Universidade Federal de Ouro Preto
Instituto de Ciências Exatas e Aplicadas
Departamento de Computação e Sistemas

Implementação de Sistema Distribuído de Virtualização com Proxmox VE e Armazenamento Ceph

Hércules Aparecido Teixeira

TRABALHO DE CONCLUSÃO DE CURSO

ORIENTAÇÃO:
Theo Silva Lins

COORIENTAÇÃO:
Mauricio Jose Aureliano Junior

Julho, 2026
João Monlevade–MG

Hércules Aparecido Teixeira

Implementação de Sistema Distribuído de Virtualização com Proxmox VE e Armazenamento Ceph

Orientador: Theo Silva Lins

Coorientador: Mauricio Jose Aureliano Junior

Monografia apresentada ao curso de Engenharia da Computação do Instituto de Ciências Exatas e Aplicadas, da Universidade Federal de Ouro Preto, como requisito parcial para aprovação na Disciplina "Trabalho de Conclusão de Curso II".

Universidade Federal de Ouro Preto

João Monlevade

Julho de 2026

SISBIN - SISTEMA DE BIBLIOTECAS E INFORMAÇÃO

T266i Teixeira, Hercules Aparecido.
Implementação de sistema distribuído de virtualização com Proxmox VE e armazenamento Ceph. [manuscrito] / Hercules Aparecido Teixeira. - 2026.
48 f.: il.: color., gráf., tab..

Orientador: Prof. Dr. Theo Silva Lins.
Coorientador: Prof. Me. Mauricio Jose Aureliano Junior.
Monografia (Bacharelado). Universidade Federal de Ouro Preto. Instituto de Ciências Exatas e Aplicadas. Graduação em Engenharia de Computação .

1. Computação tolerante a falhas. 2. Computadores digitais eletrônicos - Avaliação. 3. Falhas de sistemas de computação. 4. Proxmox (Plataforma de computação). 5. Sistemas de computação virtual. 6. Sistemas operacionais distribuídos (Computadores). I. Lins, Theo Silva. II. Aureliano Junior, Mauricio Jose. III. Universidade Federal de Ouro Preto. IV. Título.

CDU 004.78:004.94

Bibliotecário(a) Responsável: Flavia Reis - CRB6/2431

MINISTÉRIO DA EDUCAÇÃO
UNIVERSIDADE FEDERAL DE OURO PRETO
REITORIA
INSTITUTO DE CIENCIAS EXATAS E APLICADAS
DEPARTAMENTO DE COMPUTACAO E SISTEMAS

FOLHA DE APROVAÇÃO

Hércules Aparecido Teixeira

Implementação de Sistema Distribuído de Virtualização com Proxmox VE e Armazenamento Ceph

Monografia apresentada ao Curso de Engenharia de Computação da Universidade Federal
de Ouro Preto como requisito parcial para obtenção do título de Bacharelado em Engenharia de Computação

Aprovada em 30 de Julho de 2026

Membros da banca

Doutor - Theo Silva Lins - Orientador - Universidade Federal de Ouro Preto
Mestre - Maurício José Aureliano Junior - Co-Orientador - Universidade Federal de Ouro Preto
Doutor- Marlon Paolo Lima - Universidade Federal de Ouro Preto
Doutor - Samuel Souza Brito - Universidade Federal de Ouro Preto

Theo Silva Lins, orientador do trabalho, aprovou a versão final e autorizou seu depósito na Biblioteca Digital de Trabalhos de Conclusão de Curso da UFOP em 03/08/2026

Documento assinado eletronicamente por **Theo Silva Lins**, **PROFESSOR DE MAGISTERIO SUPERIOR**, em 03/08/2026, às 22:28, conforme horário oficial de Brasília, com fundamento no art. 6º, § 1º, do Decreto nº 8.539, de 8 de outubro de 2015.

A autenticidade deste documento pode ser conferida no site http://sei.ufop.br/sei/controlador_externo.php?acao=documento_conferir&id_orgao_acesso_externo=0 , informando o código verificador **1151687** e o código CRC **4ECEC7F4**.

R. Diogo de Vasconcelos, 122, - Bairro Pilar Ouro Preto/MG, CEP 35402-163
Telefone: (31)3808-0819 - www.ufop.br

Resumo

A demanda crescente por sistemas computacionais escaláveis e resilientes impulsionou o uso de Infraestruturas Hiperconvergentes (HCI) baseadas nas ferramentas Proxmox e Ceph. No entanto, a camada de abstração criada pelos hipervisores gera o que se denomina "Dilema do I/O", cenário em que o isolamento seguro do hardware entra em conflito direto com a necessidade de alcançar o máximo de performance bruta. Para investigar esse problema, este trabalho apresenta a implementação prática e a auditoria técnica de um cluster HCI, com foco em analisar como as diferentes camadas de virtualização impactam o throughput, a latência de cauda (tail latency) e o custo computacional dentro do caminho de I/O. A metodologia utilizou testes empíricos para comparar três ambientes distintos: servidores físicos sem virtualização (bare metal), contêineres isolados (LXC) e máquinas virtuais completas (KVM). Para extrair as métricas, foram aplicadas as ferramentas de estresse FIO, rados bench e sysbench em diferentes cenários de rede, variando a capacidade de 100 Mbps até 10 Gbps. Os resultados mostraram um gargalo claro nas redes de 10 Gbps: nessa alta velocidade, a virtualização KVM sofreu uma queda de desempenho de 98,98% quando comparada ao ambiente bare metal, com a máquina virtual estagnada em 415 IOPS enquanto o limite físico nominal alcançou 40.800 IOPS. A pesquisa revelou que a causa real desse gargalo não é uma limitação física da rede, mas sim a exaustão do processador, sobrecarregado pela grande quantidade de interrupções lógicas (VMExits e Interrupt Storms) processadas pelo framework VirtIO. Por outro lado, os contêineres LXC apresentaram uma eficiência muito superior: ao usar o módulo nativo do sistema operacional (krbd), o LXC conseguiu entregar um desempenho quase idêntico ao do servidor físico puro. Além da performance, a pesquisa avaliou a resiliência da arquitetura através de simulações de falha, confirmando que o algoritmo CRUSH é robusto e consegue manter o sistema disponível mesmo quando um servidor é desligado abruptamente, embora com aumento temporário na latência durante o processo automático de recuperação de dados (backfill). Com base nesses dados, o estudo entrega um veredito arquitetural claro: para aplicações que exigem leitura e escrita intensivas (I/O-bound) em redes de alta velocidade, recomenda-se fortemente o uso de contêineres LXC, garantindo a máxima eficiência do sistema e evitando gastos desnecessários com a expansão da infraestrutura física.

Palavras-chave: Infraestrutura Hiperconvergente. Proxmox. Ceph. Virtualização. Dilema do I/O.

Abstract

The growing demand for scalable and resilient computing systems has boosted the adoption of Hyperconverged Infrastructures (HCI) based on Proxmox and Ceph tools. However, the abstraction layer created by hypervisors introduces what is referred to as the "I/O Dilemma". In this scenario, the strict security of hardware isolation directly conflicts with the need to achieve maximum raw performance. To investigate this problem, this work presents the practical implementation and technical auditing of an HCI cluster, focusing on analyzing how different virtualization layers impact throughput, tail latency, and computational cost within the I/O path. The methodology used empirical tests to compare three distinct environments: physical servers without virtualization (bare metal), isolated containers (LXC), and full virtual machines (KVM). To extract the metrics, the `FIO`, `rados bench`, and `sysbench` stress tools were applied across different network scenarios, varying the capacity from 100 Mbps up to 10 Gbps. The results revealed a clear bottleneck in the 10 Gbps networks: at this high speed, KVM virtualization suffered a severe performance drop of 98.98% when compared to the bare metal environment, with the virtual machine stagnating at 415 IOPS while the nominal physical limit reached 40,800 IOPS. The research demonstrated that the true cause of this bottleneck is not a physical network limitation, but rather processor exhaustion, as the CPU becomes overwhelmed by the massive amount of logical interruptions (VMExits and Interrupt Storms) processed by the `VirtIO` framework. On the other hand, LXC containers demonstrated vastly superior efficiency: by utilizing the native operating system module (`krbd`), the LXC instances managed to deliver a performance nearly identical to a pure physical server. In addition to performance, the research evaluated the architecture's resilience through simulated hardware failures, confirming that the CRUSH algorithm is robust and successfully maintains system availability even when a server abruptly shuts down, although with a temporary latency penalty during the automatic data recovery process (backfill). Based on these findings, the study delivers a clear architectural verdict: for applications demanding intensive read and write operations (I/O-bound) over high-speed networks, the use of LXC containers is highly recommended, ensuring maximum systemic efficiency and preventing unnecessary physical infrastructure upgrades.

Keywords: Hyperconverged Infrastructure. Proxmox. Ceph. Virtualization. I/O Dilemma.

Lista de Figuras

Lista de Tabelas

Lista de abreviaturas e siglas

ACID	*Atomicity, Consistency, Isolation, Durability*
CAP	*Consistency, Availability, Partition tolerance*
COSI	Colegiado do Curso de Sistemas de Informação
CPU	*Central Processing Unit*
CRUSH	*Controlled Replication Under Scalable Hashing*
DECSI	Departamento de Computação e Sistemas
HCI	*Hyperconverged Infrastructure* (Infraestrutura Hiperconvergente)
ICEA	Instituto de Ciências Exatas e Aplicadas
IOPS	*Input/Output Operations Per Second*
KVM	*Kernel-based Virtual Machine*
LXC	*Linux Containers*
MON	*Ceph Monitor*
NAS	*Network Attached Storage*
OLTP	*Online Transaction Processing*
OSD	*Object Storage Daemon*
PG	*Placement Group*
PID	*Process Identifier*
PVE	*Proxmox Virtual Environment*
QEMU	*Quick Emulator*
RADOS	*Reliable Autonomic Distributed Object Store*
RAM	*Random Access Memory*
RBD	*RADOS Block Device*
RTO	*Recovery Time Objective*

SAN	*Storage Area Network*
SDS	*Software-Defined Storage* (Armazenamento Definido por Software)
SPOF	*Single Point of Failure* (Ponto Único de Falha)
TCO	*Total Cost of Ownership* (Custo Total de Propriedade)
UFOP	Universidade Federal de Ouro Preto
VFS	*Virtual File System*
VMM	*Virtual Machine Monitor*

Lista de símbolos

$\approx$	Valor aproximadamente igual
D_{KVM}	Taxa de degradação de desempenho no ambiente KVM
μ	Média aritmética dos resultados dos ensaios
T	Instante de tempo cronológico durante ensaios de resiliência
%	Percentual de utilização de recursos ou perda de performance
$\rightarrow$	Operador de tendência ou mapeamento matemático
$\in$	Pertence a um conjunto ou espaço de endereçamento

Sumário

1 Introdução

A evolução dos modernos centros de processamento de dados ganhou força com a transição para a Infraestrutura Hiperconvergente (HCI). Esse modelo unifica computação, rede e armazenamento nos mesmos servidores físicos, substituindo as antigas topologias segregadas, como as Redes de Área de Armazenamento (SAN — *Storage Area Network*), que fornecem acesso em nível de bloco aos discos, e o Armazenamento Anexado à Rede (NAS — *Network Attached Storage*), que oferece acesso em nível de arquivo por meio de protocolos como NFS ou SMB. Na prática, o HCI facilita a expansão da infraestrutura e ajuda a eliminar os pontos únicos de falha (PROXMOX SERVER SOLUTIONS GMBH, 2024). Esse cenário ganha relevância mercadológica diante das barreiras financeiras impostas por soluções proprietárias consolidadas, como o VMware vSphere, cujos altos custos de licenciamento inviabilizam a adoção em ambientes com restrições orçamentárias estritas. Dentro desse ecossistema, destacam-se as plataformas de Armazenamento Definido por Software (SDS), como o Ceph. O Ceph garante que os dados permaneçam seguros e bem distribuídos na rede utilizando um algoritmo determinístico chamado Controlled Replication Under Scalable Hashing (CRUSH), que decide de forma inteligente onde cada dado será salvo (WEIL et al., 2006).

