (joined[-1] + " " + ln).strip()

    # 3) Tokenize and map to columns
    rows = []
    for ln in joined:
        toks = ln.split()
        if len(toks) < 10:
            continue
        take = toks[:len(EXPECTED_COLS)]
        if len(take) < len(EXPECTED_COLS):
            take = take + [""] * (len(EXPECTED_COLS) - len(take))
        rows.append(take)
    if not rows:
        raise ValueError(f"{ff}: no valid data rows")

    df = pd.DataFrame(rows, columns=EXPECTED_COLS)

    # Convert needed numerics
    to_num = ["interval","reqrate","rate","MB/sec","resp","cpu_used"]
    for c in to_num:
        if c in df.columns:
            df[c] = pd.to_numeric(df[c], errors="coerce")

    # Ensure interval
    if "interval" in df.columns and df["interval"].notna().any():
        try:
            df["interval"] = df["interval"].astype(int)
        except Exception:
            df["interval"] = np.arange(1, len(df) + 1)
    else:
        df["interval"] = np.arange(1, len(df) + 1)

    # Label
    if "run" not in df.columns or df["run"].isna().all():
        df["run"] = "run"

    # Keep only columns 'well chart
    keep = ["interval","run","reqrate","rate","MB/sec","resp","cpu_used"]
    for k in keep:
        if k not in df.columns:
            df[k] = np.nan
    df = df[keep].copy()
    return df

def add_series(fig: go.Figure, x, y, name):
    fig.add_trace(go.Scatter(
        x=x, y=y, mode="lines+markers", name=name,
```

```
        hovertemplate="%{x}<br>%{y:.4f}<extra></extra>"
    ))

def build_chart(title, x_title, y_title):
    fig = go.Figure()
    fig.update_layout(
        template="plotly_white",
        title=title,
        xaxis_title=x_title,
        yaxis_title=y_title,
        legend_title="Run",
        height=460
    )
    return fig

def main():
    if len(sys.argv) < 2:
        print("Usage: python3 vdbench_compare_runs.py <out_dir1> [out_dir2 ...]")
        sys.exit(1)

    # Parse all dirs
    series = []
    for out_dir in sys.argv[1:]:
        try:
            df = parse_flatfile(out_dir)
        except Exception as e:
            print(f"[WARN] {out_dir}: {e}")
            continue
        label = f"{os.path.basename(os.path.normpath(out_dir))} / {df['run'].iloc[0]}"
        series.append((label, df))

    if not series:
        print("No valid output directories processed.")
        sys.exit(2)

    # Charts: IO rate (rate), Latency (resp), CPU used, Throughput (MB/sec)
    charts = []

    # 1) IO rate (IOPS)
    fig_rate = build_chart("IO rate (IOPS) -per-interval comparison", "Interval", "IOPS")
    for label, df in series:
        if df["rate"].notna().any():
            add_series(fig_rate, df["interval"], df["rate"], label)
    charts.append(fig_rate)

    # (Optional): requested rate overlay (thin dashed) if you want to visualize reqrate too
    # Uncomment to include reqrate for each run
    # for label, df in series:
    # if df["reqrate"].notna().any():
    # fig_rate.add_trace(go.Scatter(
    # x=df["interval"], y=df["reqrate"],
    # mode="lines", name=f"{label} (reqrate)",
    # line=dict(width=1, dash="dot"),
    # hovertemplate="%{x}<br>%{y:.4f}<extra></extra>"
    # ))

    # 2) Latency (resp)
    fig_lat = build_chart("Latency (resp, ms) -per-interval comparison", "Interval", "ms")
    for label, df in series:
        if df["resp"].notna().any():
            add_series(fig_lat, df["interval"], df["resp"], label)
    charts.append(fig_lat)

    # 3) CPU used (%)
    fig_cpu = build_chart("CPU used (%) -per-interval comparison", "Interval", "%")
    for label, df in series:
        if df["cpu_used"].notna().any():
            add_series(fig_cpu, df["interval"], df["cpu_used"], label)
    charts.append(fig_cpu)

    # 4) Throughput (MB/sec)
    fig_mb = build_chart("Throughput (MB/sec) -per-interval comparison", "Interval", "MB/sec")
```

```
    for label, df in series:
        if df["MB/sec"].notna().any():
            add_series(fig_mb, df["interval"], df["MB/sec"], label)
    charts.append(fig_mb)

    # Write one HTML with all charts
    out_html = "vdbench_compare_report.html"
    with open(out_html, "w", encoding="utf-8") as f:
        f.write("<!doctype html><html><head><meta charset='utf-8'><title>VDbench Compare
            ↪ Report</title></head><body>")
        f.write("<h1 style='font-family:sans-serif'>VDbench Compare Report</h1>")
        # Include plotly once (first figure)
        f.write(charts[0].to_html(full_html=False, include_plotlyjs="cdn"))
        # Remaining without JS to avoid duplication
        for fig in charts[1:]:
            f.write(fig.to_html(full_html=False, include_plotlyjs=False))
        f.write("</body></html>")

    print(f"[OK] Report generated: {out_html}")

if __name__ == "__main__":
    main()
```

Listing A.13: Script de integração com Graphite/coletores

```
#!/bin/bash

# vdbench_graphite.sh is a bash script to feed parsed vdbench output to graphite
#
# Requires netcat (nc), tee, awk, graphite, and vdbench
# http://www.oracle.com/technetwork/server-storage/vdbench-downloads-1901681.html
#
# To execute, simply run vdbench and pipe to this script. By default it will send stats to graphite
    ↪ localhost:2003
# vdbench.default.[HOSTNAME].[METRIC] There are no required options, but additional parameters can
    ↪ be passed to
# change the destination host and/or port as well as a custom tag instead of using the hostname. To
    ↪ implement multiple
# custom tags simply separate the tags with periods.
#
# It is also possible to feed the stats from an existing vdbench workload provided stdout was
    ↪ written to a file.
# Simply cat the vdbench output file and redirect to vdbench_graphite.sh the same as above. Note:
    ↪ original timestamps
# are used for the metrics so it will be necessary to look at the graph historically.
#
# ./vdbench -f foo.vdb -o outdir.foo | vdbench_graphite.sh -h [graphite_host] -p [graphite_port] -t
    ↪ [graphite_tag] -o console.foo


while [[ $# > 1 ]]
do
key="$1"

case $key in
    -h|--host)
    graphite_host="$2"
    shift
    ;;
    -p|--port)
    graphite_port="$2"
    shift
    ;;
    -t|--tag)
    tag="$2"
    shift
    ;;
    -o|--outfile)
    outfile="$2"
    ;;
    *)
```

```
41
42      ;;
43  esac
44  shift
45  done
46
47  echo
48
49  # Assume graphite is running on localhost if not defined
50  if [ -z "$graphite_host" ] ; then
51      echo "Graphite host \"-h\" not defined, assuming localhost."
52      graphite_host="localhost"
53  fi
54
55  # Addume graphite is running on port 2003 if not defined
56  if [ -z "$graphite_port" ] ; then
57      echo "Graphite port \"-p\" not defined, assuming 2003."
58      graphite_port="2003"
59  fi
60
61  # Check to make sure we can talk to graphite before continuing
62  #nc -z $graphite_host $graphite_port
63
64  #if [ "$?" != "0" ]; then
65  # echo "Can't communicate with graphite at $graphite_host on TCP port $graphite_port."
66  # exit
67  #fi
68
69  # Assign a prefix based on tag
70  if [ -z "$tag" ]; then
71      prefix=vdbench.default.`hostname -s`
72  else
73      prefix=vdbench.$tag.`hostname -s`
74  fi
75
76  # If not outfile specified, kick overything to /dev/null
77  if [ -z "$outfile" ]; then
78      outfile="/dev/null"
79  fi
80
81  echo "Sending metrics to graphite $graphite_host $graphite_port at $prefix."
82
83  # Write a copy of everything coming in to another file
84  tee $outfile |
85
86  # Begin really awkward text parsing
87  awk -v prefix=$prefix '
88  BEGIN {
89  }
90
91  # Search for keyword "interval " to define the first title header
92  / interval / {
93      # Use gsub to remove some unwanted characters
94      gsub(/sec/,"",$0)
95      gsub(/\//,"",$0)
96      gsub(/,/,"",$0)
97
98      # Feed all the titles into an array
99      title1_columns=split($0,title1_array," ")
100
101     # Vdbench uses a string for the month, this converts it to a number and assigns it a variable
102     month=(match("JanFebMarAprMayJunJulAugSepOctNovDec",title1_array[1])+2)/3
103
104     # Assign day and year to variables
105     day=title1_array[2]
106     year=title1_array[3]
107 }
108
109 # Search for keyword "rate" ot define the second title header
110 /^\s*rate/ {
111     # Replace 1024**2 with sec so we can mash it from a header from the above section to get MB/sec
112     gsub(/1024\*\*2/,"sec",$0)
```

```
113
114     # Use gsub to remove an unwanted /
115     gsub(/\//,"",$0)
116
117     # Feed all the second titles into an array
118     title2_columns=split($0,title2_array," ")
119
120     # Only get some of them because of vdbench formatting
121     title1_column=4
122     title2_column=0
123     title_column=1
124     while (title1_column <= title1_columns) {
125         title_array[title_column]=title1_array[title1_column] title2_array[title2_column]
126
127         title1_column++
128         title2_column++
129         title_column++
130     }
131 }
132
133 # Search for any lines that do not have letters, assumed to be vdbench metrics
134 !/[a-z]/ {
135     # Simple check to make sure that we have the month (and presumable day and year) before continuing
136     if ( title1_array[1] != "" ) {
137         # Feed all the metrics into an array
138         stats_columns=split($0,stats_array," ")
139         sub(/\..*/,"",stats_array[1])
140
141         split(stats_array[1],timestamp,":")
142         hour=timestamp[1]
143         min=timestamp[2]
144         sec=timestamp[3]
145
146         # Convert time to epoch time
147         epochtime=mktime(year " " month " " day " " hour " " min " " sec)
148
149         title_column=1
150         stats_column=2
151
152         # Do stuff and things
153         while (stats_column <= stats_columns) {
154
155
156       # Spit out the formatted metrics in graphite friendly format
157             print prefix"."title_array[title_column] " " stats_array[stats_column] " " epochtime
158             title_column++
159             stats_column++
160         }
161     }
162 }
163
164 END {
165 }
166 ' | (nc -q 1 $graphite_host $graphite_port)
167
168 # Wait until previous commands finish before continuing
169 wait
170
171 echo
172 echo "Done"
```

A.2 Scripts para Automação e Verificação do Estado dos Equipamentos

Os scripts deste grupo implementam autenticação via REST, recolha de métricas e alarmes, criação de relatórios e distribuição automática por email, conforme descrito na Secção 4.5.

Listing A.14: Monitorização diária e envio de relatório HTML por email

