Instituto Federal de Educação, Ciência e Tecnologia da Paraíba

Campus João Pessoa

Programa de Pós-Graduação em Tecnologia da Informação

Nível Mestrado Profissional

RODRIGO DE BRITO LIRA

ARQUITETURA PARA ORQUESTRAÇÃO DE AMBIENTES VIRTUALIZADOS EM LABORATÓRIOS DE INFORMÁTICA

DISSERTAÇÃO DE MESTRADO

JOÃO PESSOA

2024

Rodrigo de Brito Lira

Arquitetura para Orquestração de Ambientes Virtualizados em Laboratórios de Informática

Dissertação apresentada como requisito parcial para obtenção do título de Mestre em Tecnologia da Informação, pelo Programa de Pós-Graduação em Tecnologia da Informação do Instituto Federal de Educação, Ciência e Tecnologia da Paraíba – IFPB.

Orientador: Prof. Dr. Paulo Ditarso Maciel Jr.

João Pessoa

2024

Dados Internacionais de Catalogação na Publicação (CIP)
Biblioteca Nilo Peçanha - *Campus* João Pessoa, PB.

L768a Lira, Rodrigo de Brito.

Arquitetura para orquestração de ambientes virtualizados em laboratórios de informática / Rodrigo de Brito Lira. - 2024.
73 f. : il.

Dissertação (Mestrado em Tecnologia da Informação) – Instituto Federal de Educação da Paraíba / Programa de Pós-Graduação em Tecnologia da Informação (PPGTI), 2024.
Orientação : Prof. Dr. Paulo Ditarso Maciel Jr.

1.Gerência de redes. 2. Laboratórios de informática. 3. Virtualização. I. Título.

CDU 004.658(043)

Bibliotecária responsável Lucrecia Camilo de Lima – CRB 15/132

MINISTÉRIO DA EDUCAÇÃO
SECRETARIA DE EDUCAÇÃO PROFISSIONAL E TECNOLÓGICA
INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA DA PARAÍBA

PROGRAMA DE PÓS-GRADUAÇÃO *STRICTO SENSU*

MESTRADO PROFISSIONAL EM TECNOLOGIA DA INFORMAÇÃO

RODRIGO DE BRITO LIRA

ARQUITETURA PARA ORQUESTRAÇÃO DE AMBIENTES VIRTUALIZADOS EM LABORATÓRIOS DE INFORMÁTICA

Dissertação apresentada como requisito para obtenção do título de Mestre em Tecnologia da Informação, pelo Programa de Pós- Graduação em Tecnologia da Informação do Instituto Federal de Educação, Ciência e Tecnologia da Paraíba – IFPB - Campus João Pessoa.

Aprovado em 17 de dezembro de 2024

Membros da Banca Examinadora:

Dr. Paulo Ditarso Maciel Jr.

IFPB - PPGTI

Dr. Thiago Gouveia da Silva

IFPB - PPGTI

Dr. Ramon Leonn Victor Medeiros

Ministério da Cultura

João Pessoa/2024

Dedico este trabalho à minha família, pelo amor incondicional, apoio e compreensão ao longo desta jornada. A vocês, que sempre acreditaram em mim e me incentivaram a superar os desafios, minha eterna gratidão. Às amizades que caminharam ao meu lado durante esta etapa, compartilhando aprendizados, dificuldades e conquistas, meu carinho e reconhecimento. E, acima de tudo, dedico este trabalho a mim mesmo, pela persistência, dedicação e força para continuar mesmo diante das adversidades.

AGRADECIMENTOS

Agradeço primeiramente a Deus, pela força, sabedoria e discernimento concedidos ao longo desta jornada. Foi em Sua presença que encontrei conforto nos momentos de dificuldade, inspiração para seguir em frente e coragem para superar os desafios.

Agradeço, com todo o meu amor, à minha esposa Rafaela Lira, por sua paciência, compreensão e apoio incondicional durante esta jornada. Sua presença foi minha força nos momentos difíceis, seu carinho meu refúgio, e suas palavras de incentivo, meu combustível para seguir em frente. Agradeço também aos meus filhos, que com seu amor e alegria iluminaram minha vida, trazendo motivação e inspiração a cada passo. Vocês são minha maior fonte de felicidade e força, e sem vocês ao meu lado, este sonho também não teria sido possível. Rafaela, este trabalho também é seu, pois sem você e os nossos filhos, eu não teria chegado até aqui.

Expresso minha profunda gratidão ao meu orientador, Prof. Dr. Paulo Ditarso Maciel Jr., por sua orientação valiosa, paciência e dedicação durante toda esta jornada acadêmica. Suas contribuições foram fundamentais para o desenvolvimento deste trabalho, e seu compromisso com a excelência me inspirou a buscar sempre o melhor, pela confiança depositada em mim e por me desafiar a ir além dos meus limites. Este trabalho é reflexo de sua orientação exemplar, pela qual sou eternamente grato.