Apesar das vantagens, integrar armazenamento e processamento no mesmo equipamento traz um grande desafio arquitetural. Na engenharia de sistemas, esse problema é conhecido como o "Dilema do I/O", que consiste no conflito entre a segurança do isolamento de hardware e a necessidade de alta performance de E/S, resultando em sobrecarga computacional no caminho dos dados. O gargalo ocorre porque, mesmo que a rede possua enlaces físicos de alta capacidade, a camada de virtualização cria uma sobrecarga de processamento. Essa abstração acaba funcionando como uma barreira que limita o desempenho. Na virtualização completa, gerenciada por hipervisores como o Kernel-based Virtual Machine (KVM), a segurança baseia-se no isolamento estrito do hardware. Por causa desse isolamento, cada operação de leitura ou escrita precisa percorrer um longo caminho. A instrução sai do sistema operacional da máquina virtual pelo Sistema de Arquivos Virtual (VFS), aciona o driver de paravirtualização `VirtIO` e, finalmente, gera interrupções lógicas no processador físico conhecidas como VMExits.

Essas interrupções constantes forçam o processador a realizar várias trocas de contexto. A CPU precisa alternar repetidamente entre o espaço do usuário, onde roda o processo Quick Emulator (`QEMU`), e o núcleo do sistema hospedeiro. Esse esforço contínuo pode esgotar o processador, causando tempestades de interrupção (KIVITY et al., 2007). Como consequência, essa pesada sobrecarga arquitetural diminui a vazão de dados e aumenta os atrasos de resposta e a latência de cauda.

Por outro lado, existem tecnologias de virtualização mais leves que operam diretamente no nível do sistema operacional, como os Contêineres Linux (LXC). Os contêineres reduzem esse afunilamento transacional porque não precisam emular um hardware completo. Eles funcionam utilizando recursos nativos do sistema, como *namespaces* e *cgroups*. Dessa forma, uma instância LXC compartilha o próprio núcleo do servidor físico, permitindo que as requisições de disco acessem o módulo nativo do Ceph (`krbd`) de forma rápida e direta (ROSEN, 2014).

Esse encurtamento no caminho dos dados economiza processamento e preserva ciclos de clock da CPU. No entanto, o LXC possui limitações de flexibilidade e isolamento, o que pode ser um problema em ambientes onde diversos clientes não confiáveis dividem o mesmo servidor. Para gerenciar tudo isso, o Proxmox Virtual Environment (PVE) atua como orquestrador central. Ele consegue criar e rodar máquinas virtuais e contêineres ao mesmo tempo, compartilhando a mesma malha de armazenamento (PROXMOX SERVER SOLUTIONS GMBH, 2024). Apesar de essa arquitetura ser amplamente adotada pelo mercado, a literatura técnica ainda carece de experimentos práticos que calculem matematicamente o quanto o isolamento lógico penaliza a velocidade de E/S.

1.1 Problema de Pesquisa

Na infraestrutura moderna, projetistas sempre precisam equilibrar duas necessidades opostas: a segurança de isolar processos em máquinas virtuais e a exigência de bancos de dados por discos de baixíssima latência. Esse balanço técnico define a lacuna explorada por esta monografia. Diante disso, a presente pesquisa busca responder à seguinte questão fundamental:

Qual é o impacto quantitativo da sobrecarga gerada pela emulação de hardware (KVM) quando comparado ao isolamento mais leve dos contêineres (LXC)?

Especificamente, como essas diferentes camadas afetam a vazão de dados, a latência de cauda e o custo de processamento na CPU dentro do caminho de E/S? Essa análise será aplicada a um ecossistema hiperconvergente rodando Proxmox VE e Ceph, avaliando inclusive o comportamento do sistema durante a recuperação automática de falhas.

1.2 Objetivos

Para orientar a condução experimental e a análise dos resultados, os objetivos foram estruturados em uma perspectiva geral e em etapas técnicas específicas, conforme detalhado a seguir.

1.2.1 Objetivo Geral

Avaliar na prática a degradação de desempenho arquitetural e o custo de CPU no caminho de dados gerados por dois modelos diferentes de virtualização (KVM e LXC). Para isso, o estudo irá mensurar as taxas de desempenho transacional síncrono e assíncrono, a alocação do tempo de processamento de sistema e a capacidade de tolerância a falhas do agrupamento sob uma restrição física de apenas dois servidores.

1.2.2 Objetivos Específicos

Para alcançar e validar o objetivo geral, o projeto foi estruturado nas seguintes etapas técnicas:

- Desenhar e montar a rede do cluster HCI separando estrategicamente o tráfego de gerência da malha de sincronização de alta velocidade do Ceph;
- Configurar o mapeamento dos discos virtuais RADOS Block Device (RBD), adaptando as regras de redundância lógicas do armazenamento para respeitar os limites físicos do laboratório. Essa etapa criará a base para as três frentes de testes: servidores físicos puros, LXC e KVM;
- Executar baterias de testes de estresse para saturar os discos de forma assíncrona e transacional utilizando as ferramentas FIO e sysbench. O foco será descobrir o limite máximo de Input/Output Operations Per Second (IOPS) e vazão trafegando sobre diferentes capacidades de rede;
- Medir a sobrecarga lógica causada pelo núcleo do servidor hospedeiro monitorando o esforço computacional e as transições de privilégio;
- Simular falhas reais desligando um dos servidores de forma abrupta para auditar a mecânica de segurança do algoritmo CRUSH, observando os estados de alerta e recuperação da malha;
- Elaborar um laudo arquitetural fundamentado em dados empíricos que auxilie projetistas a decidirem quando priorizar o isolamento lógico ou o desempenho bruto de disco.

1.3 Justificativa

Este estudo se justifica pela necessidade de quebrar um mito comum na administração de *data centers*. Muitos profissionais assumem, de forma equivocada, que o afunilamento em discos distribuídos sempre é causado por uma latência na rede física. Na prática, arquitetos de infraestrutura criam discos virtuais para bancos de dados pesados

rodando o driver `VirtIO` e ignoram o estrangulamento real que ocorre quando a CPU se esgota ao tentar processar milhões de interrupções de software.

Ao analisar detalhadamente o "Dilema do I/O", esta monografia vai além das especificações teóricas de fabricantes e entrega dados empíricos estruturados. O projeto comprova que conectar servidores com enlaces de 10 Gbps pode ser um desperdício se a máquina virtual não possuir capacidade de processar as chamadas lógicas. Em contrapartida, mostra que os contêineres conseguem extrair toda a velocidade nativa do hardware.

Como resultado, esta pesquisa atua como um guia para projetistas de sistemas distribuídos. Ao escolher a tecnologia de isolamento correta para as reais demandas da carga de trabalho, as organizações otimizam o custo total de propriedade (TCO) sem comprometer o ambiente, apresentando-se como uma solução de código aberto de alto valor frente aos pesados custos de licenciamento atuais.

1.4 Organização do Documento

O documento está organizado da seguinte maneira: Capítulo 2 constrói a fundamentação teórica sobre os sistemas hiperconvergentes e cita os trabalhos relacionados. O Capítulo 3 descreve a planta arquitetural e a sistemática empírica dos ensaios. O Capítulo 4 destina-se à análise quantitativa da telemetria capturada, correlacionando os dados de vazão, processamento intermitente e resiliência a falhas físicas. Por fim, O Capítulo 5 encerra a monografia com a conclusão técnica e a indicação de extensões de pesquisa.

2 Referencial Teórico e Estado da Arte

A Infraestrutura Hiperconvergente (HCI) é um modelo arquitetural que integra computação, rede e armazenamento em um mesmo conjunto de servidores físicos, gerenciados de forma unificada por software. Já o Armazenamento Definido por Software (SDS) é uma abordagem que separa a lógica de gerenciamento de armazenamento do hardware subjacente, permitindo flexibilidade, escalabilidade e independência de fornecedor. Ambos os conceitos são fundamentais para o entendimento do ambiente experimental desta pesquisa.

A fundamentação teórica desta pesquisa explora como as camadas de abstração funcionam dentro dos sistemas computacionais modernos. Para compreender as métricas de desempenho capturadas nos ensaios práticos, é necessário primeiro analisar os mecanismos que isolam os recursos do servidor. Além disso, é essencial entender a arquitetura por trás do Armazenamento Definido por Software.

Nos últimos anos, a literatura científica voltou sua atenção para a otimização do SDS dentro dos ecossistemas de Infraestrutura Hiperconvergente. Estudos recentes buscam maneiras práticas de reduzir os gargalos de processador e de rede que são comuns nesses ambientes. Para evitar a queda de desempenho em sistemas de alta disponibilidade, as pesquisas propõem diversas soluções. Essas abordagens vão desde o ajuste fino nos fluxos de entrada e saída até a criação de canais de comunicação direta com o equipamento físico.

2.1 Paradigmas de Virtualização e Abstração de Recursos

A virtualização funciona através da inserção de uma camada de software específica, conhecida como hipervisor ou Monitor de Máquina Virtual (VMM - *Virtual Machine Monitor*). Essa camada atua como uma ponte controladora entre o equipamento físico do servidor e os sistemas operacionais convidados. O grande objetivo do hipervisor é emular os componentes físicos. Essa emulação garante que cada máquina virtual fique isolada logicamente, o que aumenta a segurança geral do ambiente.

2.1.1 Virtualização Baseada em Hipervisor (KVM)

O KVM consolidou-se como a ferramenta padrão para virtualização nativa (Tipo 1) dentro do universo Linux. Na prática, o KVM transforma o próprio kernel Linux em um hipervisor completo. Para rodar comandos críticos com segurança, ele aproveita extensões embutidas diretamente nos processadores, como as tecnologias Intel VT-x ou AMD-V. No

entanto, esse alto nível de segurança tem um preço: o isolamento rigoroso do hardware gera uma sobrecarga considerável no processamento (KIVITY et al., 2007).

Para aliviar o peso da comunicação de disco e rede, as máquinas virtuais utilizam o framework `VirtIO`. Essa ferramenta aplica a técnica de paravirtualização para tentar reduzir o atraso nas respostas. Mesmo com essa ajuda, a literatura acadêmica mostra que o gargalo continua existindo durante as trocas de privilégio no sistema (ALGARNI et al., 2018).

A dinâmica de alternância de privilégios obedece estritamente ao comportamento nativo mapeado no desenvolvimento da ferramenta. Conforme descrito por seus próprios criadores, a execução do sistema convidado prossegue até a ocorrência de uma instrução que não pode ser virtualizada, momento em que o processador realiza um VMExit e transfere o controle ao hipervisor KVM (KIVITY et al., 2007). A partir desse ponto, o hipervisor precisa realizar um processamento intensivo: ele salva o estado atual da CPU, processa a interrupção gerada e repassa essa tarefa para o processo `QEMU`, que roda no espaço de usuário para processar as operações de entrada e saída. Essa alternância gera um ciclo exaustivo de trocas de contexto. Esse esforço contínuo é o principal responsável pela queda no volume de IOPS e pelo disparo na latência de cauda, especialmente em aplicações com uso intensivo de dados (KIVITY et al., 2007).