```
#!/usr/bin/env python3
import requests
import json
import sys
import os
import smtplib
from email.mime.multipart import MIMEMultipart
from email.mime.text import MIMEText
from email.mime.base import MIMEBase
from email import encoders
from datetime import datetime
from requests.packages.urllib3.exceptions import InsecureRequestWarning

# Desabilitar warnings de SSL
requests.packages.urllib3.disable_warnings(InsecureRequestWarning)

def get_huawei_password():
    """Lê a senha do storage Huawei do ficheiro"""
    try:
        with open('huawei_passwd', 'r') as f:
            return f.read().strip()
    except FileNotFoundError:
        print("Erro: Ficheiro huawei_passwd não encontrado")
        sys.exit(1)

def get_smtp_password():
    """Lê a senha do SMTP do ficheiro"""
    try:
        with open('smtp_passwd', 'r') as f:
            return f.read().strip()
    except FileNotFoundError:
        print("Erro: Ficheiro smtp_passwd não encontrado")
        sys.exit(1)

def get_storage_data_final(storage_name, ip):

    # Configurações
    user_huawei = "monituser"
    password = get_huawei_password()
    port = "8088"
    sector_factor = 104965.12 / 220128240911

    base_url = f"https://{ip}:{port}/deviceManager/rest"
    session_url = f"{base_url}/xxxxx/sessions"

    headers = {
        "Content-Type": "application/json",
        "Accept": "application/json"
    }

    body = {
        "username": user_huawei,
        "password": password,
        "scope": "0"
    }

    print(f"  A obter dados do storage: {storage_name} ({ip})")

    try:
```

```
60          # Fazer login
61          session = requests.Session()
62          session.verify = False
63
64          response = session.post(session_url, headers=headers, json=body, timeout=30)
65
66          if response.status_code == 200:
67              login_data = response.json()
68              if 'data' in login_data and login_data['data']:
69                  device_id = login_data['data']['deviceid']
70                  iBaseToken = login_data['data']['iBaseToken']
71
72                  print(" Login bem-sucedido")
73
74                  # Headers com token para as próximas requisições
75                  auth_headers = {
76                      "Content-Type": "application/json",
77                      "Accept": "application/json",
78                      "iBaseToken": iBaseToken
79                  }
80
81                  storage_info = {
82                      "Name": storage_name,
83                      "IP": ip,
84                      "Serial": device_id,
85                      "Status": "UNKNOWN",
86                      "Alarmes": "UNKNOWN",
87                      "Pool_Capacity_TB": "N/A",
88                      "Pool_Used_Capacity_TB": "N/A",
89                      "Pool_Free_TB": "N/A",
90                      "Pool_Used_Percent": "N/A",
91                      "Alarm_Count": 0,
92                      "Alarm_Details": "N/A"
93                  }
94
95                  # 1. OBTER INFORMAÇÕES DO SISTEMA
96                  system_url = f"{base_url}/{device_id}/system/"
97                  print(f" A recolher informações do sistema...")
98
99                  system_response = session.get(system_url, headers=auth_headers, timeout=30)
100                 if system_response.status_code == 200:
101                     system_data = system_response.json()
102                     if 'data' in system_data and system_data['data']:
103                         system_info = system_data['data']
104
105                         # Extrair status de saúde
106                         if 'HEALTHSTATUS' in system_info:
107                             storage_info['Status'] = "OK" if str(system_info['HEALTHSTATUS']) == "1"
                                    ↪ else "ERROR"
108
109                         # Calcular capacidades
110                         if 'STORAGEPOOLFREECAPACITY' in system_info and 'STORAGEPOOLCAPACITY' in
                                ↪ system_info:
111                             try:
112                                 pool_free = (float(system_info['STORAGEPOOLFREECAPACITY']) *
                                        ↪ sector_factor) / 1024
113                                 pool_capacity = (float(system_info['STORAGEPOOLCAPACITY']) *
                                        ↪ sector_factor) / 1024
114                                 pool_used = pool_capacity - pool_free
115                                 pool_used_percent = (pool_used / pool_capacity) * 100
116
117                                 storage_info['Pool_Capacity_TB'] = round(pool_capacity, 2)
118                                 storage_info['Pool_Used_Capacity_TB'] = round(pool_used, 2)
119                                 storage_info['Pool_Free_TB'] = round(pool_free, 2)
120                                 storage_info['Pool_Used_Percent'] = f"{pool_used_percent:.2f}%"
121                             except (ValueError, TypeError) as e:
122                                 print(f" Erro no cálculo de capacidades: {e}")
123
124                 # 2. OBTER CONTAGEM DE ALARMES
125                 alarm_count_url = f"{base_url}/{device_id}/alarm/currentalarm/count"
126                 print(f" A recolher contagem de alarmes...")
127
```

```
            alarm_count_response = session.get(alarm_count_url, headers=auth_headers, timeout=30)
            if alarm_count_response.status_code == 200:
                alarm_count_data = alarm_count_response.json()
                if 'data' in alarm_count_data and alarm_count_data['data']:
                    # O campo correto é "COUNT" (em maiúsculas)
                    alarm_count = alarm_count_data['data'].get('COUNT', 0)
                    storage_info['Alarm_Count'] = int(alarm_count)
                    storage_info['Alarmes'] = "ERROR" if storage_info['Alarm_Count'] > 0 else "OK"

                    print(f"  Contagem de alarmes: {storage_info['Alarm_Count']}")

            # 3. SE HOUVER ALARMES, OBTER DETALHES
            if storage_info['Alarm_Count'] > 0:
                alarm_detail_url = f"{base_url}/{device_id}/alarm/currentalarm"
                print(f"  A recolher detalhes dos {storage_info['Alarm_Count']} alarmes...")

                alarm_detail_response = session.get(alarm_detail_url, headers=auth_headers,
                    ↪ timeout=30)
                if alarm_detail_response.status_code == 200:
                    alarm_detail_data = alarm_detail_response.json()
                    if 'data' in alarm_detail_data and alarm_detail_data['data']:
                        alarms = alarm_detail_data['data']

                        # Recolher detalhes dos primeiros 3 alarmes (os mais importantes)
                        alarm_descriptions = []
                        for i, alarm in enumerate(alarms[:3]): # Limitar a 3 alarmes
                            description = alarm.get('description', 'Sem descrição')
                            level = alarm.get('level', 'N/A')
                            name = alarm.get('name', 'Sem nome')

                            # Criar descrição resumida
                            alarm_text = f"{name} (Nível: {level})"
                            alarm_descriptions.append(alarm_text)

                        if alarm_descriptions:
                            storage_info['Alarm_Details'] = " | ".join(alarm_descriptions)
                        else:
                            storage_info['Alarm_Details'] = f"{storage_info['Alarm_Count']} alarme(s)
                                ↪ ativo(s)"
            else:
                storage_info['Alarm_Details'] = "Nenhum alarme ativo"

            # 4. OBTER INFORMAÇÕES DOS POOLS
            if storage_info['Pool_Capacity_TB'] == "N/A":
                pool_url = f"{base_url}/{device_id}/storagepool"
                print(f"  A recolher informações dos pools...")

                try:
                    pool_response = session.get(pool_url, headers=auth_headers, timeout=30)
                    if pool_response.status_code == 200:
                        pool_data = pool_response.json()
                        if 'data' in pool_data and pool_data['data']:
                            total_capacity = 0
                            total_used = 0
                            total_free = 0

                            for pool in pool_data['data']:
                                try:
                                    if 'USERTOTALCAPACITY' in pool:
                                        total_capacity += float(pool['USERTOTALCAPACITY']) *
                                            ↪ sector_factor / 1024
                                    if 'USERCONSUMEDCAPACITY' in pool:
                                        total_used += float(pool['USERCONSUMEDCAPACITY']) *
                                            ↪ sector_factor / 1024
                                    if 'USERFREECAPACITY' in pool:
                                        total_free += float(pool['USERFREECAPACITY']) * sector_factor
                                            ↪ / 1024
                                except (ValueError, TypeError):
                                    continue

                            if total_capacity > 0:
                                storage_info['Pool_Capacity_TB'] = round(total_capacity, 2)
```

```
                                storage_info['Pool_Used_Capacity_TB'] = round(total_used, 2)
                                storage_info['Pool_Free_TB'] = round(total_free, 2)
                                storage_info['Pool_Used_Percent'] =
                                    ↪ f"{(total_used/total_capacity):.2%}"
                    except Exception as e:
                        print(f"  Erro ao obter pools: {e}")

                print(f"  Dados obtidos para {storage_name}")
                print(f" Status: {storage_info['Status']}, Alarmes: {storage_info['Alarmes']}, Count:
                    ↪ {storage_info['Alarm_Count']}")
                return storage_info

        print("  Falha no login")
        return None

    except Exception as e:
        print(f"  Erro ao obter dados: {str(e)}")
        import traceback
        traceback.print_exc()
        return None

def generate_html_report_final(storage_data_list):
    """Gera relatório HTML final"""

    html_head = """
    <!DOCTYPE html>
    <html>
    <head>
        <title>Health Check Final - Storages</title>
        <style>
            body {
                font-family: Arial, sans-serif;
                margin: 20px;
                background-color: #f5f5f5;
            }
            .container {
                max-width: 1400px;
                margin: 0 auto;
                background: white;
                padding: 20px;
                border-radius: 10px;
                box-shadow: 0 2px 10px rgba(0,0,0,0.1);
            }
            h1 {
                color: #2c3e50;
                text-align: center;
                margin-bottom: 30px;
            }
            table {
                width: 100%;
                border-collapse: collapse;
                margin: 20px 0;
                font-size: 14px;
            }
            th, td {
                border: 1px solid #ddd;
                padding: 10px;
                text-align: left;
                vertical-align: top;
            }
            th {
                background-color: #3498db;
                color: white;
                font-weight: bold;
            }
            tr:nth-child(even) {
                background-color: #f8f9fa;
            }
            tr:hover {
                background-color: #e8f4f8;
            }
            .status-ok {
```