Aos professores Dr. Thiago Gouveia da Silva e Dr. Ramon Leonn Victor Medeiros, membros da banca avaliadora, expresso minha mais sincera gratidão por dedicarem seu tempo e conhecimento à análise deste trabalho. Suas observações e sugestões foram fundamentais para enriquecer este estudo e torná-lo mais consistente e significativo. Agradeço pela disponibilidade, pelo olhar crítico e pelas contribuições que ajudaram a elevar a qualidade deste trabalho. É uma honra poder contar com a experiência e a dedicação de vocês nesta etapa tão importante da minha trajetória acadêmica.

Aos demais professores que contribuíram de maneira direta ou indireta para minha formação acadêmica, deixo minha sincera gratidão. Cada aula, conselho e ensinamento foram essenciais para minha jornada, ampliando meus horizontes e fortalecendo minha base de conhecimento.

RESUMO

A gestão eficiente de laboratórios de informática representa um desafio constante em instituições públicas de ensino e pesquisa, onde a configuração dos equipamentos é frequentemente realizada de forma manual, consumindo tempo e recursos e estando sujeita a erros que impactam negativamente a usabilidade desses ambientes. Para muitas dessas instituições, a aquisição de soluções proprietárias para o gerenciamento dos laboratórios é financeiramente inviável. Nesse contexto, o uso de tecnologias abertas de virtualização emerge como uma alternativa promissora, permitindo não apenas a redução de custos, mas também o aproveitamento de equipamentos legados, ampliando a vida útil dos recursos disponíveis e promovendo uma gestão mais sustentável. Este trabalho propõe uma arquitetura para orquestração de ambientes virtualizados em laboratórios de informática, estruturada em um modelo de gestão centralizada com execução distribuída. A abordagem permite a realização de aulas virtualizadas utilizando a infraestrutura já disponível, oferecendo flexibilidade e escalabilidade no uso dos recursos. Com a centralização do gerenciamento, a solução facilita a configuração e o monitoramento dos ambientes de forma ágil e padronizada, otimizando a aplicação de atualizações e o controle de recursos. Os resultados preliminares indicam que essa arquitetura é viável e que pode ser implementada de forma gradual, permitindo a adaptação da equipe de gestão dos laboratórios e dos alunos. A solução híbrida se mostra especialmente eficaz em reduzir o custo operacional, aproveitando a infraestrutura existente e integrando novas ferramentas de *software* livre para virtualização e gerenciamento. Além disso, a implementação gradual da arquitetura híbrida promove uma transição suave para a equipe técnica, garantindo que os processos de ensino e aprendizado possam ocorrer com mínima interrupção, enquanto se incorporam novas práticas e tecnologias de virtualização.

Palavras-chaves: Gerência de Redes; Laboratórios de Informática; Virtualização.

ABSTRACT

The efficient management of computer labs represents a constant challenge in public educational and research institutions, where equipment configuration is often performed manually, consuming time and resources and being subject to errors that negatively impact the usability of these environments. For many of these institutions, the acquisition of proprietary solutions for lab management is financially unfeasible. In this context, the use of open virtualization technologies emerges as a promising alternative, allowing not only cost reduction but also the use of legacy equipment, extending the useful life of available resources and promoting more sustainable management. This work proposes an architecture for orchestrating virtualized environments in computer laboratories, structured in a centralized management model with distributed execution. The approach allows virtualized classes to be carried out using the already available infrastructure, offering flexibility and scalability in the use of resources. With centralized management, the solution facilitates the configuration and monitoring of environments in an agile and standardized manner, optimizing the application of updates and resource control. Preliminary results indicate that this architecture is feasible and can be implemented gradually, allowing the lab management team and students to adapt. The hybrid solution has proven particularly effective in reducing operational costs by leveraging existing infrastructure and integrating new open source virtualization and management tools. In addition, the gradual implementation of the hybrid architecture promotes a smooth transition for technical staff, ensuring that teaching and learning processes can occur with minimal disruption while incorporating new virtualization practices and technologies.

Key-words: Network Management; Computer Labs; Virtualization.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

API	Application Programming Interface
CERD	Cloud Educational Resource Datacenter
CIFS	Common Internet File System
CPU	Central Processing Unit
CSBC	Congresso da Sociedade Brasileira de Computação
DaaS	Desktop-as-a-Service
DSL	Domain-Specific Language
FIO	Flexible I/O Tester
FTP	File Transfer Protocol
gRPC	Google Remote Procedure Call
HCL	HashiCorp Configuration Language
HTTP	Hypertext Transfer Protocol
IDV	Intelligent Desktop Virtualization
IaC	Infrastructure-as-Code
IP	Internet Protocol
iSCSI	Internet Small Computer System Interface)
JSON	JavaScript Object Notation
KSM	Kernel same-page Merging
KVM	Kernel-based Virtual Machine
LDAP	Lightweight Directory Access Protocol
MCache	Módulo de Cache
MGer	Módulo de Gerência
MImg	Módulo de Imagens
MInfra	Módulo de Infraestrutura Física