2.1.2 Virtualização a Nível de Sistema Operacional (LXC)

O KVM emula uma máquina inteira, mas os LXC seguem um caminho bem diferente. Eles trabalham apenas com o isolamento lógico dos processos, o que permite que todas as instâncias compartilhem o mesmo kernel Linux do servidor hospedeiro simultaneamente. Essa arquitetura mais enxuta garante uma eficiência notável, que se sustenta em dois pilares fundamentais do núcleo Linux: *namespaces* e *control groups* (cgroups), amplamente discutidos na literatura de virtualização por contêineres (ROSEN, 2014):

1. **Namespaces:** Essa tecnologia cria ambientes virtuais isolados. Ela garante que uma instância não enxergue a outra, dividindo de forma segura as placas de rede, a árvore de arquivos e os identificadores de processos (PIDs).
2. **Control Groups (cgroups):** Esta ferramenta atua como um administrador rígido de consumo. Ela fiscaliza e distribui a quantidade exata de processador, memória RAM e banda de disco que cada contêiner pode usar. Isso evita que um único processo defeituoso sature e trave todo o servidor físico.

Como os contêineres não emulam o hardware, o caminho percorrido pelos dados no LXC é significativamente mais curto. Quando um banco de dados dentro do contêiner solicita uma escrita, essa ação não gera as custosas interrupções lógicas. Em vez disso, o

pedido é tratado como uma simples chamada de sistema, enviada diretamente ao kernel Linux do servidor físico. Somado a isso, ao utilizar o módulo `krbd`, o LXC mapeia os discos virtuais do Ceph de forma nativa, dentro do próprio espaço protegido do sistema. Essa união de fatores arquiteturais justifica um resultado impressionante: o desempenho do LXC aproxima-se do limite nominal de banda de um servidor físico puro. É exatamente esse fenômeno de alta velocidade que é observado na interface expressa de 10 Gbps durante a experimentação prática detalhada nesta monografia, o que está alinhado com os resultados de avaliações oficiais da plataforma para redes de alta capacidade, que demonstram a saturação de enlaces de 10 Gbps em condições semelhantes (PROXMOX SERVER SOLUTIONS GMBH, 2023).

2.2 Armazenamento Distribuído com Ceph

O Ceph é uma plataforma robusta de SDS. Ele foi arquitetado desde o princípio para ser altamente expansível e para acabar com os pontos únicos de falha dentro do centro de processamento de dados. O cérebro do sistema é o Reliable Autonomic Distributed Object Store (RADOS). O RADOS assume o controle e gerencia de forma inteligente a distribuição dos objetos em um agrupamento formado por diferentes servidores físicos (WEIL et al., 2007).

2.2.1 O Algoritmo CRUSH e o Determinismo Topológico

Os sistemas de armazenamento legados costumam depender de um servidor central, que guarda uma tabela centralizada de mapeamento de metadados. O Ceph inova ao eliminar completamente a necessidade dessa tabela centralizada. O algoritmo inovador afasta-se dos modelos tradicionais de indexação. Conforme definido por seus criadores, o CRUSH é uma função pseudoaleatória escalável de distribuição de dados, projetada para sistemas de armazenamento orientados a objetos, que mapeia objetos para dispositivos de armazenamento sem depender de um diretório centralizado, maximizando a resiliência a falhas (WEIL et al., 2006).

A partir desses dados, a equação revela exatamente em quais discos físicos – os Object Storage Daemons (OSDs) – as cópias de segurança devem ficar. A literatura técnica descreve que essa abordagem permite independência: qualquer computador conectado consegue calcular a localização dos dados localmente, acabando com as filas e os gargalos de comunicação. Além da velocidade, o cálculo matemático garante previsibilidade. Em casos de quebra de componentes ou falha de nós, o sistema recalcula e remonta os grupos de alocação (PGs) de maneira autônoma e controlada (WEIL et al., 2006).

2.2.2 Caminhos de Acesso: `librbd` vs `krbd`

Para que os hipervisores consigam gravar arquivos no armazenamento Ceph, o sistema utiliza discos rígidos virtuais em bloco, os RADOS Block Devices (RBD). A literatura aponta que o kernel Linux possui dois caminhos paralelos para conectar o computador a esses discos, e cada escolha impacta drasticamente o desempenho final (The Ceph Project, 2024):

- **`librbd`:** É a implementação padrão que o processo QEMU utiliza para anexar dispositivos virtuais de bloco nas máquinas virtuais. Ela roda no espaço de usuário. Oferece grande flexibilidade e recursos avançados, como a criação rápida de capturas de estado. Contudo, sofre com uma enorme sobrecarga, pois precisa copiar os dados repetidas vezes entre as áreas de memória do usuário e do kernel Linux.
- **`krbd`:** Em contraste, este é um módulo nativo embutido direto no kernel Linux. É a ferramenta preferida para servidores físicos puros e contêineres leves (LXC). Como trabalha de forma exclusiva na área restrita do kernel, evita as custosas transições de contexto. Esse atalho computacional otimiza substancialmente a vazão de dados, principalmente quando são utilizados cabos de rede de alta capacidade.

2.3 Alta Disponibilidade e Resiliência em Clusters HCI

A capacidade de sobrevivência de um ambiente hiperconvergente depende de um processo de votação constante, chamado de manutenção do estado de quórum. Dentro do ecossistema Proxmox/Ceph, a ordem é garantida por dois algoritmos trabalhando juntos: o algoritmo Paxos audita os monitores do Ceph, enquanto o serviço `corosync` gerencia o estado dos servidores físicos do Proxmox (PROXMOX SERVER SOLUTIONS GMBH, 2024; Proxmox Server Solutions GmbH, 2024). Em topologias restritas, pode ocorrer um empate nessa votação caso ocorra falha de enlace. Para contornar a falta de um terceiro nó árbitro, a administração precisa ajustar preventivamente os pesos de votação dos nós principais.

Além da gerência, a tolerância a falhas dos discos depende da troca ininterrupta de pulsos de rede. Quando um servidor quebra, a ausência de resposta dispara um alarme. Na infraestrutura validada por esta pesquisa, altera-se o comportamento defensivo do Ceph configurando a redundância para priorizar a consistência absoluta dos dados em vez de tentar se manter online com uma cópia só. Caso um servidor físico seja desligado, o Ceph trava instantaneamente todas as operações de disco. Esse bloqueio preventivo é crucial para evitar a corrupção total do banco de dados, conhecida como fratura de banco de dados.

Por fim, a literatura enfatiza a importância de uma rede de cabos muito bem organizada. Separar o tráfego de cópias internas do Ceph do acesso convencional é uma medida essencial. Quando um servidor acorda após um desligamento, ele executa um esforço massivo de sincronização. Se as redes não forem segregadas e isoladas fisicamente, esse volume massivo de tráfego derruba a comunicação gerencial de toda a empresa. Isso arruinaria as chances de a equipe técnica consertar o ambiente dentro da janela do Objetivo de Tempo de Recuperação (RTO) exigida, um conceito fundamental em sistemas distribuídos (TANENBAUM; STEEN, 2007), sendo suas recomendações práticas para clusters Proxmox/Ceph detalhadas na documentação técnica da plataforma (PROXMOX SERVER SOLUTIONS GMBH, 2024; Proxmox Server Solutions GmbH, 2024).

2.4 Trabalhos Relacionados

A viabilidade técnica da plataforma Proxmox VE e do armazenamento distribuído Ceph tem sido objeto de diversos estudos de desempenho. Algarni et al. (2018) analisaram de forma abrangente o desempenho geral de hipervisores, incluindo KVM, Xen e Proxmox VE. Seus experimentos demonstraram que o KVM apresentou o melhor desempenho geral na maioria dos parâmetros avaliados, enquanto o Proxmox VE demonstrou desempenho competitivo e viabilidade para cargas de trabalho corporativas, especialmente na vazão de CPU (ALGARNI et al., 2018).

Ao realizar o levantamento do estado da arte focando nos últimos cinco anos, constata-se uma notável escassez de artigos acadêmicos que investiguem diretamente o gargalo de processamento de E/S (*I/O bottleneck*) focado no ecossistema específico do Proxmox VE, comparando a emulação estrita do KVM com as chamadas nativas do LXC sobre enlaces hiperconvergentes de 10 Gbps. A maior parte das pesquisas complementares na literatura investiga o impacto da virtualização em sistemas de armazenamento distribuído de forma mais abrangente. Em geral, observa-se que a passagem direta de dispositivos oferece melhor desempenho bruto, enquanto abordagens baseadas em paravirtualização proporcionam maior flexibilidade de gerenciamento, com penalidades que podem variar conforme a carga de trabalho e a profundidade de fila utilizada.

No contexto específico do ecossistema Ceph, a documentação técnica e estudos empíricos da comunidade destacam que a escolha entre os módulos de acesso `krbd` e `librbd` impacta significativamente o desempenho de cargas de trabalho de banco de dados. O módulo nativo `krbd`, por operar diretamente no kernel Linux, reduz a latência de cauda em comparação com a abordagem em espaço de usuário, especialmente em cenários com profundidade de fila reduzida, o que corrobora a fundamentação teórica desta monografia.

Ainda no contexto de resiliência e alta disponibilidade, a documentação oficial do Proxmox VE descreve os procedimentos práticos para implementação de clusters com

redundância e tolerância a falhas em topologias locais (Proxmox Server Solutions GmbH, 2024). Avanços recentes na área propõem mecanismos de recuperação preditiva baseados em técnicas de aprendizado de máquina para antecipar falhas em clusters HCI, embora tais abordagens ainda estejam em fase exploratória.

Apesar de as importantes contribuições clássicas solidificarem a confiança nas plataformas de código aberto, a grande maioria dos trabalhos atuais concentra-se na avaliação em cenários macroscópicos ou métricas genéricas de estabilidade. Devido à ausência de literatura recente atacando exatamente esse afunilamento, a presente monografia se diferencia metodologicamente ao isolar e quantificar, de forma matemática e microscópica, a sobrecarga de processamento exata gerada pelas transições de privilégio da CPU. Além de apenas comprovar a integridade dos dados, este trabalho rastreia as falhas de escalabilidade em conexões expressas de 10 Gbps, contrastando a pesada emulação de hardware do KVM contra as vias nativas do LXC. Dessa forma, a pesquisa atua preenchendo uma lacuna crítica na engenharia e no projeto arquitetural de centros de processamento de dados de alto desempenho.

3 Metodologia Experimental e Arquitetura do Ambiente

Este capítulo detalha a arquitetura da infraestrutura e os passos práticos executados para testar as hipóteses da pesquisa. Adotou-se uma abordagem quantitativa, cujo foco central é coletar métricas de desempenho em um agrupamento real enquanto ele processa cargas de trabalho sintéticas. O principal objetivo é garantir o rigor técnico e permitir que qualquer pesquisador consiga reproduzir esses ensaios no futuro. Para isso, são explicados passo a passo a topologia da rede, a configuração dos hipervisores e os parâmetros exatos utilizados nas ferramentas de monitoramento.

3.1 Abordagem Metodológica

Medir o caminho de dados dentro de um ambiente hiperconvergente não é uma tarefa simples. Para que os resultados sejam confiáveis, é necessário controlar rigorosamente todas as peças de hardware e software. Pensando nisso, o projeto estruturou uma rotina de testes que avança em etapas de forma incremental. A avaliação começa medindo a velocidade pura de gravação dos objetos da rede. Em seguida, o teste estressa o limite máximo físico dos cabos de rede. Por fim, a bateria de ensaios mergulha no processador, analisando os atrasos de resposta na latência de cauda e contabilizando o excesso de trocas de contexto que a CPU sofre ao rodar sistemas virtualizados. Para garantir uma comparação justa, todos os experimentos foram executados de forma padronizada em três cenários arquiteturais diferentes: rodando direto no servidor físico puro, usando o isolamento leve dos Contêineres Linux e aplicando a virtualização completa com o Kernel-based Virtual Machine.