```
            background-color: #d4edda;
            color: #155724;
            font-weight: bold;
            text-align: center;
        }
        .status-error {
            background-color: #f8d7da;
            color: #721c24;
            font-weight: bold;
            text-align: center;
        }
        .status-unknown {
            background-color: #fff3cd;
            color: #856404;
            font-weight: bold;
            text-align: center;
        }
        .summary {
            background-color: #e7f3ff;
            padding: 15px;
            border-radius: 5px;
            margin: 20px 0;
            border-left: 5px solid #3498db;
        }
        .timestamp {
            text-align: center;
            color: #666;
            font-style: italic;
            margin-bottom: 20px;
        }
        .alarm-details {
            max-width: 300px;
            word-wrap: break-word;
            font-size: 12px;
            line-height: 1.4;
        }
        .critical {
            background-color: #ffcccc;
            font-weight: bold;
        }
        .warning {
            background-color: #fff3cd;
        }
        .alarm-count-high {
            color: #dc3545;
            font-weight: bold;
            font-size: 16px;
        }
    </style>
</head>
<body>
    <div class="container">
        <h1 > Health Check Final - Storages</h1>
"""

# Resumo
total_storages = len(storage_data_list)
ok_storages = len([s for s in storage_data_list if s.get('Status') == 'OK'])
error_storages = len([s for s in storage_data_list if s.get('Status') == 'ERROR'])
unknown_storages = len([s for s in storage_data_list if s.get('Status') not in ['OK', 'ERROR']])

total_alarms = sum([s.get('Alarm_Count', 0) for s in storage_data_list])

current_time = datetime.now().strftime("%d/%m/%Y %H:%M:%S")

html_body = html_head + f"""
        <div class="timestamp" > Relatório gerado em: {current_time}</div>

        <div class="summary">
            <h3 > Resumo do Sistema</h3>
            <p><strong>Total de Storages:</strong> {total_storages}</p>
            <p><strong > Status OK:</strong> {ok_storages}</p>
```

```
            <p><strong > Status ERROR:</strong> {error_storages}</p>
            <p><strong > Status UNKNOWN:</strong> {unknown_storages}</p>
            <p><strong > Total de Alarmes Ativos:</strong> <span
                ↪ class="alarm-count-high">{total_alarms}</span></p>
        </div>

        <table>
            <thead>
                <tr>
                    <th>Storage Name</th>
                    <th>Serial Number</th>
                    <th>Pool Capacity (TB)</th>
                    <th>Pool Used (TB)</th>
                    <th>Pool Free (TB)</th>
                    <th>Pool Used %</th>
                    <th>Status</th>
                    <th>Alarm Status</th>
                    <th>Alarm Count</th>
                    <th>Alarm Details</th>
                </tr>
            </thead>
            <tbody>
    """

    # Adicionar linhas da tabela
    for storage in storage_data_list:
        status_class = f"status-{storage.get('Status', 'unknown').lower()}"
        alarm_class = f"status-{storage.get('Alarmes', 'unknown').lower()}"

        # Destacar linhas com alarmes
        row_class = ""
        if storage.get('Alarm_Count', 0) > 0:
            row_class = "critical"
        elif storage.get('Status') == 'ERROR':
            row_class = "warning"

        # Destacar contagem de alarmes altas
        alarm_count_display = storage.get('Alarm_Count', 'N/A')
        if isinstance(alarm_count_display, int) and alarm_count_display > 0:
            alarm_count_display = f'<span class="alarm-count-high">{alarm_count_display}</span>'

        html_body += f"""
                <tr class="{row_class}">
                    <td><strong>{storage.get('Name', 'N/A')}</strong><br/><small>IP:
                        ↪ {storage.get('IP', 'N/A')}</small></td>
                    <td>{storage.get('Serial', 'N/A')}</td>
                    <td>{storage.get('Pool_Capacity_TB', 'N/A')}</td>
                    <td>{storage.get('Pool_Used_Capacity_TB', 'N/A')}</td>
                    <td>{storage.get('Pool_Free_TB', 'N/A')}</td>
                    <td>{storage.get('Pool_Used_Percent', 'N/A')}</td>
                    <td class="{status_class}">{storage.get('Status', 'N/A')}</td>
                    <td class="{alarm_class}">{storage.get('Alarmes', 'N/A')}</td>
                    <td>{alarm_count_display}</td>
                    <td class="alarm-details">{storage.get('Alarm_Details', 'N/A')}</td>
                </tr>
        """

    html_body += """
            </tbody>
        </table>

        <div class="summary">
            <h3 > Legenda</h3>
            <p><span class="status-ok">OK</span> - Sistema operacional normal</p>
            <p><span class="status-error">ERROR</span> - Problema detectado no sistema</p>
            <p><span class="status-unknown">UNKNOWN</span> - Status não pôde ser determinado</p>
            <p><span class="critical" ></span> - Storage com alarmes ativos</p>
            <p><span class="warning" ></span> - Storage com status ERROR</p>
            <p><span class="alarm-count-high">Número vermelho</span> - Contagem de alarmes
                ↪ crítica</p>
        </div>
    </div>
```

```
406     </body>
407     </html>
408     """
409 
410     # Escrever ficheiro HTML
411     with open("storage_report_final.html", "w", encoding="utf-8") as f:
412         f.write(html_body)
413 
414     print(f"  Relatório HTML final gerado: storage_report_final.html")
415     return html_body
416 
417 def send_email_final(html_content):
418     """Envia o relatório por email - Versão Final"""
419 
420     # Configurações do email
421     smtp_server = "mail.a2o3.com"
422     smtp_port = 465 # SSL
423     username = "mei@a2o3.com"
424     password = get_smtp_password()
425     from_addr = "mei@a2o3.com"
426     to_addr = "aluno24391@ipt.pt"
427 
428     current_date = datetime.now().strftime("%d/%B/%Y")
429     subject = f"Health Check - {current_date}"
430 
431     print(f"  A preparar envio de email final...")
432     print(f" From: {from_addr}")
433     print(f" To: {to_addr}")
434     print(f" SMTP: {smtp_server}:{smtp_port}")
435     print(f" User: {username}")
436     print(f" Password: {'*' * len(password)}") # Mostra apenas asteriscos
437 
438     try:
439         # Criar mensagem
440         msg = MIMEMultipart()
441         msg['From'] = from_addr
442         msg['To'] = to_addr
443         msg['Subject'] = subject
444 
445         # Corpo do email em HTML
446         msg.attach(MIMEText(html_content, 'html'))
447 
448         # Anexar o arquivo HTML
449         with open("storage_report_final.html", "rb") as attachment:
450             part = MIMEBase('application', 'octet-stream')
451             part.set_payload(attachment.read())
452 
453         encoders.encode_base64(part)
454         part.add_header(
455             'Content-Disposition',
456             'attachment; filename="storage_health_report_final.html"'
457         )
458         msg.attach(part)
459 
460         # Conectar e enviar email
461         print("  A conectar ao servidor SMTP...")
462         server = smtplib.SMTP_SSL(smtp_server, smtp_port)
463 
464         print("  A efetuar login...")
465         server.login(username, password)
466 
467         print("  A enviar email...")
468         server.sendmail(from_addr, to_addr, msg.as_string())
469         server.quit()
470 
471         print("   EMAIL FINAL ENVIADO COM SUCESSO!   ")
472         return True
473 
474     except Exception as e:
475         print(f"   ERRO AO ENVIAR EMAIL: {str(e)}   ")
476         import traceback
477         traceback.print_exc()
```

```
        return False

def main_final():
    """Função principal - Versão Final"""

    print("  A iniciar Health Check")
    print("=" * 60)

    # Lista de storages para verificar
    storages = [
        {"name": "STG A", "ip": "10.10.10.27"},
        {"name": "STG B", "ip": "10.10.10.24"}
    ]

    all_storage_data = []

    # Coletar dados de cada storage
    for storage in storages:
        print(f"\n{'='*50}")
        data = get_storage_data_final(storage["name"], storage["ip"])
        if data:
            all_storage_data.append(data)
        print(f"{'='*50}")

    if all_storage_data:
        # Gerar relatório HTML
        print(f"\ n A gerar relatório final com {len(all_storage_data)} storages...")
        html_content = generate_html_report_final(all_storage_data)

        # Calcular estatísticas dinâmicas
        total_alarms = sum([s.get('Alarm_Count', 0) for s in all_storage_data])
        storages_with_alarms = [s for s in all_storage_data if s.get('Alarm_Count', 0) > 0]

        # Enviar email
        print(f"\ n A enviar relatório final por email...")
        email_sent = send_email_final(html_content)

        # MENSAGEM FINAL DINÂMICA
        success_msg = f"\ n  PROCESSO FINAL CONCLUÍDO COM SUCESSO!   "
        success_msg += f"\n  Relatório: storage_report_final.html"
        success_msg += f"\n  Email enviado para: aluno24391@ipt.pt"

        if total_alarms > 0:
            success_msg += f"\n  ALARMES DETETADOS:"
            for storage in storages_with_alarms:
                alarm_count = storage.get('Alarm_Count', 0)
                success_msg += f"\n •{storage['Name']}: {alarm_count} alarme(s) ativo(s)"
            success_msg += f"\n  Total de alarmes no sistema: {total_alarms}"
        else:
            success_msg += f"\n  Nenhum alarme ativo detetado no sistema"

        if email_sent:
            print(success_msg)
        else:
            print(f"\ n Relatório final gerado mas email não enviado")
            print(f"  Relatório disponível em: storage_report_final.html")
            # Mostrar resumo dos alarmes mesmo se o email falhar
            if total_alarms > 0:
                print(f"  ALARMES DETETADOS (relatório local):")
                for storage in storages_with_alarms:
                    alarm_count = storage.get('Alarm_Count', 0)
                    print(f" •{storage['Name']}: {alarm_count} alarme(s) ativo(s)")
    else:
        print("\ n Nenhum dado foi obtido dos storages")

if __name__ == "__main__":
    main_final()
```

Listing A.15: Utilitários de depuração de alarmes (endpoints e payloads)