MMon	Módulo de Monitoramento
MIPS	Milhões de Instruções por Segundo
MUser	Módulo de Autenticação e Perfis
MVirt	Módulo de Configuração Virtual
NFS	Network File System
NUMA	Non-Uniform Memory Access
PoC	Proof-of-Concept
RAM	Random Access Memory
RDS	Remote Desktop Service
REST	Representational State Transfer
RFEPCT	Rede Federal de Educação Profissional, Científica e Tecnológica
RPC	Remote Procedure Call
SAML	Security Assertion Markup Language
SEMISH	Seminário Integrado de Software e Hardware
SMB	Server Message Block
SO	Sistema Operacional
SOAP	Simple Object Access Protocol
SSH	Secure Shell
SVN	Subversion
TI	Tecnologia da Informação
TCP	Transmission Control Protocol
TLB	Transaction Lookaside Buffer
URL	Uniform Resource Locator
vCLASS	virtual CLassroom AS a Service
VDI	Virtual Desktop Infraestructure
VirtIO	Virtual IO Device

VM	Virtual Machine
VMM	Virtual Machine Manager
VOI	Virtual OS Infrastructure
XML	Extensible Markup Language
YAML	YAML Ain’t Markup Language

SUMÁRIO

1 INTRODUÇÃO

Na introdução, são apresentados os elementos fundamentais que orientam o desenvolvimento deste trabalho. Primeiramente, serão discutidas as motivações que justificam a realização desta pesquisa, seguidas pela definição clara do problema a ser investigado. Em seguida, são delineados os objetivos que orientam a investigação e as metas a serem alcançadas. Por fim, apresenta-se a estrutura do documento, oferecendo uma visão geral dos conteúdos abordados em cada capítulo e facilitando o entendimento da organização lógica da dissertação.

1.1 Motivação

O uso de tecnologias no cotidiano de trabalho das pessoas acelera o processo de transformação digital que muito se espera, principalmente em ambientes educacionais e de pesquisa. Esta realidade já está presente em diversas instituições de ensino, sobretudo na Rede Federal de Educação Profissional, Científica e Tecnológica[1] (RFEPCT), composta pelos mais de 600 Campi dos Institutos Federais espalhados em todo o território nacional. Com um perfil de ensino profissionalizante dedicado à formação técnica de estudantes para o mercado de trabalho, encontra-se em tais instituições uma numerosa quantidade de laboratórios de informática e pesquisa, que demandam uma gerência cíclica e repetitiva de seus equipamentos.

Além disso, percebe-se que uma utilização mais efetiva de computadores e demais dispositivos poderia alavancar um processo de transformação digital da maneira como os conteúdos são ministrados, como por exemplo em Moreno, Barbosa e Manfio (2019), e compartilhados entre professores, estudantes ou mesmo instituições, como abordado em Sampaio et al. (2015). Contudo, administrar esses recursos de maneira eficaz, extraindo o máximo possível de sua utilização, não é uma atividade trivial e geralmente exige uma capacitação profissional da equipe responsável.

A diversidade de ferramentas voltadas à administração de ambientes interligados através de redes de computadores traz complexidade à formação de um profissional com habilidades nesta área de conhecimento. O uso de soluções de gerência que empregam a virtualização de recursos busca facilitar essa tarefa, porém demanda um alto custo financeiro e, geralmente, necessita de *hardware* específico para o funcionamento. Estas soluções são conhecidas como *Virtual Desktop Infraestructure* (VDI) e baseiam-se em hipervisores de virtualização em um modelo de servidor remoto que hospeda praticamente toda a computação necessária.

Os serviços são usualmente acessados via terminais simples, com quase nenhum poder computacional. Entretanto, essa não é a realidade de muitas instituições de ensino públicas brasileiras; a aquisição de soluções comerciais proprietárias representa um custo elevado, o que