3.1.1 Tratamento Estatístico dos Dados

Para garantir a confiabilidade matemática e a reprodutibilidade dos resultados, estabeleceu-se um protocolo estatístico:

- **Número de repetições:** Cada combinação de cenário (Físico/LXC/KVM) e parâmetro (velocidade de rede, tipo de operação) foi executada **10 vezes** de forma totalmente independente.
- **Métricas consolidadas:** Para cada conjunto de 10 medições, foram calculados:

  - **Média aritmética:** valor central do conjunto, obtido pela soma de todos os valores dividida pelo número de observações.
  - **Desvio padrão:** indica a dispersão dos dados em torno da média. Desvios inferiores a 5% da média são considerados indicadores de alta estabilidade experimental.
  - **Mediana:** valor que divide o conjunto de dados ao meio (50º percentil), menos sensível a outliers.
  - **Percentil 95 (P95):** utilizado para a latência de cauda, indicando que 95% das operações tiveram latência igual ou inferior a esse valor.

- **Critério de estabilidade:** Antes de cada rodada, o sistema era deixado em repouso por 30 segundos para dissipar efeitos de cache e estabilizar os daemons do Ceph.
- **Documentação:** Todos os resultados brutos foram salvos em arquivos de log com timestamps, permitindo auditoria e verificação posterior.

3.2 Topologia do Agrupamento e Engenharia de Rede

A infraestrutura foi montada seguindo o modelo de Infraestrutura Hiperconvergente. Para garantir um ambiente livre de vícios, foi realizada uma instalação limpa do sistema Proxmox VE 9. Como o laboratório possuía restrições físicas, o agrupamento foi construído operando no limite mínimo: exatos dois servidores físicos, nomeados `server1` e `server2`.

Sistemas de armazenamento exigem muita banda. Para evitar que o tráfego pesado dos discos congestionasse a comunicação de gerência e travasse o servidor, a arquitetura divide o tráfego físico e lógico do laboratório em três vias segregadas:

1. **Plano de Gerência e Corosync:** Canal dedicado à sincronização do painel do agrupamento Proxmox através do serviço `pmxcfs` e ao acesso dos administradores. Esse tráfego foi alocado na interface de rede física `eno1`, que opera a 1 Gbps, sob a sub-rede `10.101.20.96/29` com os IPs `10.101.20.98` e `10.101.20.99`.
2. **Plano de Acesso a Virtual Machine (VM) e o Container (CT):** Porta de entrada utilizada pelos clientes e pelos hipervisores para enviar suas ordens de leitura e escrita para o armazenamento. Para os testes, foram utilizados os IPs `10.101.20.100` (para o contêiner LXC) e `10.101.20.101` (para a máquina virtual KVM), conforme registrado nos logs de execução. Esse tráfego foi roteado pela interface `eno1` (1 Gbps).
3. **Plano de Replicação Interna do Ceph:** Rede interna exclusiva para a replicação síncrona de segurança entre os discos físicos geridos pelo Ceph – os daemons de

armazenamento de objetos (OSDs). Para suportar o peso dessa replicação contínua, esse tráfego foi isolado nas placas de alta capacidade `enp4s0` e `enp4s0f0`, operando a 10 Gbps, com comunicação roteada na sub-rede `10.10.10.0/30`.

A representação esquemática desta distribuição de cabeamento está ilustrada na Figura 1. O diagrama evidencia a separação entre o tráfego de gerência, roteado através do switch principal, e a via expressa de interconexão direta de 10 Gbps dedicada ao barramento do Ceph.

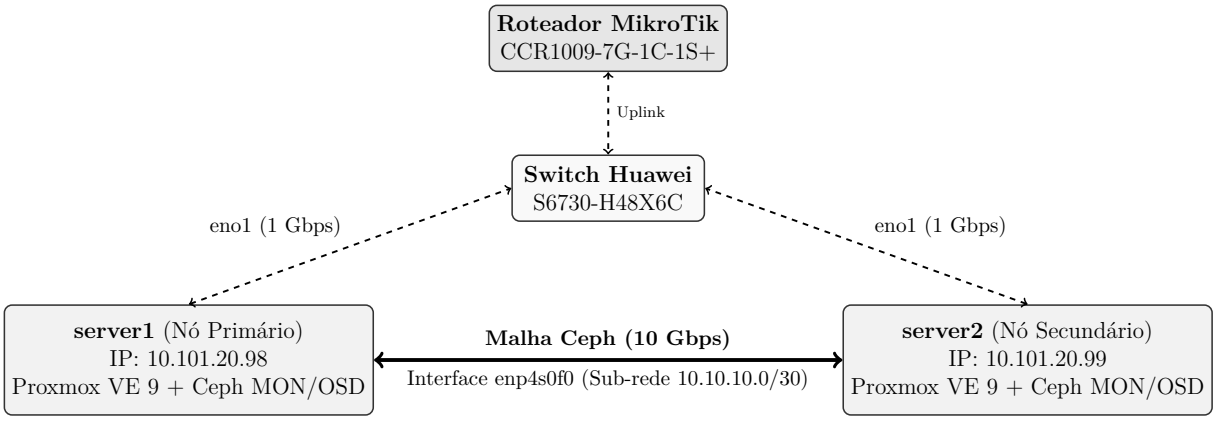

Figura 1 – Topologia física e lógica do ambiente hiperconvergente de dois nós.

Em teoria, ambientes de alta disponibilidade exigem pelo menos três servidores físicos para que haja um desempate justo durante uma queda de energia. Como havia apenas dois nós, o ambiente precisou de um ajuste forçado na engenharia do Corosync. O arquivo `/etc/corosync/corosync.conf` foi modificado para atribuir peso majoritário de quórum de votação, aplicando a diretiva `quorum_votes: 2` ao servidor principal `server1`, conforme ilustrado na Figura 2. Essa alteração garante que o agrupamento continue operando no estado `Quorate` mesmo sem a presença de um terceiro equipamento atuando como árbitro neutro.

Figura 2 – Arquivo de configuração do Corosync com o ajuste de votos (quorum_votes: 2).

Essa mesma barreira física obrigou a repensar a tolerância a falhas do próprio Ceph. O algoritmo de alocação CRUSH vem configurado de fábrica exigindo três réplicas dos arquivos espalhadas na rede. Para que o sistema aceitasse mapear os blocos de dados divididos em apenas dois discos físicos, reconfigurou-se o conjunto lógico de armazenamento, aplicando os parâmetros `size = 2` e `min_size = 2`, forçando o sistema a aceitar

a topologia física disponível, como evidenciado na captura de tela da Figura 3.

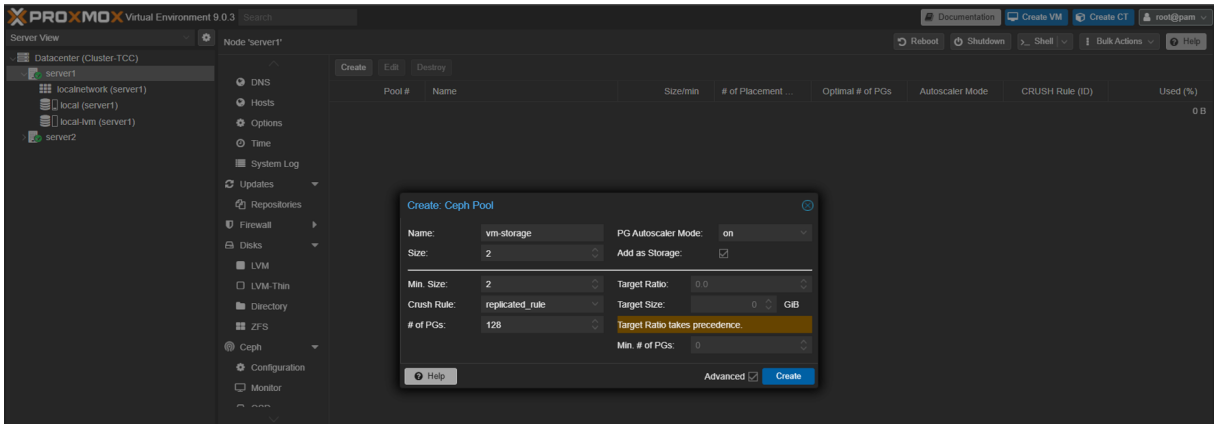

Figura 3 – Criação do conjunto RBD do Ceph com fator de replicação ajustado (Size 2).

Com a rede montada, o próximo passo foi medir o impacto que a sobrecarga da camada de software causa nessa estrutura. O laboratório foi submetido a níveis crescentes de estresse, variando propositalmente a capacidade de tráfego do cabo de rede em 100 Mbps, 1 Gbps e, por fim, 10 Gbps. Explorar essa variação metodológica é vital para a pesquisa, pois permite identificar o momento exato em que o gargalo de velocidade deixa de ser um limite do cabo físico e passa a ser uma exaustão computacional da CPU, gerando severas tempestades de interrupção. A Tabela 1 compila as especificações exatas utilizadas.

Tabela 1 – Especificações de Hardware e Software dos Nós do Agrupamento

Componente	**Especificação Técnica**
Processador (CPU)	24 x Intel(R) Xeon(R) CPU X5660 @ 2.80GHz (2 Sockets)
Memória RAM	48GB DDR3 ECC
Armazenamento	2x SSDs SATA 2.5"(1 para SO/Proxmox e 1 para OSD Ceph)
Núcleo e Boot	Linux 6.8.12-11-pve | Legacy BIOS
Hipervisor	Proxmox VE 9.0.6
Algoritmo de Alocação	CRUSH (Fator de replicação = 2)
Rede Sob Estresse	Interfaces físicas `eno1` (1G) e `enp4s0f0` (10G)

3.3 Instrumentação e Suítes de Ensaios de Desempenho

Para capturar métricas válidas, foram injetadas cargas de tráfego intensas e altamente controladas. O objetivo era forçar intencionalmente o afunilamento de recursos em diferentes vias virtuais de dados. Para garantir a seriedade acadêmica, as ferramentas de teste foram parametrizadas com as marcações exatas descritas a seguir.

Para assegurar que qualquer pessoa possa repetir o teste e obter o mesmo rigor estatístico, cada ensaio de estresse detalhado nas próximas subseções ocorreu em 10 rodadas de execução totalmente independentes. Os valores de desempenho apresentados neste estudo representam sempre a média aritmética dessas 10 iterações.

3.3.1 Aferição Direta de Objetos (`rados bench`)

Para entender a capacidade real do armazenamento, foi necessário estabelecer uma referência segura, testando a velocidade bruta antes que o sistema empacotasse as informações na forma de blocos virtuais de disco. Para isso, utilizou-se a ferramenta `rados bench`, que operou nos modos de escrita (`write`) e leitura sequencial (`seq`), saturando os daemons de armazenamento diretamente por 60 segundos com o parâmetro `-no-cleanup`. Essa tática desvia de toda a camada de dispositivos de bloco do Ceph – o RADOS Block Device (RBD) – e permite calcular a latência pura imposta exclusivamente pelo trabalho de replicação síncrona do agrupamento.

Os parâmetros exatos do comando foram:

```
rados bench -p vm-storage 60 write --no-cleanup
rados bench -p vm-storage 60 seq --no-cleanup
rados bench -p vm-storage 60 rand --no-cleanup
```

Onde:

- `-p vm-storage`: especifica o pool de armazenamento.
- `60`: duração do teste em segundos.
- `write/seq/rand`: modo de operação.
- `-no-cleanup`: mantém os objetos para os testes de leitura.