```
#!/usr/bin/env python3
```

```
import requests
import json
import sys
from requests.packages.urllib3.exceptions import InsecureRequestWarning

requests.packages.urllib3.disable_warnings(InsecureRequestWarning)

def get_password():
    try:
        with open('huawei_passwd', 'r') as f:
            return f.read().strip()
    except FileNotFoundError:
        print("Erro: Ficheiro huawei_passwd não encontrado")
        sys.exit(1)

def debug_alarms(ip, storage_name):
    user_huawei = "monituser"
    password = get_password()
    port = "8088"

    base_url = f"https://{ip}:{port}/deviceManager/rest"
    session_url = f"{base_url}/xxxxx/sessions"

    headers = {"Content-Type": "application/json", "Accept": "application/json"}
    body = {"username": user_huawei, "password": password, "scope": "0"}

    print(f"  Debug de alarmes para: {storage_name} ({ip})")

    try:
        session = requests.Session()
        session.verify = False

        response = session.post(session_url, headers=headers, json=body, timeout=30)

        if response.status_code == 200:
            login_data = response.json()
            if 'data' in login_data and login_data['data']:
                device_id = login_data['data']['deviceid']
                iBaseToken = login_data['data']['iBaseToken']

                print(" Login bem-sucedido")
                print(f"  Device ID: {device_id}")

                auth_headers = {
                    "Content-Type": "application/json",
                    "Accept": "application/json",
                    "iBaseToken": iBaseToken
                }

                # Testar TODOS os endpoints possíveis de alarmes
                alarm_endpoints = [
                    f"/{device_id}/alarm/currentalarm/count",
                    f"/{device_id}/alarm/currentalarm",
                    f"/{device_id}/alarm/current_alarm",
                    f"/{device_id}/alarm/current_alarm/count",
                    f"/{device_id}/alarm",
                    f"/{device_id}/alarm/count",
                    f"/{device_id}/alarms",
                    f"/{device_id}/alarms/count",
                    f"/{device_id}/fault",
                    f"/{device_id}/fault/current",
                    f"/{device_id}/fault/current/count",
                ]

                for endpoint in alarm_endpoints:
                    url = f"{base_url}{endpoint}"
                    print(f"\ n Testando: {endpoint}")

                    try:
                        response = session.get(url, headers=auth_headers, timeout=15)
                        print(f" Status: {response.status_code}")

```

```
                    if response.status_code == 200:
                        data = response.json()
                        print(f"  Resposta: {json.dumps(data, indent=2)}")
                    else:
                        print(f"  Erro: {response.text[:200]}")

                except Exception as e:
                    print(f"  Exception: {str(e)}")

    except Exception as e:
        print(f"  Erro geral: {str(e)}")
        import traceback
        traceback.print_exc()

if __name__ == "__main__":
    if len(sys.argv) != 3:
        print("Uso: python3 debug_alarms.py <storage_name> <ip>")
        sys.exit(1)

    debug_alarms(sys.argv[2], sys.argv[1])
```

Listing A.16: Testes de conectividade/variação de endpoints REST

```
#!/usr/bin/env python3
import requests
import json
import sys
from requests.packages.urllib3.exceptions import InsecureRequestWarning
requests.packages.urllib3.disable_warnings(InsecureRequestWarning)

def debug_storage(ip, storage_name):
    user_huawei = "monituser"
    password = "Huawei12#$" # Coloque sua senha real
    port = "8088"

    base_url = f"https://{ip}:{port}/deviceManager/rest"
    session_url = f"{base_url}/xxxxx/sessions"

    headers = {"Content-Type": "application/json", "Accept": "application/json"}
    body = {"username": user_huawei, "password": password, "scope": "0"}

    try:
        response = requests.post(session_url, headers=headers, json=body, verify=False, timeout=30)
        print(f"Status Login: {response.status_code}")

        if response.status_code == 200:
            data = response.json()
            print(f"Resposta Login: {json.dumps(data, indent=2)}")

            serial_number = data['data']['deviceid']
            baseid = data['data']['iBaseToken']

            # Verificar sistema
            system_url = f"{base_url}/{serial_number}/system/"
            auth_header = {"iBaseToken": baseid}

            system_response = requests.get(system_url, headers=auth_header, verify=False, timeout=30)
            print(f"Status Sistema: {system_response.status_code}")
            system_data = system_response.json()
            print(f"Dados Sistema: {json.dumps(system_data, indent=2)}")

        else:
            print(f"Erro no login: {response.text}")

    except Exception as e:
        print(f"Erro: {str(e)}")

if __name__ == "__main__":
    if len(sys.argv) != 3:
        print("Uso: python3 debug_api.py <storage_name> <ip>")
        sys.exit(1)
```

```
49
50      debug_storage(sys.argv[2], sys.argv[1])
```

Listing A.17: Gerador de relatórios HTML (formatação e síntese de dados)

```
1   #!/usr/bin/env python3
2   import json
3   import os
4   import smtplib
5   from email.mime.multipart import MIMEMultipart
6   from email.mime.text import MIMEText
7   from email.mime.base import MIMEBase
8   from email import encoders
9   import glob
10  from datetime import datetime
11
12  def generate_html_reports():
13      # Coletar todos os arquivos temporários
14      temp_files = glob.glob("temp_*.json")
15
16      storage_data = []
17      for temp_file in temp_files:
18          try:
19              with open(temp_file, 'r') as f:
20                  data = json.load(f)
21                  storage_data.append(data)
22          except Exception as e:
23              print(f"Erro ao ler {temp_file}: {str(e)}")
24
25      # Gerar HTML
26      html_head = """
27      <style>
28          body { font-family: Arial, sans-serif; margin: 20px; }
29          h2 { color: #4CAF50; }
30          table { width: 100%; border-collapse: collapse; margin: 20px 0; }
31          th, td { border: 1px solid #ddd; padding: 12px; text-align: left; }
32          th { background-color: #4CAF50; color: white; }
33          tr:nth-child(even) { background-color: #f2f2f2; }
34          tr:hover { background-color: #ddd; }
35          tr.highlight { background-color: #ffcccc; }
36          .summary { background-color: #e7f3ff; padding: 15px; border-radius: 5px; margin: 20px 0; }
37      </style>
38      """
39
40      # HTML para storage info
41      if storage_data:
42          # Resumo
43          total_storages = len(storage_data)
44          error_storages = len([s for s in storage_data if s['Status'] == 'ERROR' or s['Alarmes'] ==
                ↪ 'ERROR'])
45
46          html_body = f"<html><head>{html_head}</head><body>"
47          html_body += f"<h2>Health Check Huawei</h2>"
48          html_body += f"<div class='summary'>"
49          html_body += f"<strong>Resumo:</strong> {total_storages} storages verificados,
                ↪ {error_storages} com problemas"
50          html_body += f"</div>"
51          html_body += "<table>"
52
53          # Cabeçalho da tabela
54          headers = ["Name", "Modelo", "Version", "Serial", "Pool_Capacity_TB", "Pool_Used_Capacity_TB",
55                   "Pool_Free_TB", "Pool_Used_Percent", "Alarmes", "Status", "IP"]
56          html_body += "<tr>" + "".join(f"<th>{header}</th>" for header in headers) + "</tr>"
57
58          for storage in storage_data:
59              row_class = "highlight" if storage['Status'] == "ERROR" or storage['Alarmes'] == "ERROR"
                    ↪ else ""
60              html_body += f"<tr class='{row_class}'>"
61              for header in headers:
62                  value = storage.get(header, 'N/A')
63                  html_body += f"<td>{value}</td>"
```

```
64            html_body += "</tr>"
65
66        html_body += "</table></body></html>"
67
68        with open("inventory_huawei.html", "w") as f:
69            f.write(html_body)
70        print(f"Relatório HTML gerado com {len(storage_data)} storages")
71
72    else:
73        print("Nenhum dado de storage encontrado para gerar relatório")
74        # Criar relatório vazio
75        html_body = f"<html><head>{html_head}</head><body>"
76        html_body += "<h2>Health Check Huawei</h2>"
77        html_body += "<p>Nenhum storage foi verificado com sucesso.</p>"
78        html_body += "<p>Verifique os logs para mais informações.</p>"
79        html_body += "</body></html>"
80
81        with open("inventory_huawei.html", "w") as f:
82            f.write(html_body)
83
84    # Limpar arquivos temporários
85    for temp_file in temp_files:
86        try:
87            os.remove(temp_file)
88        except:
89            pass
90
91 def send_email():
92    # Configurações do email CORRIGIDAS
93    smtp_server = "mail.a2o3.com"
94    smtp_port = 465 # SSL
95    from_addr = "mei@a2o3.com"
96    to_addr = "aluno24391@ipt.pt"
97    username = "mei@a2o3.com"
98    password = "meiPassword1"
99
100   date = datetime.now().strftime("%d/%B/%Y")
101   subject = f"Daily Health Check - HUAWEI - {date}"
102
103   # Criar mensagem
104   msg = MIMEMultipart()
105   msg['From'] = from_addr
106   msg['To'] = to_addr
107   msg['Subject'] = subject
108
109   # Corpo do email
110   body = """
111   <html>
112       <body>
113           <h2>Daily Health Check - Storages Huawei</h2>
114           <p>Relatório gerado automaticamente em """ + datetime.now().strftime("%d/%m/%Y %H:%M:%S")
                  ↪ + """</p>
115           <p>Verifique o anexo para detalhes completos.</p>
116       </body>
117   </html>
118   """
119   msg.attach(MIMEText(body, 'html'))
120
121   # Anexar arquivo HTML
122   if os.path.exists("inventory_huawei.html"):
123       with open("inventory_huawei.html", "rb") as attachment:
124           part = MIMEBase('application', 'octet-stream')
125           part.set_payload(attachment.read())
126
127       encoders.encode_base64(part)
128       part.add_header(
129           'Content-Disposition',
130           f'attachment; filename=inventory_huawei.html'
131       )
132       msg.attach(part)
133
134   # Enviar email
```