[1] http://portal.mec.gov.br/rede-federal-inicial/

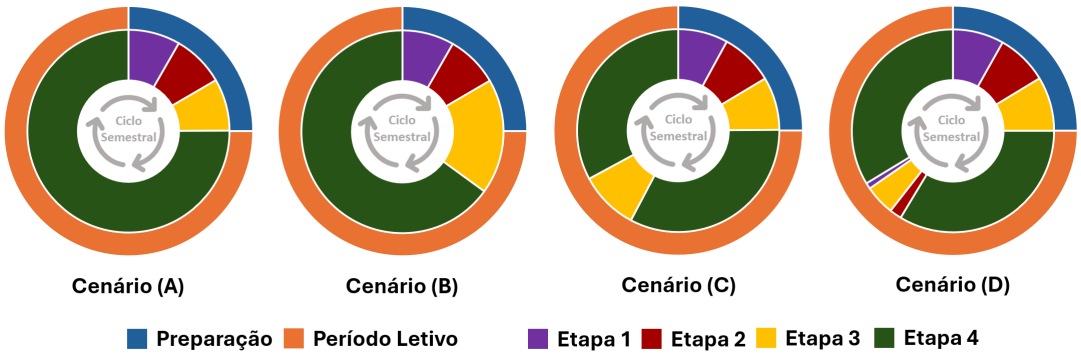
as torna economicamente inviáveis para muitas instituições de ensino. Em especial, aquelas com orçamentos limitados, como universidades públicas e centros de pesquisa. Se por um lado é economicamente inviável comprar soluções prontas que necessitam de *software* e *hardware* extras, por outro lado é indispensável utilizar os recursos legados da maneira mais eficiente possível, até mesmo como uma contrapartida aos impostos pagos pela população.

1.2 Definição do Problema

Em um cenário comum nos laboratórios da RFEPCT, *estações de trabalho* possuem sistemas `Windows` e `Linux` em modo *dual boot*, capazes de executar o sistema hospedeiro e, quando necessário, máquinas virtuais para um ambiente mais "customizável" durante a aula. Antes do início do período letivo, os professores indicam quais aplicativos devem ser instalados em tais estações, que são formatadas e preparadas pelo setor de manutenção de acordo com a alocação dos laboratórios, que leva em conta apenas os requisitos de *hardware*. Estas ações comumente se repetem após o final de cada período letivo, geralmente realizadas de maneira manual e suscetível a erros, o que torna o processo custoso e lento. Finalmente, a realização das práticas requer que os estudantes instalem as máquinas virtuais ou baixem-nas pré-instaladas.

Visando ilustrar a problemática, um conjunto de ações necessário para uma boa condução das aulas em laboratórios de informática pode ser dividido em quatro etapas, como ilustrado na Figura 1. Espera-se que essas etapas sejam realizadas em dois momentos distintos: o de preparação, no qual são tomadas todas as medidas para garantir a efetividade das aulas; e o período letivo, que corresponde à própria execução das aulas práticas nos laboratórios.

Figura 1 – Etapas gerais da realização de aulas virtualizadas em laboratórios de informática.



A Etapa 1, a ser realizada pelo(s) docente(s) encarregado(s) pelo componente curricular (aula), corresponde à preparação das práticas. Nesta fase, devem ser definidos os requisitos para a realização de cada exercício, tais como o sistema operacional, os aplicativos que precisam ser instalados e suas respectivas configurações.

Levando em consideração os requisitos definidos na etapa anterior, a Etapa 2 representa a alocação dos laboratórios às disciplinas, devendo ser realizada pelos responsáveis por gerenciar

o ambiente físico da instituição.

Na Etapa 3, a equipe de suporte é responsável por preparar os laboratórios e computadores para atender às demandas identificadas previamente.

Por fim, a Etapa 4 é conduzida pelo professor e envolve os estudantes na realização das práticas propriamente ditas.

Considerando a Figura 1, o **Cenário A** representa um ciclo semestral onde a realização das etapas foram realizadas nos momentos adequados. Ou seja, as Etapas 1, 2 e 3 ocorreram no período de preparação dos ambientes de laboratório, enquanto a Etapa 4 foi realizada ao longo do semestre letivo. Este exemplo retrata um cenário onde não houve a necessidade de ajustes posteriores no ambiente, que nem sempre acontece devido a diversos problemas de configuração ou de componentes físicos.

O **Cenário B**, por sua vez, exemplifica a finalização tardia de uma etapa de preparação (Etapa 3), ocasionando a indisponibilidade do laboratório para o início do período letivo. Neste caso, o problema pode acontecer caso haja atraso em qualquer uma das três etapas de preparação, não apenas na Etapa 3 como ilustrado. Em relação à Etapa 1, o professor pode querer adicionar, remover ou alterar alguma prática, de acordo com o *feedback* recebido do semestre anterior. Problemas físicos ou de *hardware* também podem exigir a realocação dos laboratórios na Etapa 2, enquanto que a manutenção preventiva, o monitoramento e as alterações realizadas nas Etapas 1 e 2 podem requerer ações adicionais na Etapa 3. O importante é que a indisponibilidade do laboratório para o início do semestre letivo pode prejudicar a disciplina, ainda mais para atrasos relativamente grandes.