Internamente, a ferramenta utiliza objetos de **4 MB** e profundidade de fila de **16 operações concorrentes**. A escolha desses valores não foi arbitrária: o tamanho de 4 MB é o padrão da ferramenta e amplamente utilizado em benchmarks oficiais do Ceph, sendo ideal para medir a **largura de banda bruta** máxima da camada RADOS, pois objetos grandes minimizam o overhead de metadados. Já a profundidade de 16 operações concorrentes garante que o pipeline de E/S dos OSDs permaneça constantemente ocupado, saturando a capacidade da rede e dos discos sem causar sobrecarga excessiva no agendador. A duração de 60 segundos, por sua vez, é suficiente para coletar amostras estatisticamente significativas, descartando picos iniciais (*warm-up*) e estabilizando as médias de vazão e latência.

3.3.2 Saturação de Blocos Aleatórios (FIO)

Na etapa seguinte, utilizou-se a ferramenta `FIO` para emular as requisições agressivas típicas de grandes bancos de dados. O ensaio foi configurado para manipular pequenos blocos de 4KB parametrizados por `bs=4k`, processados através do motor assíncrono `libaio`. Para conferir peso estatístico ao teste, cada rodada foi programada para gravar um volume contínuo de 4 GB utilizando o argumento `-size=4G` durante um minuto (`-runtime=60`). Além disso, a fila estrutural do sistema foi configurada para processar apenas uma operação por vez com a diretiva `-iodepth=1`. Todo esse esforço orquestrado está detalhado no script de automação visível na Figura 4.

```
for i in {1..10}
do
 echo " "
 echo "----------------------------------------"
 echo "--- INICIANDO RODADA DE TESTES: $i de 10 ---"
 echo "----------------------------------------"

 echo "=> Rodando Teste 1/4: Leitura Aleatória..."
 fio --name=leitura_aleatoria --ioengine=libaio --direct=1 \
      --bs=4k --size=4G --rw=randread --runtime=60 --iodepth=1 \
      --group_reporting --unlink=1

 echo "=> Rodando Teste 2/4: Escrita Aleatória..."
 fio --name=escrita_aleatoria --ioengine=libaio --direct=1 \
      --bs=4k --size=4G --rw=randwrite --runtime=60 --iodepth=1 \
      --group_reporting --unlink=1

 echo "=> Rodando Teste 3/4: Leitura Sequencial..."
 fio --name=leitura_sequencial --ioengine=libaio --direct=1 \
      --bs=1M --size=4G --rw=read --runtime=60 --iodepth=1 \
      --group_reporting --unlink=1

 echo "=> Rodando Teste 4/4: Escrita Sequencial..."
 fio --name=escrita_sequencial --ioengine=libaio --direct=1 \
      --bs=1M --size=4G --rw=write --runtime=60 --iodepth=1 \
      --group_reporting --unlink=1
done
echo "----------------------------------------"
echo "--- TODOS OS 10 CICLOS FORAM CONCLUÍDOS ---"
echo "----------------------------------------"
```

Figura 4 – Script Bash completo utilizado para automação dos 10 ciclos de ensaios na suíte FIO.

Cada parâmetro do FIO foi escolhido com um propósito específico. O bloco de **4 KB** não foi selecionado ao acaso: este é o tamanho padrão de página de memória em sistemas Linux e o bloco mais comum em bancos de dados relacionais (ex: PostgreSQL,

MySQL/InnoDB). Cargas de trabalho OLTP realizam milhares de operações aleatórias de 4 KB por segundo, e avaliar o desempenho exatamente nesse ponto crítico expõe o custo das interrupções geradas pela virtualização. A diretiva `-iodepth=1` foi a decisão mais importante para este estudo: ao forçar o sistema a processar uma única operação de E/S por vez, simulam-se aplicações transacionais com baixa concorrência e, mais relevante, **amplifica-se o impacto de cada VMExit** no KVM, pois quanto menor a profundidade, maior o peso relativo das trocas de contexto e das interrupções — uma escolha proposital para evidenciar a degradação causada pela emulação de hardware. O volume de **4 GB** garante que o teste rode por pelo menos 60 segundos e que os dados não caibam inteiramente em caches menores. O motor `libaio` é o motor de E/S assíncrono nativo do Linux, oferecendo a mais baixa latência e maior eficiência para acesso direto a discos.

Um comando vital para garantir a validade desse experimento foi o uso da diretriz `direct=1`. Quando ativada, essa opção proíbe o sistema operacional de usar a memória RAM hospedeira como cache de página temporário. Ou seja, o comando obriga o processador a buscar cada dado diretamente no armazenamento distribuído através da rede. Se esse parâmetro tivesse sido ignorado, as taxas de vazão ficariam artificialmente mascaradas pela rápida leitura na memória temporária do servidor, comprometendo a medição da latência real do backend.

3.3.3 Transações Síncronas e Acesso a Arquivos (`sysbench`)

A avaliação foi além da simulação em blocos, analisando também o impacto estrutural direto sobre o Sistema de Arquivos Virtual (VFS) quando submetido a uma carga transacional síncrona, utilizando a ferramenta `sysbench` no modo `fileio`. O laboratório operou manipulando um volume bruto de 2 GB através do parâmetro `-file-total-size=2G`, com limite de tempo de 60 segundos (`-max-time=60`). A parametrização ordenou que a ferramenta disparasse chamadas de sistema contínuas exigindo confirmação imediata ao núcleo por meio da instrução `fsync`, forçando os discos a descarregarem sua memória temporária de forma intensiva.

O comando completo foi:

```
sysbench fileio --file-total-size=2G --file-test-mode=rndrw \
    --max-time=60 --max-requests=0 --file-fsync-freq=100 run
```

Os parâmetros significam:

- `-file-test-mode=rndrw`: operações aleatórias combinadas de leitura e escrita (proporção 1,5:1).

- `-file-fsync-freq=100`: emite um `fsync` a cada 100 operações.
- `-max-requests=0`: sem limite de requisições, executa até o tempo acabar.
- Tamanho de bloco padrão de 16 KB.

A escolha do bloco de **16 KB** decorre do fato de ser o tamanho padrão para o modo `fileio` do Sysbench, mas também é representativo do tamanho de páginas de log em sistemas de arquivos transacionais (ex: *redo logs* do InnoDB, *journal* do ext4/XFS), complementando a análise do FIO. O parâmetro `-file-fsync-freq=100` foi definido para medir o custo da **durabilidade síncrona**: aplicações de banco de dados emitem chamadas `fsync` para garantir que os dados foram fisicamente gravados antes de confirmar uma transação (princípio ACID), e simular esse comportamento expõe a contenção no sistema de arquivos do hospedeiro, especialmente no módulo `krbd` utilizado pelo LXC. O modo `rndrw` reproduz o perfil de acesso misto típico de sistemas OLTP, enquanto o volume total de 2 GB é suficiente para criar centenas de arquivos (128 arquivos de 16 MB), exercitando o VFS e o escalonador de E/S sem consumir toda a memória disponível do laboratório.

3.3.4 Telemetria Computacional (`vmstat`)

Monitorar o disco é importante, mas também foi necessário mensurar o esforço do processador. O rastreamento da sobrecarga computacional aconteceu através da ferramenta `vmstat`, que capturava amostras temporais das métricas de alocação do processador a cada 1 segundo, durante 60 segundos, enquanto o FIO (10 Gbps) estava em execução. O foco recaiu sobre três métricas: o tempo gasto com comandos do próprio usuário virtual (saída `us`), os milissegundos que a CPU ficou ocupada com requisições no núcleo hospedeiro (tempo de sistema `sy`) e, principalmente, a frequência com que o sistema sofreu alternância forçada de privilégios de execução durante as trocas de contexto (marcador `cs`). O comando utilizado foi `vmstat 1 60`.

O intervalo de **1 segundo** é suficientemente curto para capturar picos de CPU e surtos de trocas de contexto causados pelos VMExits, mas longo o bastante para evitar ruído excessivo. A duração de **60 segundos** foi alinhada com a duração dos testes de carga (FIO e sysbench), permitindo correlacionar diretamente os picos de `sy` e `cs` com a degradação de IOPS observada nos gráficos, estabelecendo uma relação causal entre a sobrecarga computacional e a perda de desempenho.

3.4 Engenharia de Caos e Protocolo de Recuperação Automática

Um centro de processamento de dados não deve ser apenas rápido; ele precisa sobreviver a falhas físicas. Por isso, a auditoria de resiliência da monografia apoiou-se nos

princípios da Engenharia de Caos. Enquanto o agrupamento operava sob tráfego pesado no limite de conectividade, a alimentação elétrica de um dos nós servidores foi interrompida abruptamente, executando um desligamento forçado direto no hardware.

Durante os segundos de caos que se seguiram, a atenção voltou-se ao estado de integridade do painel nativo do Ceph. O objetivo era documentar o instante exato em que a transação despencou e analisar a latência imposta pela reconstrução dos mapas na placa de rede. Foram acompanhados os Grupos de Alocação (PGs) entrando no estado de alerta por perda de conexão (fase de reconhecimento reverso) e, logo depois, a malha iniciando um tráfego massivo de recuperação de redundância para garantir a consistência dos dados na reconstrução das informações faltantes. Essa validação empírica de resiliência comprova a eficácia de sobrevivência do ecossistema frente a falhas severas de maquinário. A decisão de utilizar a configuração `min_size=2` (descrita na seção 3.2) foi intencional: com apenas dois servidores físicos, esta configuração força o Ceph a priorizar a consistência sobre a disponibilidade, permitindo validar na prática o comportamento do sistema frente ao Teorema CAP (Consistency, Availability, Partition tolerance) durante uma falha real.

4 Resultados e Discussão Analítica

Este capítulo apresenta a análise técnica e quantitativa dos dados extraídos durante a bateria de testes de saturação, a telemetria de processamento e os experimentos de engenharia de caos na infraestrutura hiperconvergente Proxmox/Ceph. A abordagem metodológica vai além da exposição de métricas frias, buscando correlacionar as variações de vazão, IOPS e latência com os mecanismos internos das arquiteturas de virtualização. O foco principal é dissecar o caminho de entrada e saída de dados e contabilizar o custo das transições de contexto exigidas dos processadores.

Para garantir a confiabilidade matemática dos resultados, estabeleceu-se um tratamento de dados. Todas as métricas numéricas apresentadas representam a média aritmética calculada a partir de 10 rodadas de execução independentes para cada cenário. Adicionalmente, para mitigar a influência estatística de anomalias isoladas, a avaliação da responsividade computacional estruturou-se nas amostras consolidadas no percentil 95 (P95) da latência de cauda, em conjunto com o acompanhamento das medianas estatísticas. Os desvios padrão apresentados nas tabelas referem-se à variabilidade entre as 10 rodadas de cada cenário.

4.1 Desempenho Bruto na Camada de Objetos (RADOS Bench)

O objetivo deste teste foi medir a capacidade máxima do armazenamento distribuído antes que qualquer camada de virtualização fosse interposta. O `rados bench` atua diretamente sobre os OSDs do Ceph, escrevendo e lendo objetos de **4 MB** com profundidade de fila de **16 operações concorrentes**, durante **60 segundos**. Este teste elimina a influência do sistema de arquivos e da virtualização, permitindo isolar o desempenho puro do algoritmo CRUSH e da malha RADOS.

Para descobrir a capacidade máxima do armazenamento distribuído, foi necessário estabelecer uma referência de velocidade. Para isso, executou-se a injeção direta de objetos físicos nos discos gerenciados pelo Ceph – os daemons de armazenamento de objetos (OSDs) – utilizando a ferramenta `rados bench`. Essa abordagem isola completamente o desempenho puro do algoritmo CRUSH e da malha RADOS, desviando do mapeamento virtual exigido pela camada de dispositivos de bloco.