```
    try:
        print("Conectando ao servidor SMTP...")
        server = smtplib.SMTP_SSL(smtp_server, smtp_port)
        print("Efetuando login...")
        server.login(username, password)

        text = msg.as_string()
        print("Enviando email...")
        server.sendmail(from_addr, to_addr, text)
        server.quit()
        print("Email enviado com sucesso!")
    except Exception as e:
        print(f"Erro ao enviar email: {str(e)}")
        import traceback
        traceback.print_exc()

if __name__ == "__main__":
    generate_html_reports()
    send_email()
```

Listing A.18: Cliente REST para recolha de métricas (capacidade, saúde, pools)

```
#!/usr/bin/env python3
import requests
import json
import sys
import os
from requests.packages.urllib3.exceptions import InsecureRequestWarning

# Desabilitar warnings de SSL
requests.packages.urllib3.disable_warnings(InsecureRequestWarning)

def get_password():
    """Lê a senha do arquivo"""
    try:
        with open('huawei_passwd', 'r') as f:
            return f.read().strip()
    except FileNotFoundError:
        print("Erro: Arquivo huawei_passwd não encontrado")
        sys.exit(1)

def get_storage_data(storage_name, ip):
    # Configurações
    user_huawei = "monituser"
    password = get_password()
    port = "8088"
    sector_factor = 104965.12 / 220128240911

    base_url = f"https://{ip}:{port}/deviceManager/rest"
    session_url = f"{base_url}/xxxxx/sessions"

    headers = {
        "Content-Type": "application/json",
        "Accept": "application/json"
    }

    body = {
        "username": user_huawei,
        "password": password,
        "scope": "0"
    }

    print(f"  Obtendo dados do storage: {storage_name} ({ip})")

    try:
        # Fazer login
        session = requests.Session()
        session.verify = False

        response = session.post(session_url, headers=headers, json=body, timeout=30)

```

```
50          if response.status_code == 200:
51              login_data = response.json()
52              if 'data' in login_data and login_data['data']:
53                  device_id = login_data['data']['deviceid']
54                  iBaseToken = login_data['data']['iBaseToken']
55
56                  print(" Login bem-sucedido")
57
58                  # Headers com token para as próximas requisições
59                  auth_headers = {
60                      "Content-Type": "application/json",
61                      "Accept": "application/json",
62                      "iBaseToken": iBaseToken
63                  }
64
65                  storage_info = {
66                      "Name": storage_name,
67                      "IP": ip,
68                      "Serial": device_id,
69                      "Status": "UNKNOWN",
70                      "Alarmes": "UNKNOWN",
71                      "Modelo": "N/A",
72                      "Version": "N/A",
73                      "Pool_Capacity_TB": "N/A",
74                      "Pool_Used_Capacity_TB": "N/A",
75                      "Pool_Free_TB": "N/A",
76                      "Pool_Used_Percent": "N/A",
77                      "Alarm_Count": 0,
78                      "Alarm_Description": "N/A"
79                  }
80
81                  # 1. Tentar obter informações do device/system
82                  device_url = f"{base_url}/{device_id}/device"
83                  print(f" Buscando informações do device...")
84
85                  device_response = session.get(device_url, headers=auth_headers, timeout=30)
86                  if device_response.status_code == 200:
87                      device_data = device_response.json()
88                      if 'data' in device_data and device_data['data']:
89                          device_info = device_data['data'][0] if isinstance(device_data['data'], list)
                                ↪ else device_data['data']
90
91                          # Extrair informações básicas
92                          if 'NAME' in device_info:
93                              storage_info['Name'] = f"{storage_name} ({device_info['NAME']})"
94                          if 'HEALTHSTATUS' in device_info:
95                              storage_info['Status'] = "OK" if device_info['HEALTHSTATUS'] == "1" else
                                    ↪ "ERROR"
96                          if 'PRODUCTMODE' in device_info:
97                              storage_info['Modelo'] = device_info['PRODUCTMODE']
98                          if 'SOFTWAREVERION' in device_info:
99                              storage_info['Version'] = device_info['SOFTWAREVERION']
100                         if 'PRODUCTVERSION' in device_info:
101                             storage_info['Version'] = device_info['PRODUCTVERSION']
102
103                 # 2. Tentar obter alarmes atuais
104                 alarm_url = f"{base_url}/{device_id}/alarm/current_alarm"
105                 print(f" Buscando alarmes...")
106
107                 alarm_response = session.get(alarm_url, headers=auth_headers, timeout=30)
108                 if alarm_response.status_code == 200:
109                     alarm_data = alarm_response.json()
110                     if 'data' in alarm_data:
111                         if isinstance(alarm_data['data'], list):
112                             alarm_count = len(alarm_data['data'])
113                             storage_info['Alarm_Count'] = alarm_count
114                             storage_info['Alarmes'] = "ERROR" if alarm_count > 0 else "OK"
115
116                             if alarm_count > 0 and len(alarm_data['data']) > 0:
117                                 first_alarm = alarm_data['data'][0]
118                                 storage_info['Alarm_Description'] = first_alarm.get('DESCRIPTION',
                                        ↪ 'Alarme ativo')
```

```
                    elif isinstance(alarm_data['data'], dict) and 'count' in alarm_data['data']:
                        alarm_count = alarm_data['data']['count']
                        storage_info['Alarm_Count'] = alarm_count
                        storage_info['Alarmes'] = "ERROR" if alarm_count > 0 else "OK"

            # 3. Tentar obter informações dos storage pools
            pool_url = f"{base_url}/{device_id}/storagepool"
            print(f"  Buscando informações dos pools...")

            pool_response = session.get(pool_url, headers=auth_headers, timeout=30)
            if pool_response.status_code == 200:
                pool_data = pool_response.json()
                if 'data' in pool_data and pool_data['data']:
                    total_capacity = 0
                    total_used = 0
                    total_free = 0

                    for pool in pool_data['data']:
                        try:
                            if 'USERTOTALCAPACITY' in pool:
                                total_capacity += float(pool['USERTOTALCAPACITY']) * sector_factor /
                                    ↪ 1024
                            if 'USERCONSUMEDCAPACITY' in pool:
                                total_used += float(pool['USERCONSUMEDCAPACITY']) * sector_factor /
                                    ↪ 1024
                            if 'USERFREECAPACITY' in pool:
                                total_free += float(pool['USERFREECAPACITY']) * sector_factor / 1024
                        except (ValueError, TypeError):
                            continue

                    if total_capacity > 0:
                        storage_info['Pool_Capacity_TB'] = round(total_capacity, 2)
                        storage_info['Pool_Used_Capacity_TB'] = round(total_used, 2)
                        storage_info['Pool_Free_TB'] = round(total_free, 2)
                        storage_info['Pool_Used_Percent'] = f"{(total_used/total_capacity):.2%}"

            print(f"  Dados obtidos para {storage_name}")
            return storage_info

        print("  Falha no login")
        return None

    except Exception as e:
        print(f"  Erro ao obter dados: {str(e)}")
        import traceback
        traceback.print_exc()
        return None

def generate_html_report(storage_data_list):
    """Gera relatório HTML com os dados dos storages"""

    html_head = """
    <!DOCTYPE html>
    <html>
    <head>
        <title>Health Check - Storages Huawei</title>
        <style>
            body {
                font-family: Arial, sans-serif;
                margin: 20px;
                background-color: #f5f5f5;
            }
            .container {
                max-width: 1200px;
                margin: 0 auto;
                background: white;
                padding: 20px;
                border-radius: 10px;
                box-shadow: 0 2px 10px rgba(0,0,0,0.1);
            }
            h1 {
                color: #2c3e50;
```

```
            text-align: center;
            margin-bottom: 30px;
        }
        table {
            width: 100%;
            border-collapse: collapse;
            margin: 20px 0;
        }
        th, td {
            border: 1px solid #ddd;
            padding: 12px;
            text-align: left;
        }
        th {
            background-color: #3498db;
            color: white;
            font-weight: bold;
        }
        tr:nth-child(even) {
            background-color: #f8f9fa;
        }
        tr:hover {
            background-color: #e8f4f8;
        }
        .status-ok {
            background-color: #d4edda;
            color: #155724;
            font-weight: bold;
        }
        .status-error {
            background-color: #f8d7da;
            color: #721c24;
            font-weight: bold;
        }
        .status-unknown {
            background-color: #fff3cd;
            color: #856404;
            font-weight: bold;
        }
        .summary {
            background-color: #e7f3ff;
            padding: 15px;
            border-radius: 5px;
            margin: 20px 0;
            border-left: 5px solid #3498db;
        }
    </style>
</head>
<body>
    <div class="container">
        <h1 > Health Check - Storages Huawei</h1>
"""

# Resumo
total_storages = len(storage_data_list)
ok_storages = len([s for s in storage_data_list if s.get('Status') == 'OK'])
error_storages = len([s for s in storage_data_list if s.get('Status') == 'ERROR'])
unknown_storages = len([s for s in storage_data_list if s.get('Status') not in ['OK', 'ERROR']])

html_body = html_head + f"""
        <div class="summary">
            <h3 > Resumo</h3>
            <p><strong>Total de Storages:</strong> {total_storages}</p>
            <p><strong > OK:</strong> {ok_storages}</p>
            <p><strong > ERROR:</strong> {error_storages}</p>
            <p><strong > UNKNOWN:</strong> {unknown_storages}</p>
        </div>

        <table>
            <thead>
                <tr>
                    <th>Name</th>
```

```
                    <th>Modelo</th>
                    <th>Version</th>
                    <th>Serial</th>
                    <th>Pool Capacity (TB)</th>
                    <th>Pool Used (TB)</th>
                    <th>Pool Free (TB)</th>
                    <th>Pool Used %</th>
                    <th>Alarmes</th>
                    <th>Status</th>
                    <th>Alarm Count</th>
                </tr>
            </thead>
            <tbody>
    """