Já no **Cenário C**, é possível visualizar que as Etapas 1, 2 e 3 foram realizadas como planejado, ou seja, dentro do momento de preparação. Além disso, observa-se também que a Etapa 4 começa no momento correto em período letivo. Contudo, neste exemplo hipotético, a Etapa 4 é interrompida por algum eventual problema de *software* ou *hardware*. Neste caso, a equipe de suporte necessita intervir no laboratório e executar atividades relacionadas à Etapa 3, o que acaba prejudicando o período letivo com um tempo menor do que o planejado. No geral, a realização de tarefas em um momento posterior ao naturalmente esperado, pode ocasionar interrupções ou mesmo o impedimento de uso do laboratório.

Por fim, no **Cenário D**, observamos que as etapas anteriores ocorreram conforme planejado, cada uma em seu devido tempo. No entanto, neste exemplo hipotético, o processo foi interrompido pela necessidade de troca de laboratório (Etapa 2). Posteriormente, a equipe de suporte precisou preparar esse novo ambiente (Etapa 3), e, por fim, os professores tiveram que adaptar suas aulas ao novo laboratório (Etapa 1).

Ressalta-se que cada etapa possui os seus próprios desafios e definir métricas para avaliar o seu sucesso não é uma tarefa simples. O bom andamento da Etapa 1 pode ser aferido pelo conjunto de ferramentas disponíveis, pela facilidade para preparação dos laboratórios e pela

homogeneidade entre o ambiente de preparação e a sala de aula. Uma boa métrica para a Etapa 2 é o *hardware* necessário para alocação de todas as disciplinas e a flexibilidade para fazer alterações nas alocações. Por sua vez, os tempos de provisionamento e manutenção são bons indicativos de sucesso na Etapa 3. Por fim, o desempenho dos laboratórios virtuais se sobressai como a métrica mais importante da Etapa 4. Adicionalmente, a qualidade da experiência dos usuários pode ser medida através de questionários e entrevistas, oferecendo um indicador que permeia todas as etapas. No entanto, é válido mencionar que tais métricas são exemplos hipotéticos, apresentados para exemplificar possíveis indicadores de desempenho em cada uma das etapas. No contexto deste trabalho, é de particular importância a métrica de desempenho dos laboratórios virtuais na Etapa 4, a qual será explorada mais adiante.

A pesquisa propõe uma arquitetura híbrida para gerenciar laboratórios de informática e pesquisa em instituições de educação técnica e profissionalizante, denominada `vCLASS` (*virtual CLassroom AS a Service*). A arquitetura proposta é composta pela integração de soluções abertas (*open source*), caracterizada pela gerência centralizada dos recursos dos laboratórios e pela execução distribuída das aulas virtualizadas, nos dispositivos finais (estações de trabalho). Desta forma, além de orquestrar o uso dos laboratórios mais efetivamente, será possível utilizar os recursos legados de *hardware* de maneira mais produtiva. Em contrapartida às soluções proprietárias e centralizadas, que geralmente utilizam recursos de *hardware* e *software* próprios. Para isso, é necessário definir quais tecnologias abertas podem ser adaptadas para a `vCLASS` e como integrar de maneira eficiente as diferentes soluções encontradas.

1.3 Objetivos

Objetivo Geral:

Propor uma arquitetura híbrida para gerenciar laboratórios de ensino e pesquisa em instituições de educação técnica e profissionalizante, caracterizada pela gerência centralizada dos recursos e pela execução distribuída das aulas virtualizadas nas estações de trabalho.

Objetivos Específicos:

- Investigar os trabalhos na literatura relacionados com a virtualização de laboratórios de informática, que utilizem ferramentas livres e de código aberto.
- Avaliar quais ferramentas utilizadas se enquadram ao ambiente proposto neste trabalho, pesquisar demais soluções de *software* que possam se enquadrar à arquitetura proposta.
- Especificar uma arquitetura com base nas ferramentas pesquisadas.
- Implementar uma prova de conceito com base na arquitetura proposta, utilizando ambientes virtualizados e avaliar o desempenho da prova de conceito implementada, com o intuito de validar a interação e a eficiência dos componentes.

1.4 Estrutura do Documento

Este documento está organizado em quatro capítulos.

O **Capítulo 1** descreve a motivação da pesquisa, a definição do problema estudado, considerando as etapas realizadas para virtualização das aulas, os objetivos geral e específicos, e a estrutura do documento.

Já o **Capítulo 2** descreve os conceitos básicos sobre as tecnologias que pretendemos utilizar no projeto e quais são as suas ferramentas de código aberto e gratuitos mais populares. além dos trabalhos relacionados encontrados na literatura.

No **Capítulo 3** são apresentados os trabalhos relacionados identificados na literatura, com o objetivo de contextualizar o tema abordado, destacar as contribuições existentes e identificar lacunas que justificam a proposta deste estudo.

O **Capítulo 4** é apresentada a arquitetura `vCLASS`, através do detalhamento de seus módulos e de exemplos do fluxo de comunicação entre cada módulo. Adicionalmente, apresenta-se a metodologia utilizada e os resultados parciais a partir de uma prova de conceito.