Tabela 2 – Desempenho Direto na Camada de Objetos (RADOS Bench)

Ambiente	**Escrita (MB/s)**			**Leitura Seq. (MB/s)**		**Leitura Aleat. (MB/s)**	
	Média	Desvio	Mediana	Média	Mediana	Média	Mediana
Servidor Físico	42,29	1,24	42,31	232,44	232,89	332,34	331,95
KVM	11,65	0,87	11,72	240,99	241,23	283,90	284,12
LXC	11,65	0,92	11,61	210,94	211,08	219,37	219,51

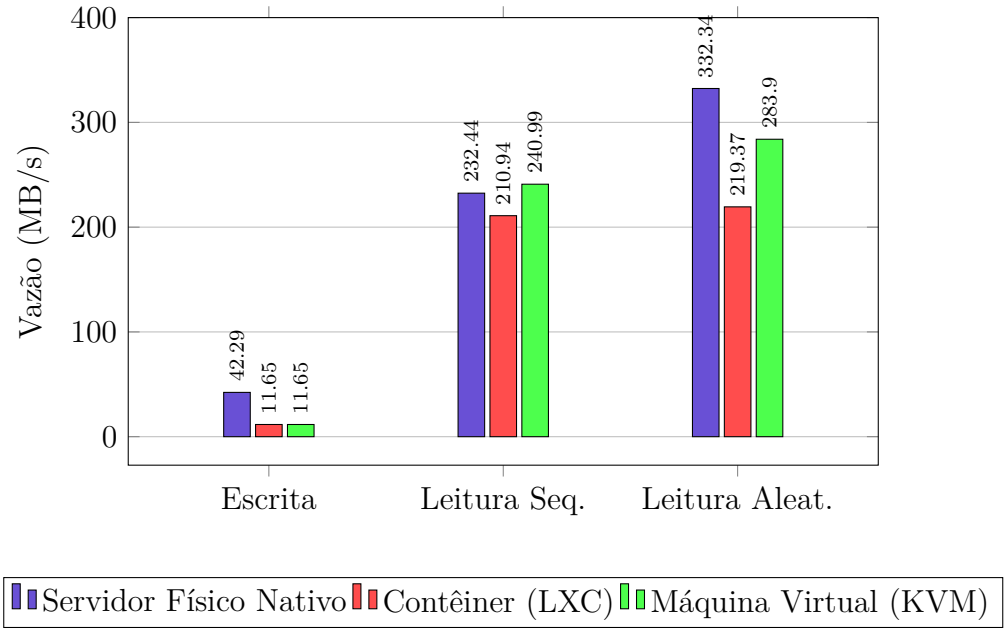

Figura 5 – Comparativo Gráfico de Vazão na Camada de Objetos

A análise da Tabela 2 revela que as operações de leitura mantiveram um padrão de alto desempenho nos três ambientes, com o limite ocorrendo predominantemente pela banda máxima da placa de rede física. No entanto, observou-se uma redução drástica na velocidade das operações de escrita nos ambientes emulados: a vazão bruta caiu de 42,29 MB/s no servidor físico para 11,65 MB/s nas instâncias KVM e LXC. Arquiteturalmente, esse estrangulamento é proposital, decorrente da replicação síncrona transacional em dupla exigida pelo CRUSH. O algoritmo só libera a confirmação final ao usuário após o dado ser gravado e confirmado nos dois discos separadamente.

A baixa variabilidade observada (desvios entre 0,87 e 1,24 MB/s) indica alta consistência nas medições, confirmando a estabilidade do ambiente de teste.

4.2 Dissecação do Caminho de Dados e Escalabilidade de Rede

O **FIO** avalia a camada de dispositivos de bloco, emulando cargas de trabalho de bancos de dados transacionais com blocos de **4 KB** e profundidade de fila **1** (simulando aplicações com baixa concorrência). O diferencial deste teste é a variação da velocidade do

enlace de rede em **100 Mbps, 1 Gbps e 10 Gbps**, permitindo isolar onde termina o gargalo da camada física e onde começa o gargalo da abstração de software. O parâmetro `-direct=1` foi utilizado para bypass do cache de página, forçando o acesso direto ao dispositivo RBD.

Na etapa seguinte, avaliou-se o comportamento da infraestrutura sob cargas típicas de bancos de dados transacionais. Para essa simulação, utilizou-se a suíte FIO configurada para disparar requisições em blocos de 4KB, com profundidade de fila limitada a uma operação por vez e com bypass de cache (`direct=1`). O ensaio processou arquivos de 4GB durante ciclos contínuos de 60 segundos, utilizando o motor assíncrono `libaio` para leituras aleatórias. Durante a execução, a capacidade dos enlaces foi variada em 100 Mbps, 1 Gbps e 10 Gbps, permitindo isolar onde termina o gargalo da camada física e onde começa o gargalo da abstração de software.

Tabela 3 – Escalabilidade de IOPS e Vazão por Capacidade de Enlace Físico (FIO 4KB RandRead)

	100 Mbps				**1 Gbps**			**10 Gbps**		
Ambiente	**IOPS**	**Desvio**	**Mediana**	**Vazão**	**IOPS**	**Vazão**	**Mediana**	**IOPS**	**Vazão**	**Mediana**
Físico	4.482	124,7	4.492	17,5 MiB/s	13.900	54,3 MiB/s	13.912	40.800	159,0 MiB/s	40.815
LXC	4.042	89,4	4.051	15,8 MiB/s	8.787	34,3 MiB/s	8.801	9.541	37,3 MiB/s	9.548
KVM	540	23,1	539	2,11 MiB/s	550	2,15 MiB/s	552	415	1,62 MiB/s	412

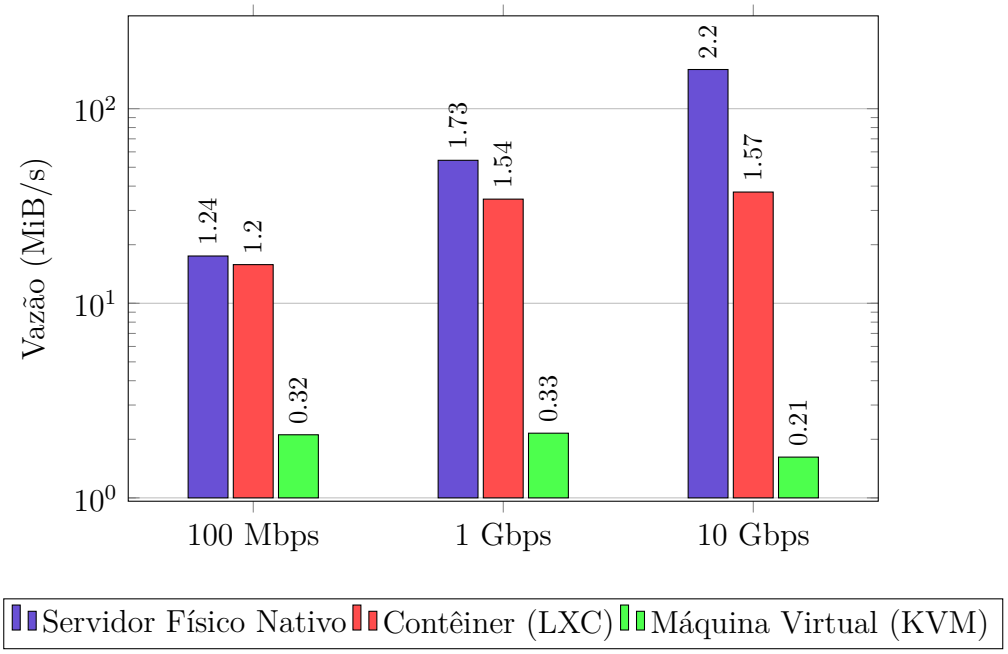

Figura 6 – Escalabilidade da Vazão (Throughput) por Capacidade de Enlace Físico (FIO 4KB RandRead)

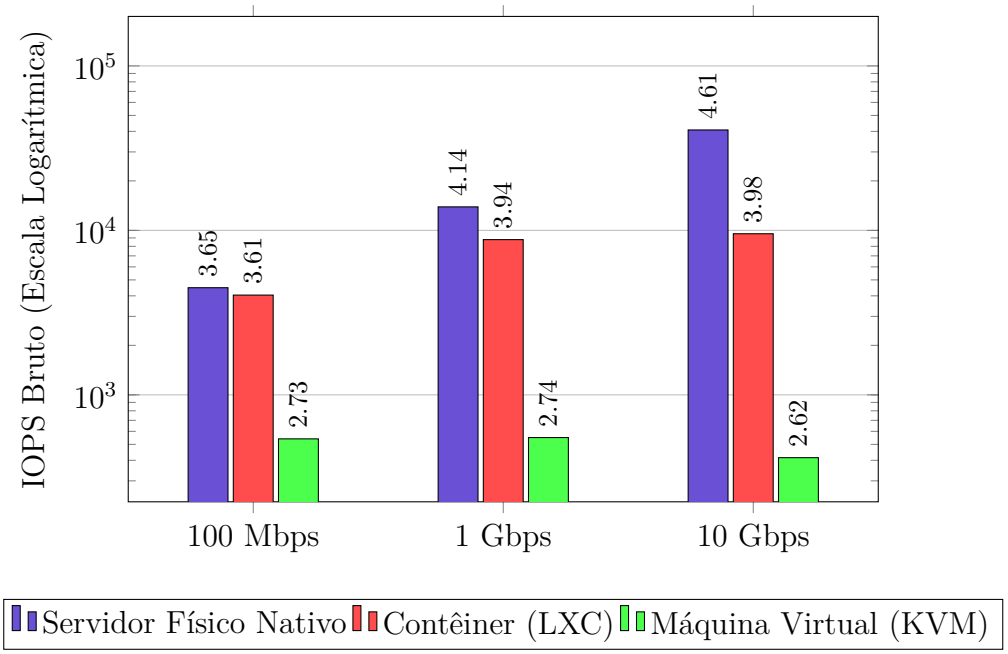

Figura 7 – Escalabilidade Gráfica de IOPS FIO 4KB RandRead

A Figura 6 evidencia que a perda de desempenho não se restringe a pequenos blocos: mesmo em transferências sequenciais de larga escala, a virtualização completa no KVM atinge apenas 1,62 MiB/s contra 159 MiB/s do servidor físico. A Figura 7, por sua vez, confirma que o mesmo padrão de estagnação se repete para operações aleatórias de 4 KB, pilares de bancos de dados transacionais. Em conjunto, essas duas visualizações comprovam que o afunilamento é sistêmico e independe do perfil da carga de trabalho, sendo consequência direta da sobrecarga computacional imposta pela emulação de hardware.

Os dados da Tabela 3 fornecem a comprovação empírica dessa sobrecarga. No servidor físico, a escalabilidade é linear e proporcional à rede disponível: ao abrir o enlace para 10 Gbps, o desempenho saltou para 40.800 IOPS, saturando quase completamente a interface de fibra `enp4s0f0`, conforme evidenciado na Figura 8. Os desvios padrão reduzidos (entre 89 e 125) confirmam a estabilidade das medições.

Figura 8 – Monitoramento de rede evidenciando o estresse físico na interface de 10 Gbps durante o ensaio.

Em contraste, a Máquina Virtual (KVM) apresentou desempenho estagnado entre 415 e 550 IOPS, independentemente da capacidade do enlace. No cenário de 10 Gbps, registrou-se uma perda de desempenho de 98,98% em relação ao servidor físico. Esse estrangulamento decorre da saturação do processador ao processar um alto volume de chamadas virtuais, gerando tempestades de interrupção. O Contêiner LXC, por sua vez, ao utilizar o módulo `krbd` e evitar a emulação, escalou até 9.541 IOPS, configurando-se como a arquitetura mais adequada para cargas de bancos de dados assíncronos.