    # Adicionar linhas da tabela
    for storage in storage_data_list:
        status_class = f"status-{storage.get('Status', 'unknown').lower()}"
        alarm_class = f"status-{storage.get('Alarmes', 'unknown').lower()}"

        html_body += f"""
                <tr>
                    <td><strong>{storage.get('Name', 'N/A')}</strong></td>
                    <td>{storage.get('Modelo', 'N/A')}</td>
                    <td>{storage.get('Version', 'N/A')}</td>
                    <td>{storage.get('Serial', 'N/A')}</td>
                    <td>{storage.get('Pool_Capacity_TB', 'N/A')}</td>
                    <td>{storage.get('Pool_Used_Capacity_TB', 'N/A')}</td>
                    <td>{storage.get('Pool_Free_TB', 'N/A')}</td>
                    <td>{storage.get('Pool_Used_Percent', 'N/A')}</td>
                    <td class="{alarm_class}">{storage.get('Alarmes', 'N/A')}</td>
                    <td class="{status_class}">{storage.get('Status', 'N/A')}</td>
                    <td>{storage.get('Alarm_Count', 'N/A')}</td>
                </tr>
        """

    html_body += """
            </tbody>
        </table>
    </div>
    </body>
    </html>
    """

    # Escrever arquivo HTML
    with open("storage_report.html", "w", encoding="utf-8") as f:
        f.write(html_body)

    print(f" Relatório HTML gerado: storage_report.html")

if __name__ == "__main__":
    # Lista de storages para verificar
    storages = [
        {"name": "STG A", "ip": "10.10.10.27"},
        {"name": "STG B", "ip": "10.10.10.24"}
    ]

    all_storage_data = []

    for storage in storages:
        print(f"\n{'='*50}")
        data = get_storage_data(storage["name"], storage["ip"])
        if data:
            all_storage_data.append(data)
        print(f"{'='*50}")

    if all_storage_data:
        generate_html_report(all_storage_data)
        print(f"\ n Relatório gerado com {len(all_storage_data)} storages")
    else:
        print("\ n Nenhum dado foi obtido dos storages")
```

Listing A.19: Teste de autenticação e gestão de sessão iBaseToken

```
#!/usr/bin/env python3
import requests
import json
import sys
from requests.packages.urllib3.exceptions import InsecureRequestWarning

# Desabilitar warnings de SSL
requests.packages.urllib3.disable_warnings(InsecureRequestWarning)

def test_authentication(ip):
    # Configurações
    user_huawei = "monituser"
    password = "Huawei12#$"
    port = "8088"

    base_url = f"https://{ip}:{port}/deviceManager/rest"
    session_url = f"{base_url}/xxxxx/sessions"

    headers = {
        "Content-Type": "application/json",
        "Accept": "application/json"
    }

    body = {
        "username": user_huawei,
        "password": password,
        "scope": "0"
    }

    print(f" Testando autenticação no IP: {ip}")
    print(f" URL: {session_url}")
    print(f" User: {user_huawei}")
    print(f" Password: {'*' * len(password)}")

    try:
        # Fazer o POST para autenticar
        print("\ n Enviando pedido de autenticação...")
        response = requests.post(
            session_url,
            headers=headers,
            json=body,
            verify=False,
            timeout=30
        )

        print(f" Status Code: {response.status_code}")
        print(f" Headers Response: {dict(response.headers)}")

        if response.status_code == 200:
            data = response.json()
            print(f" Response JSON: {json.dumps(data, indent=2)}")

            if 'data' in data and data['data']:
                print("  AUTENTICAÇÃO BEM SUCEDIDA!  ")
                print(f" Device ID: {data['data'].get('deviceid', 'N/A')}")
                print(f" iBaseToken: {data['data'].get('iBaseToken', 'N/A')}")
                print(f" User ID: {data['data'].get('userid', 'N/A')}")
                print(f" Role: {data['data'].get('roleId', 'N/A')}")
                return True
            else:
                print("  AUTENTICAÇÃO FALHOU!  ")
                print(f" Erro: {data.get('error', {})}")
                return False
        else:
            print(f"  HTTP ERROR: {response.status_code}  ")
            print(f" Response Text: {response.text}")
            return False