Por fim, no **Capítulo 5**, são delineadas as considerações finais e a proposta para continuação desta pesquisa.

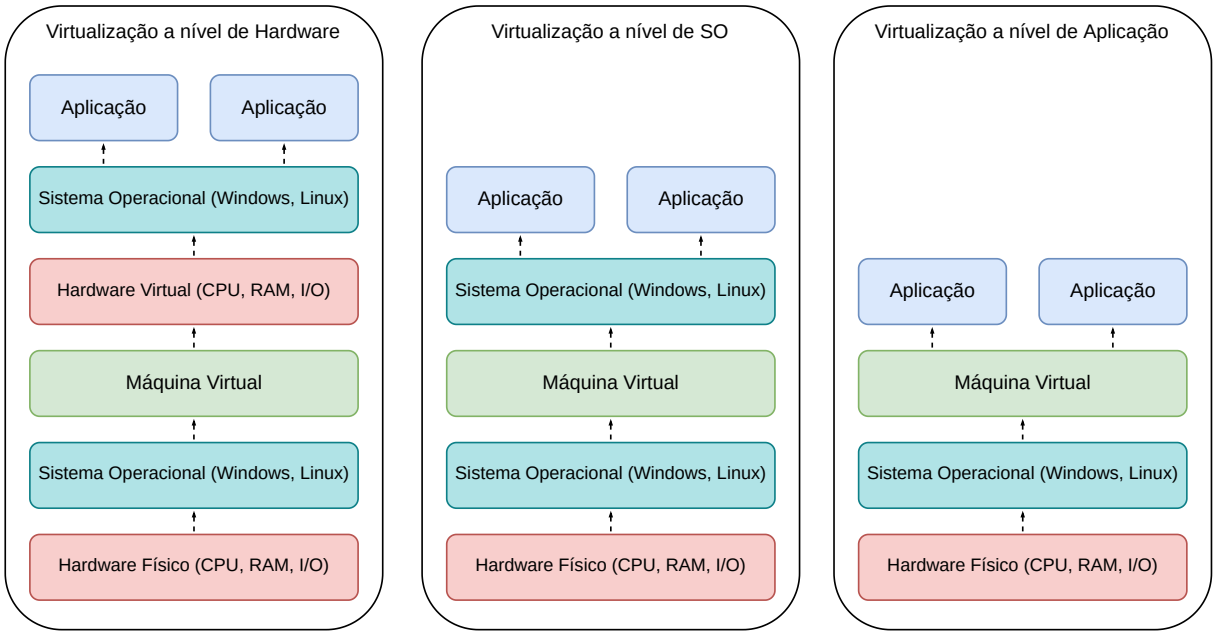
2 FUNDAMENTAÇÃO TEÓRICA

Nesta seção, serão apresentados os conceitos fundamentais que constituem a base teórica deste trabalho, abordando temas cruciais para o desenvolvimento e a compreensão dos objetivos de pesquisa. Serão explorados os seguintes tópicos: virtualização, infraestrutura como código, versionamento de código, protocolos e serviços de rede, e *benchmarking*.

2.1 Virtualização

De acordo com (VERAS, 2016), existem três categorias de virtualização: a primeira em nível de *hardware*, onde é emulado um computador virtual similar ao original; uma segunda no nível do sistema operacional (SO), que permite a partição lógica do ambiente de maneira isolada e compartilhando o mesmo SO; e a terceira em nível de linguagens de programação, onde a camada de virtualização é um programa. A Figura 2 exemplifica essa categorização, através das camadas de virtualização implementadas em cada tipo.

Figura 2 – Categorias de Virtualização.

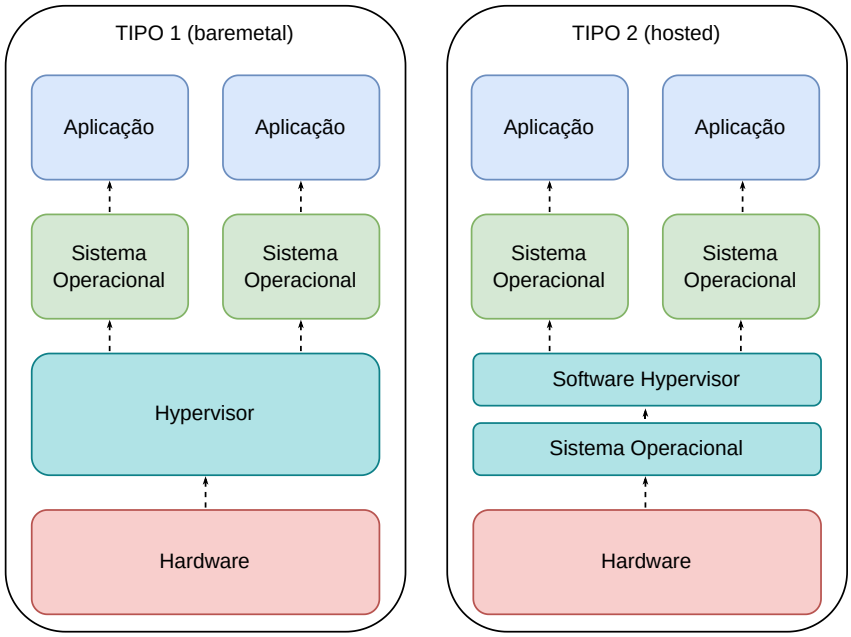


Fonte: adaptado de (BOAVENTURA et al., 2013)