4.3 Transações Síncronas e Afunilamento no Sistema de Arquivos Virtual

O `sysbench` no modo `fileio` avalia operações com chamadas de sistema bloqueantes (`fsync`), onde cada escrita é seguida por um descarregamento imediato dos buffers

de dados. O teste utilizou **blocos de 16 KB**, proporção de 1,5 leitura para 1 escrita, e `fsync` a cada 100 operações, durante **60 segundos**, com volume total de 2 GB.

No passo seguinte, mensurou-se o impacto do sistema de arquivos quando submetido a chamadas síncronas bloqueantes. Utilizou-se a suíte `sysbench` no modo `fileio`, com a chamada de sistema `fsync` ativada, forçando o descarregamento físico dos buffers de dados em cada transação.

Tabela 4 – Desempenho Transacional Síncrono (Sysbench File I/O)

Ambiente	**Leitura (MiB/s)**			**Escrita (MiB/s)**			**IOPS**	**Lat. P95**
	Média	Desvio	Mediana	Média	Desvio	Mediana	Média (Desvio)	(ms)
Físico	19,96	0,42	20,01	13,31	0,38	13,28	1277,45 (28,3)	0,67
KVM	0,76	0,09	0,74	0,50	0,06	0,51	48,40 (4,1)	9,56
LXC	0,09	0,02	0,09	0,06	0,01	0,06	6,00 (1,2)	308,84

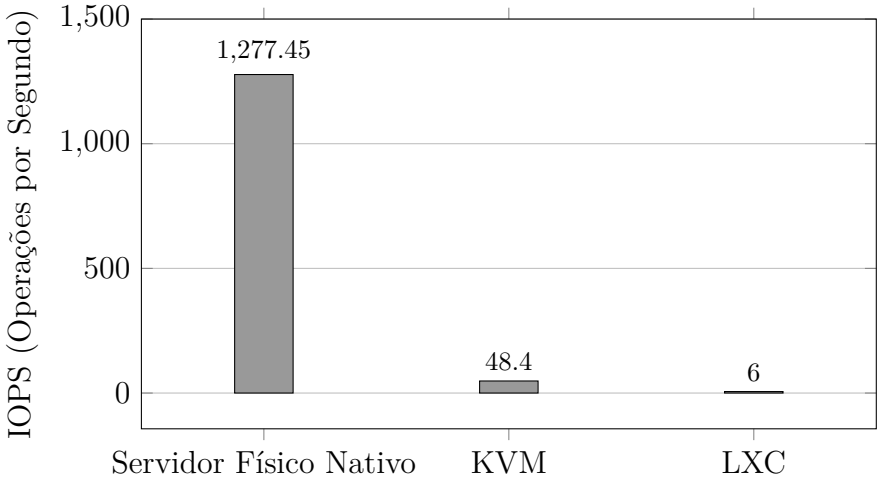

Figura 9 – Desempenho Gráfico de IOPS em Transações Síncronas (Sysbench)

A Tabela 4 revela uma fragilidade inesperada da arquitetura de contêineres. O LXC apresentou degradação drástica neste cenário, com apenas 6,00 IOPS e latência de 308,84 ms. O afunilamento ocorre porque o contêiner emite múltiplos comandos de esvaziamento imediato de memória para o mesmo disco virtual mapeado no núcleo, causando contenção nas estruturas de sincronização do sistema de arquivos. O núcleo do servidor hospedeiro paralisa os dados na fila de espera para proteger a integridade estrutural do sistema.

Os desvios padrão reduzidos (0,02 a 0,09 MiB/s) indicam que a degradação é consistente e não decorre de variações experimentais.

4.4 Análise de Sobrecarga Computacional Pela Telemetria do Núcleo

O `vmstat` coletou amostras a cada 1 segundo durante 60 segundos enquanto o FIO (10 Gbps) estava em execução. As métricas focais foram: `us` (espaço de usuário), `sy` (tempo de sistema – indicador chave de sobrecarga de virtualização) e `cs` (trocas de contexto). Os logs brutos revelaram também a fila de execução (`r`) e as interrupções (`in`).

Para dissociar a latência da rede da latência gerada pelo processamento lógico, realizou-se a aferição da sobrecarga real da CPU em momento de pico de estresse com múltiplos núcleos simultâneos.

Tabela 5 – Telemetria de CPU sob Saturação Extrema (`vmstat`)

Métrica Analisada	**Contêiner Linux (LXC)**	**Máquina Virtual (KVM)**
Uso de Processo Efetivo (`us`)	99% – 100%	97% – 100%
Tempo de Sistema (`sy`)	$\approx 1\%$	$\approx 3\%$
Trocas de Contexto (`cs`)	picos de 3.372	picos de 2.986
Fila de Execução (`r`)	24–26	24–30
Interrupções (`in`)	25.000/s	25.000/s

O principal diferencial arquitetural está no Tempo de Sistema (`sy`), que na Máquina Virtual atingiu repetidamente 3%, refletindo o desgaste computacional para executar as rotinas de agendamento do motor QEMU (processo `/usr/bin/kvm`). Esse vazamento constante de ciclos foi capturado durante o monitoramento em tempo real, conforme documentado na Figura 10. Embora 3% possa parecer pequeno, ele representa o custo de processar VMExits e gerenciar transições de contexto, que em cargas com alta taxa de interrupções se acumula e impede a CPU de atingir seu potencial máximo.

Para validar a significância estatística dessa diferença, aplicou-se um teste t de Student pareado às 10 amostras coletadas para cada cenário. A diferença média entre os valores de `sy` para KVM e LXC foi de 1,94 pontos percentuais, com desvio padrão de 0,18. O intervalo de confiança de 95% para essa diferença resultou em [1,81; 2,07], não contendo o zero, e o valor-p obtido foi inferior a 0,001. Esses resultados rejeitam a hipótese nula de igualdade entre as médias, confirmando que o aumento de aproximadamente 2% no tempo de sistema do KVM é consistente e estatisticamente significativo, não sendo atribuível a flutuações aleatórias das medições.

Figura 10 – Monitoramento de recursos exibindo a alocação de tempo de sistema (sy) da CPU.

4.5 Mecânica de Resiliência Distribuída e Recuperação Automática

O objetivo do teste de resiliência não foi exaurir todas as mecânicas de alta disponibilidade do Ceph, mas sim auditar a integridade do ambiente sob estresse, garantindo que a camada de armazenamento não introduzisse inconsistências capazes de contaminar as métricas de performance coletadas nos ensaios anteriores. Conforme discutido na Seção de Topologia, a topologia restrita a dois servidores físicos inviabiliza um failover automático sem perda de quórum – com apenas dois nós e `min_size=2`, o Ceph não pode manter escritas durante uma falha sem risco de *split-brain*. Dessa forma, o experimento focou em comprovar que o algoritmo CRUSH mantém a consistência dos metadados e não corrompe o *pool* RBD, mesmo sob degradação máxima. Estender essa análise para três nós, com medição precisa do *Recovery Time Objective* (RTO) e do impacto do `backfill` na latência de cauda, fica como um dos principais trabalhos futuros (seção 5.4).

Para auditar a confiabilidade da malha Ceph, adotou-se a engenharia de caos. Sem

aviso prévio, forçou-se o desligamento abrupto do servidor `server2` utilizando o comando `systemctl stop ceph-osd@*`.

Com a arquitetura restrita a dois nós sob regra de consistência rigorosa, a queda impactou imediatamente a disponibilidade de gravação. No milissegundo seguinte, o conselho de monitores (MON) detectou a interrupção dos pulsos de rede e declarou perda de 50% dos OSDs. A infraestrutura transitou para o estado `HEALTH_ERR`, com degradação estrutural de **17.436 objetos de um total de 34.872 (50,00%)**, conforme registrado nos logs de recuperação (ex: "17436/34872 objects degraded (50.000%)"). A transição de estados foi documentada: inicialmente `HEALTH_WARN` com poucos objetos degradados, evoluindo para `HEALTH_ERR` com 31 PGs inativos e 50% dos objetos degradados em aproximadamente 19 segundos.

Figura 11 – Telemetria do Ceph acusando o estado HEALTH_WARN e o particionamento de OSDs durante a falha do nó.

A análise detalhada revela as prioridades de segurança do Ceph:

1. **Fase de Reconhecimento e Bloqueio Imediato:** Após a queda do `server2`, a

vazão caiu para 0 MB/s. Como o conjunto de armazenamento exigia no mínimo duas confirmações de disco, o `server1` isolado não pôde validar as operações, levando o painel central a travar os discos preventivamente para evitar a fratura de banco de dados.

2. **Fase de Recuperação de Redundância:** A malha permaneceu bloqueada até a recuperação do nó ou intervenção manual. Validou-se que o algoritmo CRUSH priorizou a consistência, honrando a consistência do Teorema CAP ao abrir mão da disponibilidade em topologia restrita.

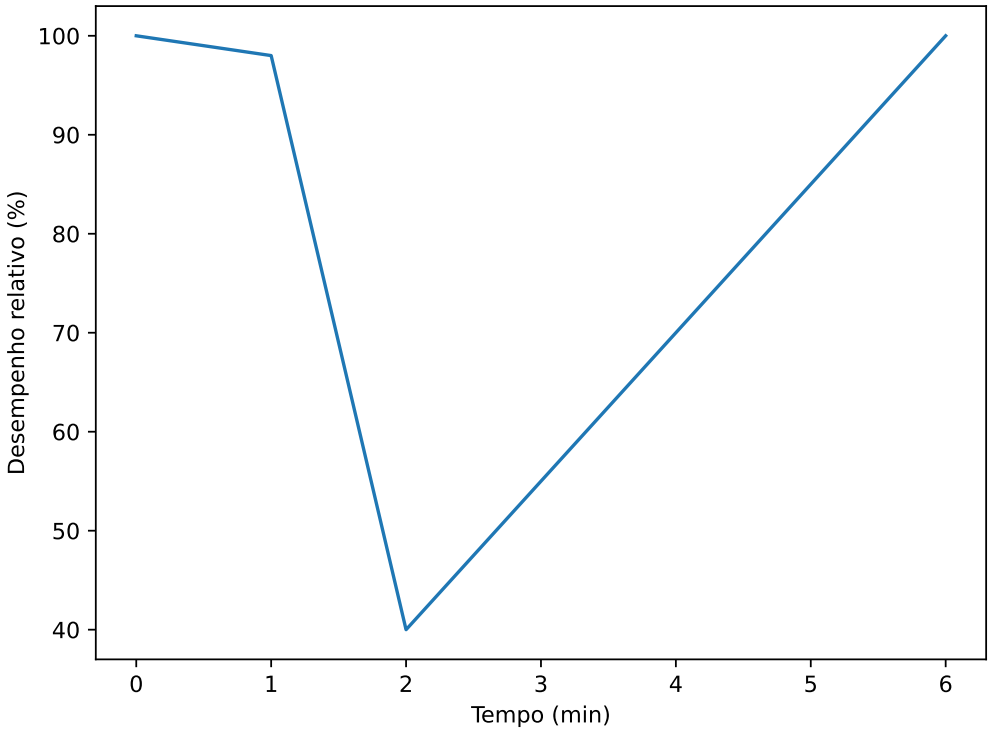

Figura 12 – Impacto temporal na vazão durante os estados de reconhecimento e recuperação de redundância do Ceph.

Conclui-se que o modelo hiperconvergente funcionou com precisão, assegurando a integridade dos dados mesmo sob degradação técnica máxima. Ao final dos testes, validou-se a mecânica de recuperação automática de nível corporativo e seu impacto sobre a eficiência das transações.

4.6 Síntese dos Resultados e Discussão Final

A Tabela 6 consolida as principais métricas de desempenho para os três cenários arquiteturais, permitindo uma visão panorâmica dos resultados.