    except Exception as e:
        print(f"  EXCEPTION: {str(e)}  ")
```

```
71          import traceback
72          traceback.print_exc()
73          return False
74
75  if __name__ == "__main__":
76      if len(sys.argv) != 2:
77          print("Uso: python3 test_auth.py <ip_do_storage>")
78          sys.exit(1)
79
80      ip = sys.argv[1]
81      success = test_authentication(ip)
82
83      if success:
84          print("\n  TESTE DE AUTENTICAÇÃO PASSOU!   ")
85      else:
86          print("\n  TESTE DE AUTENTICAÇÃO FALHOU!   ")
```

Esta página foi intencionalmente deixada em branco.

Apêndice B

Repositório de Imagens

Este apêndice reúne as imagens geradas durante os ensaios e a análise comparativa, preservando os nomes de ficheiro como referência direta ao conjunto de dados correspondente.

Solução Single – 4K

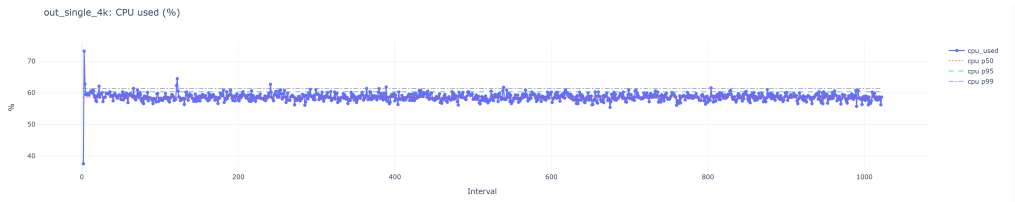


Figura B.1: Solução Single – utilização de CPU do controlador (bloco 4K)

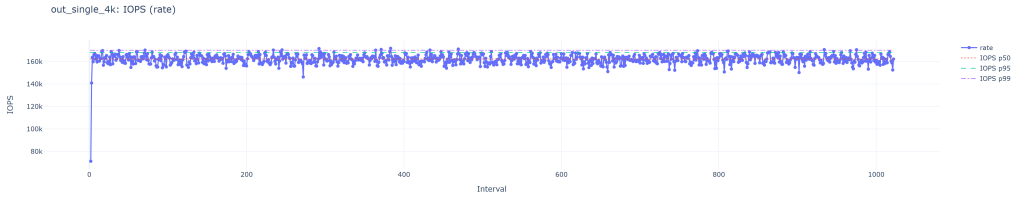


Figura B.2: Solução Single – IOPS (bloco 4K)

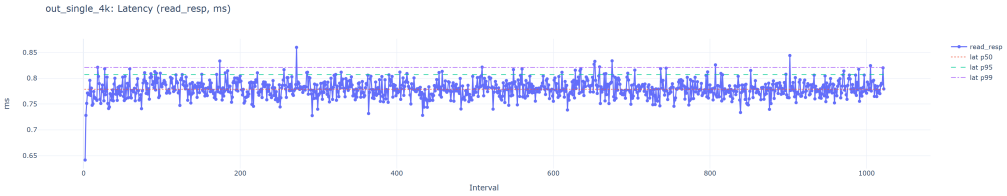


Figura B.3: Solução Single – latência de leitura (bloco 4K)

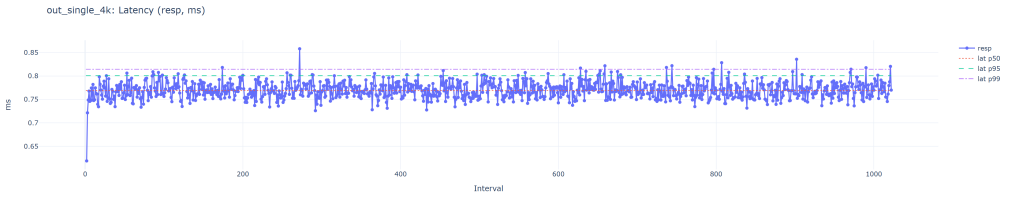


Figura B.4: Solução Single – latência de resposta (bloco 4K)

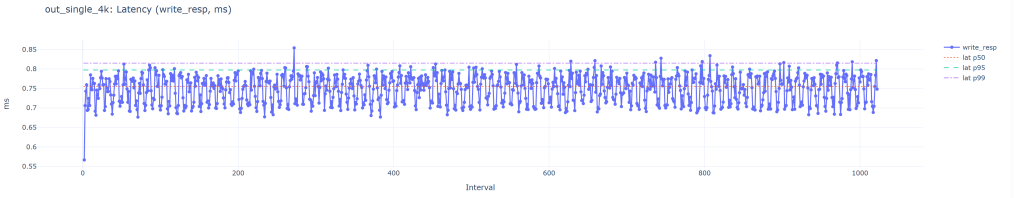


Figura B.5: Solução Single – latência de escrita (bloco 4K)

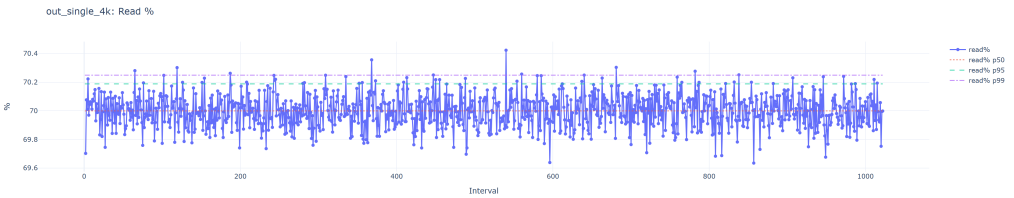


Figura B.6: Solução Single – percentagem de leitura (bloco 4K)

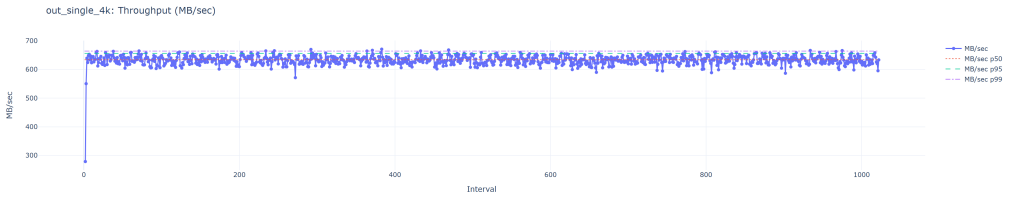


Figura B.7: Solução Single – throughput (bloco 4K)

Solução Single – 64K

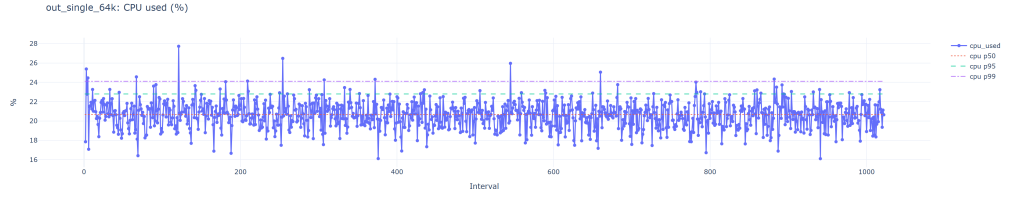


Figura B.8: Solução Single – utilização de CPU do controlador (bloco 64K)

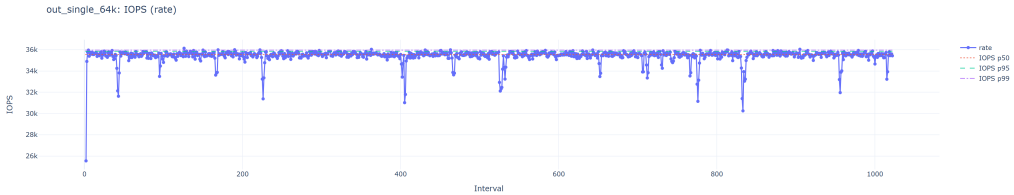


Figura B.9: Solução Single – IOPS (bloco 64K)

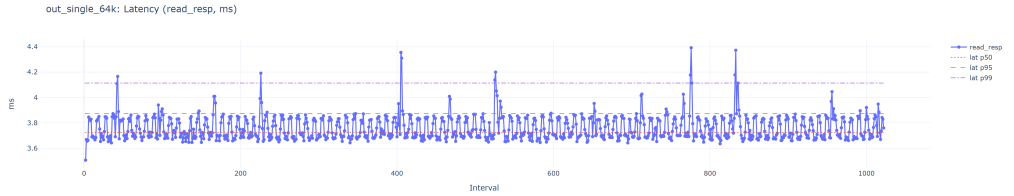


Figura B.10: Solução Single – latência de leitura (bloco 64K)

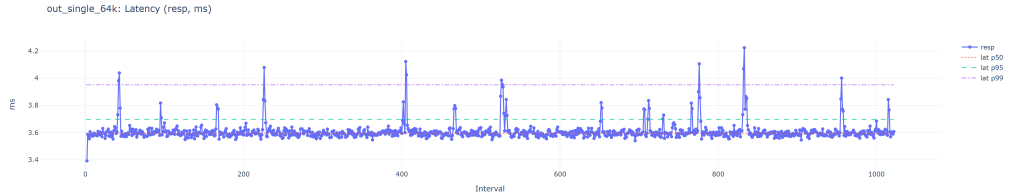


Figura B.11: Solução Single – latência de resposta (bloco 64K)

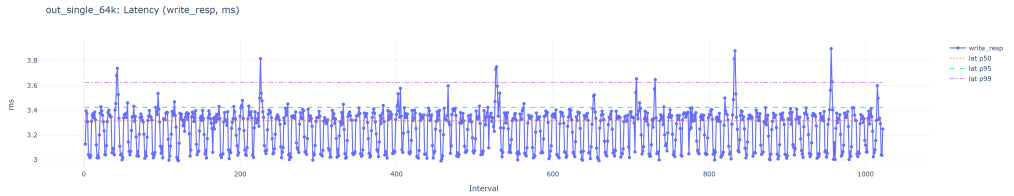


Figura B.12: Solução Single – latência de escrita (bloco 64K)

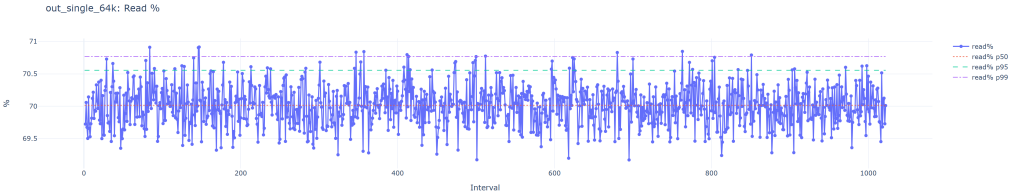


Figura B.13: Solução Single – percentagem de leitura (bloco 64K)

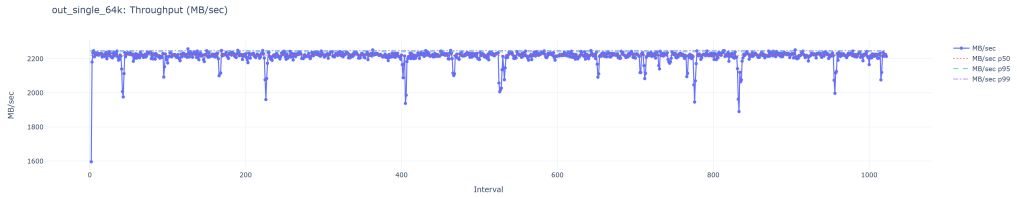