(VERAS, 2016) também diz que os sistemas de virtualização, também conhecidos por *hypervisors* ou *Virtual Machine Managers* (VMM), podem ser classificados em dois tipos, conforme ilustrado na Figura 3. O **Tipo 1** (*bare-metal*) executa o sistema de virtualização diretamente no servidor e possui dois subtipos, o *hypervisor monolítico* e o *hypervisor microkernelizado*. Já no **Tipo 2** (*hosted*), o *hypervisor* é executado através de uma camada de aplicação ou *software* em um sistema operacional convencional. Este último tipo, por se tratar de uma aplicação executada

em um SO e, consequentemente, possuir desempenho limitado, não será considerado para a arquitetura proposta.

Figura 3 – Tipos de Hipervisores.



Fonte: adaptado de (VERAS, 2016)

Além desta classificação tradicional, a virtualização em servidores com arquitetura x86 pode ser classificada também em três tipos, de acordo com a forma como é realizada (VERAS, 2016): virtualização total, paravirtualização ou virtualização assistida por *hardware*.

Virtualização Total: Nesta abordagem, o *hardware* físico é completamente abstraído, de modo que todas as características de uma máquina física são emuladas para a máquina virtual. Dessa forma, o sistema operacional convidado é executado como se estivesse em um ambiente real, sem qualquer modificação. Essa emulação completa permite que o sistema operacional funcione independentemente do tipo de *hardware* subjacente, mas pode gerar uma sobrecarga de desempenho devido à necessidade de emular cada instrução.

Para-virtualização: Nesse modelo, a máquina virtual não precisa ser idêntica ao *hardware* físico original. Em vez disso, o sistema operacional convidado é modificado para ser "consciente" da virtualização, permitindo que algumas instruções mais simples sejam enviadas diretamente ao *hardware*, sem a mediação do *hypervisor*. Isso reduz a sobrecarga de desempenho e aumenta a eficiência, mas requer modificações no sistema operacional convidado, o que pode limitar a compatibilidade.

Virtualização Assistida por Hardware: Nesse tipo de virtualização, o próprio *hardware* (como CPUs modernas da Intel e AMD) oferece suporte específico para virtualização, facilitando a execução de sistemas operacionais virtualizados. Esse suporte minimiza a necessidade de emulação completa e aumenta a eficiência ao permitir que o *hypervisor* gerencie diretamente

as interações entre o sistema operacional convidado e o *hardware*. Como resultado, a virtualização assistida por *hardware* combina a flexibilidade da virtualização total com a eficiência da para-virtualização, permitindo compatibilidade com sistemas operacionais não modificados e melhorando o desempenho.

Também temos outros tipos de virtualização que desempenham papéis fundamentais na computação moderna, como, por exemplo, a virtualização de redes, aplicações, *desktop* ou armazenamento (VERAS, 2011). Cada tipo oferece vantagens específicas que atendem a diferentes necessidades de infraestrutura e otimização de recursos.

Existem diversas soluções de virtualização de servidores que são utilizadas no mercado, cada uma com características e funcionalidades específicas. Entre elas, podemos citar:

VMware vSphere/ESXi: Uma das soluções de virtualização mais populares e robustas, especialmente em ambientes corporativos. O vSphere, com seu *hypervisor* ESXi, oferece alta performance, escalabilidade e recursos avançados, como vMotion (migração ao vivo de VMs), alta disponibilidade e gestão centralizada.

Microsoft Hyper-V: Integrado ao Windows Server, o Hyper-V é amplamente usado em ambientes Windows. É uma solução que oferece boa integração com outras ferramentas da Microsoft, como o System Center, facilitando o gerenciamento e a automação de VMs. Ele também suporta redes virtuais, backup e recuperação simplificados.

Xen/XenServer: Desenvolvido inicialmente como uma solução de código aberto, o Xen oferece virtualização baseada em para-virtualização e, mais recentemente, virtualização assistida por *hardware*. A Citrix disponibiliza o Citrix Hypervisor (anteriormente XenServer), uma versão com suporte comercial, amplamente usada em data centers e ambientes de nuvem.