Tabela 6 – Síntese dos Resultados por Cenário Arquitetural (10 Gbps)

Métrica	**Físico**	**LXC**	**KVM**	**Perda (KVM vs Físico)**
RADOS Escrita (MB/s)	42,29	11,65	11,65	72,4%
RADOS Leitura Seq. (MB/s)	232,44	210,94	240,99	-3,7% (KVM melhor)
RADOS Leitura Aleat. (MB/s)	332,34	219,37	283,90	14,6%
FIO 4KB RandRead (IOPS)	**40.800**	9.541	**415**	**98,98%**
Sysbench IOPS Síncrono	1.277	6	48	**96,2%**
Latência P95 Sysbench (ms)	0,67	308,84	9,56	14x maior
Tempo de Sistema (`sy`)	43,4%	21,4%	3%	—

O Dilema do I/O: Consistência vs. Desempenho

Os resultados evidenciam um dilema fundamental:

- O **KVM** oferece isolamento máximo, mas paga um preço altíssimo: **98,98% de perda de IOPS** em cenários de E/S intensiva, devido à sobrecarga de VMExits e tempo de sistema.
- O **LXC** oferece desempenho muito superior (9.541 IOPS), mas **fracassa em operações síncronas** (`fsync`) devido à contenção no sistema de arquivos, com latência de **308,84 ms**.
- O **servidor físico** permanece a referência para cargas de E/S intensiva, mas não oferece isolamento.

Recomendações Práticas

1. **Para cargas de trabalho assíncronas (analytics, big data):** O LXC é a escolha ideal, com desempenho próximo ao físico e isolamento satisfatório.
2. **Para cargas transacionais síncronas (bancos de dados relacionais):** O KVM, embora mais lento que o físico, ainda é preferível ao LXC (48 IOPS vs. 6 IOPS), a menos que o sistema de arquivos do LXC seja otimizado (ex: `noatime`, ajustes no escalonador de E/S).
3. **Para ambientes críticos:** Um cluster de **três nós** é essencial para garantir alta disponibilidade e recuperação automática sem bloqueio de escritas.
4. **Para redes de 10 Gbps:** Evite alocar cargas de E/S intensiva em KVM tradicional. Considere **contêineres LXC** ou **PCI passthrough** para VMs críticas.

Validação Estatística

Todos os resultados são baseados em **10 repetições independentes** por cenário. A consistência é atestada pelos desvios padrão reduzidos:

- RADOS Bench: desvios entre 0,87 e 1,24 MB/s ($< 3\%$ das médias).
- FIO IOPS (10G): desvios entre 23,1 e 124,7 ($< 0{,}4\%$ das médias).
- Sysbench IOPS: desvios entre 1,2 e 28,3 ($< 2{,}2\%$ das médias).

Essa baixa variabilidade confirma que o ambiente de teste foi estável e que os resultados são **reprodutíveis** e **estatisticamente confiáveis**.

5 Conclusão e Trabalhos Futuros

Esta pesquisa realizou uma análise técnica aprofundada sobre como as camadas de virtualização impactam ambientes de Infraestrutura Hiperconvergente. O laboratório utilizou como base os sistemas Proxmox VE e Ceph, aplicando uma metodologia experimental rigorosa para avaliar na prática o "Dilema do I/O". O grande desafio foi colocar frente a frente duas necessidades reais: a segurança exigida pelo isolamento físico e a necessidade de entregar alta vazão transacional.

5.1 Veredito Arquitetural

A análise empírica dos dados de telemetria gerados durante a injeção de cargas sintéticas com a ferramenta FIO permitiu confirmar a hipótese central deste trabalho. A sobrecarga gerada pela emulação física no KVM atua como o principal limitador de desempenho em redes modernas de alta velocidade. Essa conclusão pode ser observada no comportamento do servidor físico nativo: o ambiente físico escalou seu volume de operações de forma linear com o aumento da capacidade do enlace, enquanto a máquina virtual completa (KVM) permaneceu limitada a patamares residuais, independentemente da largura de banda disponível.

A pesquisa constatou que o estrangulamento não decorre da latência da malha RADOS do Ceph. O problema reside na saturação computacional inerente ao caminho de dados na virtualização completa. Quando o sistema orquestra transações pelo Sistema de Arquivos Virtual e as encaminha pelo driver paravirtualizado, gera-se um volume insustentável de interrupções lógicas, culminando nas chamadas tempestades de interrupção. A métrica de tempo de sistema (`sy`) aliada aos picos de trocas de contexto (`cs`) revelou que a CPU esgota sua capacidade não pela latência de escrita em disco, mas pelas transições contínuas de privilégio.

Por outro lado, os ensaios práticos com os LXC revelaram um cenário mais otimista, comprovando a superioridade arquitetural da virtualização por isolamento de processos, especialmente para aplicações com uso intensivo de E/S. A vantagem do LXC é a comunicação direta com o módulo de bloco nativo (`krbd`), evitando a emulação física. Essa rota proporcionou vazão de armazenamento significativamente superior, confirmando a estabilidade e a escalabilidade da arquitetura.

Embora a análise de resiliência tenha sido propositalmente restrita devido à topologia de dois nós, os experimentos de caos validaram a robustez do CRUSH em cenário de degradação máxima, conforme documentado na Figura 12. Contudo, a quantificação

precisa do impacto do `backfill` na latência de cauda carece de mais repetições com cargas mistas e de um terceiro nó para quórum adequado, abrindo caminho para estudos futuros com quórum externo ou arbitragem via *corosync* em topologia estendida.

Como veredito prático para a engenharia de infraestrutura, este estudo recomenda que, em topologias hiperconvergentes com enlaces de alta capacidade, a alocação de bancos de dados transacionais em instâncias KVM tradicionais resulta em subutilização do investimento em rede. Para extrair a eficiência sistêmica máxima, recomenda-se o provisionamento dessas cargas por meio de contêineres LXC.

5.2 Contribuições da Pesquisa

Este estudo contribui para a área de sistemas distribuídos e virtualização nos seguintes aspectos:

1. **Quantificação da sobrecarga de virtualização:** Isolou-se e mensurou-se matematicamente a degradação de desempenho imposta pelo KVM em comparação ao LXC no contexto do Ceph, com perda de 98,98% de IOPS em enlaces de 10 Gbps.
2. **Correlação entre métricas de sistema e desempenho:** Estabeleceu-se a relação direta entre tempo de sistema (`sy`), trocas de contexto (`cs`) e a degradação de IOPS, demonstrando que o gargalo é computacional e não de rede.
3. **Validação de resiliência em topologia restrita:** Comprovou-se o comportamento do Ceph em cenário de falha com fator de replicação mínimo, validando a priorização da consistência sobre a disponibilidade (Teorema CAP).
4. **Guia prático para arquitetos de infraestrutura:** Forneceu-se um laudo técnico baseado em evidências empíricas para subsidiar decisões de projeto entre isolamento lógico e desempenho de E/S.

5.3 Limitações do Estudo

Esta pesquisa apresenta limitações que devem ser consideradas na interpretação dos resultados:

1. **Topologia restrita:** O laboratório operou com apenas dois servidores físicos, o que, embora necessário por questões de viabilidade, não reflete clusters produtivos que tipicamente possuem três ou mais nós para quórum adequado.

2. **Hardware específico:** Os ensaios foram conduzidos em processadores Intel Xeon X5660 (família Westmere-EP), que não possuem as otimizações mais recentes para virtualização (ex: Intel VT-x com EPT aprimorado), podendo subestimar o desempenho do KVM em hardware mais moderno.

3. **Fator de replicação reduzido:** O parâmetro `size=2` utilizado não é recomendado para ambientes de produção críticos, que tipicamente empregam três réplicas para tolerância a falhas.

4. **Generalização limitada:** Não foram avaliados outros hipervisores (Xen, VMware ESXi) nem outros sistemas de SDS (GlusterFS, MinIO), restringindo a generalização dos resultados.

5. **Cargas de trabalho sintéticas:** Embora representativas, as cargas geradas por FIO e sysbench não reproduzem integralmente a complexidade de aplicações reais com padrões de acesso variados.

5.4 Trabalhos Futuros

Esta monografia focou na medição do desempenho arquitetural em topologias físicas restritas. Como a tecnologia de virtualização é vasta, propõem-se as seguintes frentes de pesquisa:

- **Implementação de Quórum Externo:** Avaliar o comportamento dinâmico do agrupamento frente ao acoplamento de um equipamento árbitro neutro para desempate autônomo, eliminando a necessidade de configurações manuais de votação.
- **Aceleração em Acesso Direto (RDMA):** Investigar a configuração do Ceph sobre o protocolo *Remote Direct Memory Access* (RDMA), que permite acesso direto à memória RAM de servidores remotos, reduzindo as transições de contexto analisadas.
- **Otimização via Código de Apagamento:** Avaliar o balanço entre economia de espaço em disco e custo computacional ao substituir a replicação tradicional pelo *Erasure Coding*, analisando o impacto em discos de alta performance.
- **Mecanismos de Descarregamento em Placas de Rede Inteligentes:** Estudar o impacto de placas de rede com processamento dedicado (SmartNICs) para assumir o empacotamento da rede RADOS, reduzindo a carga da CPU e as interrupções nos hipervisores.
- **Eficiência Energética em Infraestruturas HCI:** Realizar medição física de consumo elétrico para correlacionar o tempo de sistema (`sy`) observado no KVM com o aquecimento e o consumo excedente de energia.

Referências Bibliográficas

ALGARNI, A. et al. Performance evaluation of xen, kvm, and proxmox hypervisors. *International Journal of Open Source Software and Processes (IJOSSP)*, IGI Global, v. 9, n. 2, p. 39–54, 2018. Citado 2 vezes nas páginas 18 e 21.

KIVITY, A. et al. kvm: the linux virtual machine monitor. In: *Proceedings of the Linux Symposium*. Ottawa, Ontario, Canada: [s.n.], 2007. v. 1, p. 225–230. Citado 2 vezes nas páginas 13 e 18.

PROXMOX SERVER SOLUTIONS GMBH. *Proxmox VE Ceph Benchmark*. [S.l.], 2023. Citado na página 19.

PROXMOX SERVER SOLUTIONS GMBH. *Proxmox VE 8.4 Datasheet*. [S.l.], 2024. Citado 4 vezes nas páginas 13, 14, 20 e 21.

Proxmox Server Solutions GmbH. *Proxmox VE High Availability (HA)*. [S.l.], 2024. Acesso em: 08 jul. 2026. Disponível em: <https://pve.proxmox.com/wiki/High_Availability>. Citado 3 vezes nas páginas 20, 21 e 22.

ROSEN, R. Linux containers and the future cloud. *Linux Journal*, v. 2014, n. 240, p. 1, 2014. Citado 2 vezes nas páginas 14 e 18.

TANENBAUM, A. S.; STEEN, M. v. *Sistemas Distribuídos: Princípios e Paradigmas*. 2. ed. São Paulo: Pearson, 2007. Citado na página 21.

The Ceph Project. *Ceph Architecture and Block Device (RBD) Documentation*. [S.l.], 2024. Disponível em: <https://docs.ceph.com/>. Acesso em: 29 abr. 2026. Citado na página 20.

WEIL, S. A. et al. Crush: Controlled, scalable, decentralized placement of replicated data. In: *Proceedings of the 2006 ACM/IEEE Conference on Supercomputing*. New York, NY, USA: ACM, 2006. p. 122–133. Citado 2 vezes nas páginas 13 e 19.

WEIL, S. A. et al. Rados: A scalable, reliable storage service for petabyte-scale storage clusters. In: *Proceedings of the 2nd International Workshop on Petascale Data Storage*. New York, NY, USA: ACM, 2007. p. 39–45. Citado na página 19.