Figura B.14: Solução Single – throughput (bloco 64K)

Solução Ativo Ativo – 4K

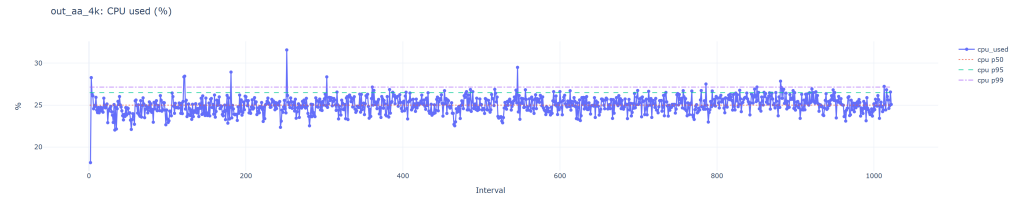


Figura B.15: Solução Ativo Ativo – utilização de CPU do controlador (bloco 4K)

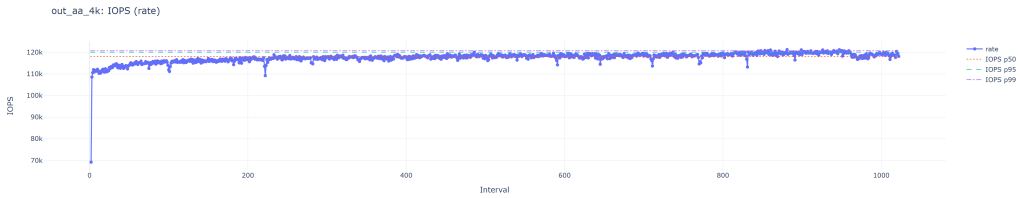


Figura B.16: Solução Ativo Ativo – IOPS (bloco 4K)

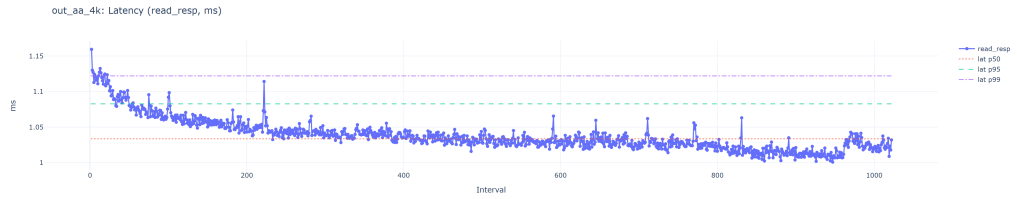


Figura B.17: Solução Ativo Ativo – latência de leitura (bloco 4K)

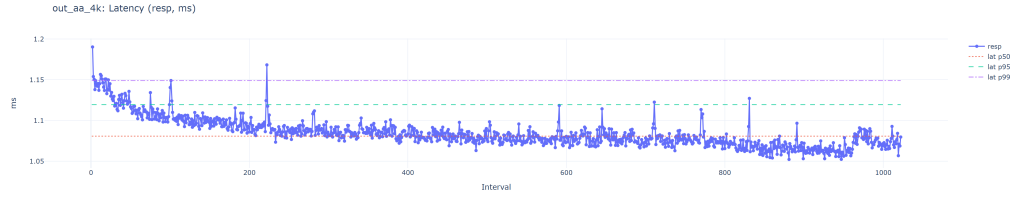


Figura B.18: Solução Ativo Ativo – latência de resposta (bloco 4K)

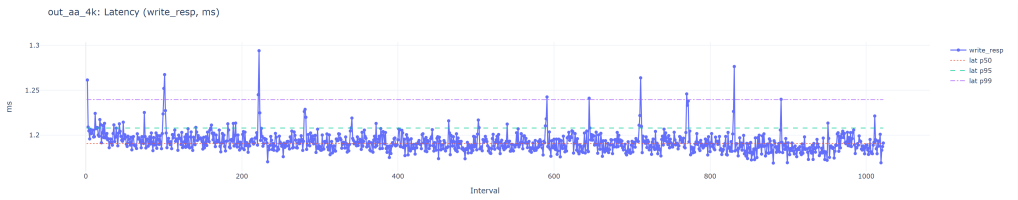


Figura B.19: Solução Ativo Ativo – latência de escrita (bloco 4K)

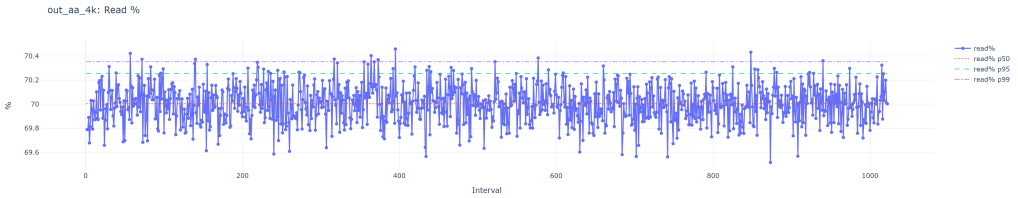


Figura B.20: Solução Ativo Ativo – percentagem de leitura (bloco 4K)

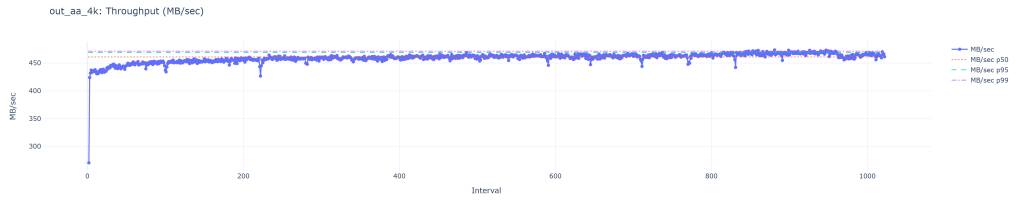


Figura B.21: Solução Ativo Ativo – throughput (bloco 4K)

Solução Ativo Ativo – 64K

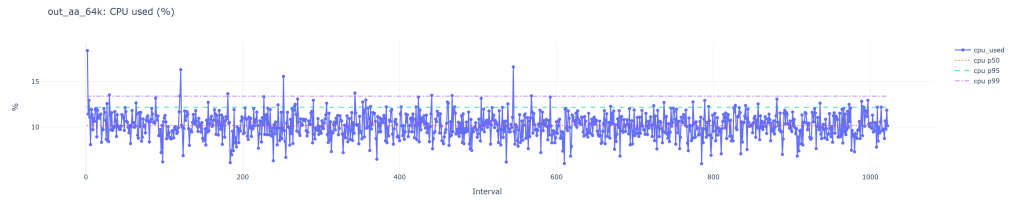


Figura B.22: Solução Ativo Ativo – utilização de CPU do controlador (bloco 64K)

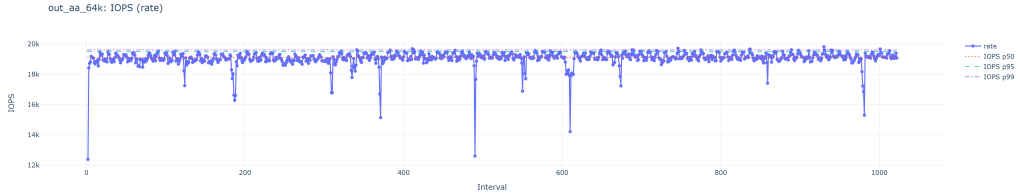


Figura B.23: Solução Ativo Ativo – IOPS (bloco 64K)

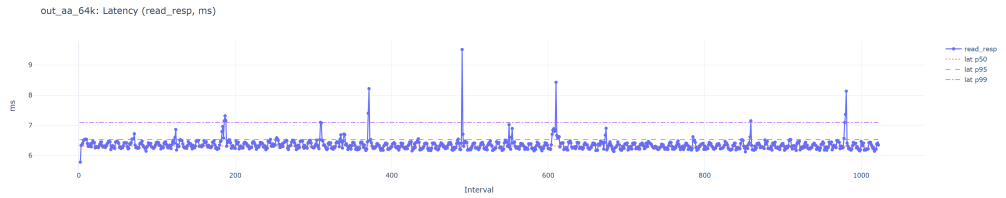


Figura B.24: Solução Ativo Ativo – latência de leitura (bloco 64K)

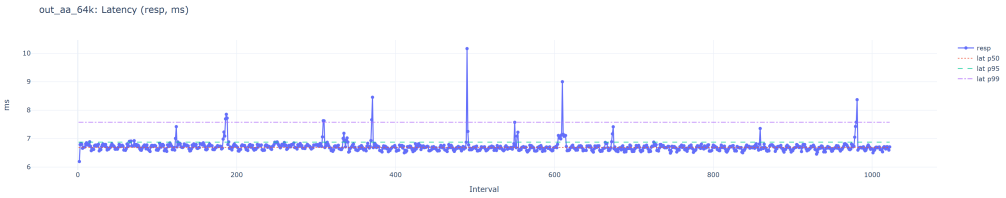


Figura B.25: Solução Ativo Ativo – latência de resposta (bloco 64K)

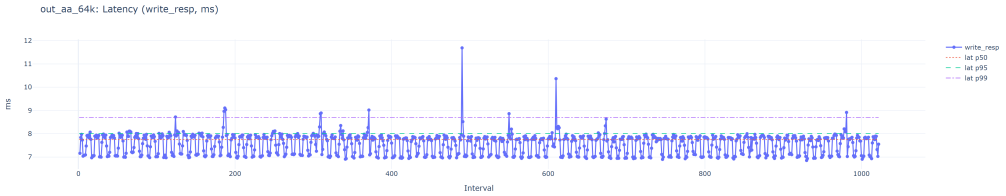


Figura B.26: Solução Ativo Ativo – latência de escrita (bloco 64K)

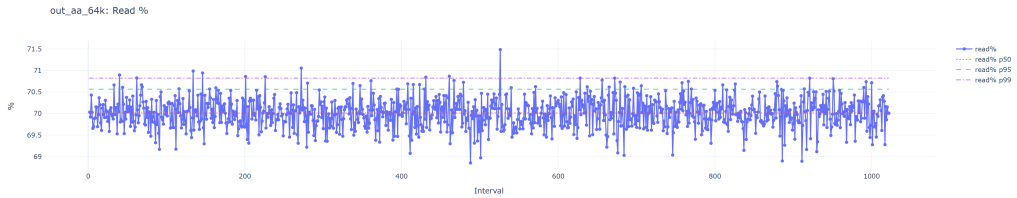


Figura B.27: Solução Ativo Ativo – percentagem de leitura (bloco 64K)

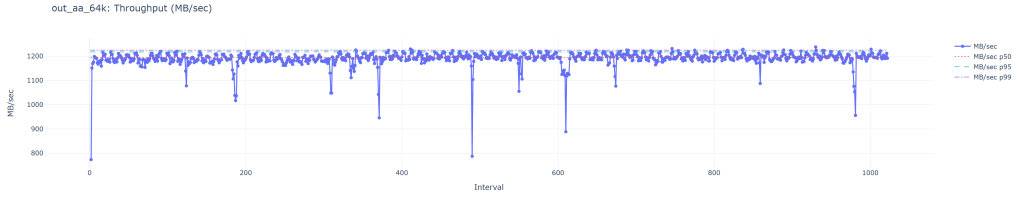


Figura B.28: Solução Ativo Ativo – throughput (bloco 64K)

Comparação entre soluções por tipo de métrica

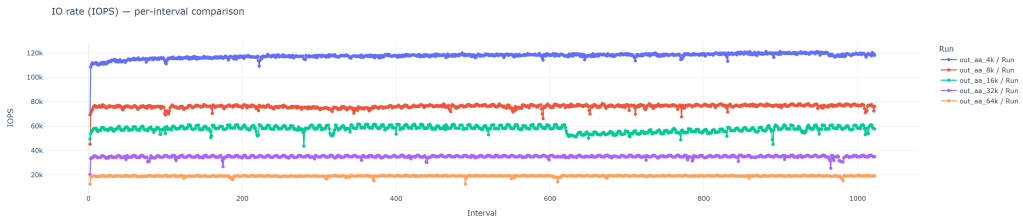


Figura B.29: Solução Ativo Ativo – IOPS em função do tamanho de bloco

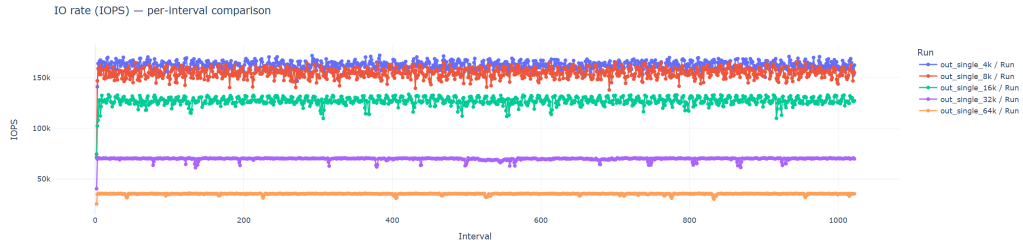


Figura B.30: Solução Single – IOPS em função do tamanho de bloco

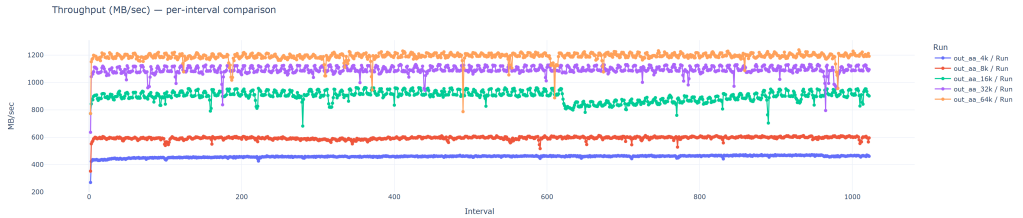


Figura B.31: Solução Ativo Ativo – throughput em função do tamanho de bloco

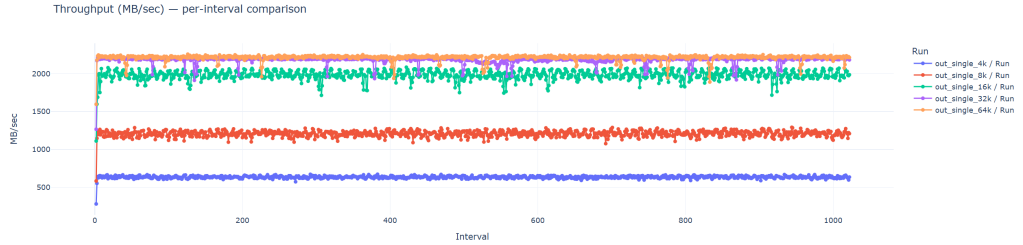


Figura B.32: Solução Single – throughput em função do tamanho de bloco

Esta página foi intencionalmente deixada em branco.

This page is intentionally left blank

ipt Instituto Politécnico de Tomar