KVM (*Kernel-based Virtual Machine*): Solução de virtualização de código aberto integrada ao Kernel do Linux, o KVM é altamente flexível e pode ser usado em ambientes de nuvem pública ou privada. É muito popular em plataformas baseadas em Linux e suporta virtualização completa. O KVM permite a execução de VMs com boa eficiência e é a base para muitas soluções de nuvem, como o OpenStack[1].

Oracle VM Server: Desenvolvida pela Oracle, essa solução é baseada no *hypervisor* Xen e é otimizada para executar aplicações da Oracle. Oferece recursos como gerenciamento centralizado e integração com ferramentas de monitoramento, sendo especialmente útil em ambientes que fazem uso intensivo de bancos de dados e soluções empresariais da Oracle.

Proxmox VE: Uma plataforma de virtualização de código aberto que combina KVM e LXC (Linux Containers), permitindo a criação de VMs e containers em um único ambiente. Proxmox é conhecido por sua interface web intuitiva e oferece uma solução de gerenciamento simplificada para clusters de alta disponibilidade, com suporte a backup e recuperação integrados.

Red Hat Virtualization (RHV): Baseado no KVM, o RHV é uma solução de virtu-

[1] https://www.openstack.org/

alização corporativa que fornece gerenciamento avançado para ambientes de VMs, incluindo alta disponibilidade, automação e integração com as ferramentas Red Hat. É muito utilizado em ambientes que já utilizam o ecossistema Red Hat.

VirtualBox: Uma solução adequada para usuários que precisam executar múltiplos sistemas operacionais convidados em diversos sistemas operacionais host, com ou sem suporte de virtualização assistida por hardware, além de ser uma solução gratuita e de código aberto.

VMware Workstation/Player: O VMware Workstation é uma solução paga e avançada de virtualização para *desktops*, com suporte a snapshots, clonagem e redes avançadas, enquanto o VMware Player é uma versão gratuita e básica, ideal para uso pessoal e não comercial, sem recursos como snapshots e clonagem.

Parallels Desktop: Uma solução paga, projetada para executar máquinas virtuais no MacOS, com foco na integração e desempenho, permitindo rodar sistemas operacionais como Windows e Linux de forma otimizada em dispositivos Apple.

Cada uma dessas soluções possui características distintas, desde o suporte a funcionalidades avançadas, como migração ao vivo e alta disponibilidade, até integrações específicas com outros produtos ou sistemas operacionais.

A Tabela 1 apresenta uma comparação entre as soluções de virtualização citadas anteriormente, destacando suas categorias, tipo de virtualização e classificação.

Tabela 1 – Soluções de Virtualização

SOLUÇÃO	**CATEGORIA**			**TIPO**		**CLASSIFICAÇÃO**		
	Nível de Hardware	Nível de SO	Nível de Aplicação	Tipo1	Tipo 2	Virtualização Total	Para-virtualização	Virtualização assistida por Hardware
VMware vSphere/ESXi	X			X		X		X
Microsoft Hyper-V	X			X		X		X
Xen/XenServer	X			X			X	X
KVM	X			X		X		X
Oracle VM Server	X			X			X	X
Proxmox VE	X			X		X		X
Red Hat Virtualization	X			X		X		X
VirtualBox	X				X	X		X
VMware Workstation/Player	X				X	X		X
Parallels Desktop	X				X	X		X

A escolha de uma solução de virtualização depende das necessidades de desempenho, escalabilidade, orçamento e do ecossistema de TI em que será implantada.

Os dois maiores projetos de virtualização Tipo 1 que são gratuitos, de código aberto e se adéquam bem às necessidades da arquitetura `vCLASS`, são: `Xen Project`[2] e `KVM`[3] (*Kernel-based Virtual Machine*).

O `Xen Project` disponibiliza uma tecnologia de virtualização do Tipo 1 e de código aberto, que se destaca pelo suporte a diversos tipos de arquiteturas de computadores, como:

[2] https://xenproject.org/ [3] https://www.linux-kvm.org/

x86, x86-64, IA-32, IA-64 e PowerPC. O projeto foi inicialmente criado na Universidade de *Cambridge* e vem sendo mantido por uma comunidade global de desenvolvedores desde o seu lançamento em 2003. Seu código pode ser executado em diversos tipos de sistemas operacionais, como `FreeBSD` e `Linux`. O projeto é a base de diversas soluções proprietárias de fabricantes, como: `Huawei FusionSphere`[4], `Oracle VM Server`[5], `Citrix Hypervisor`[6] e `XCP-ng`[7].

A solução de virtualização `KVM` é de código aberto e foi integrada ao código-fonte do Kernel Linux a partir da versão 2.6.20. Por estar disponível nativamente no kernel do Linux e não requerer modificações específicas, o KVM serve como base para diversas soluções de virtualização, tanto desenvolvidas por empresas quanto pela própria comunidade de *software* livre, como por exemplo: `Oracle Linux Virtualization Manager`[8], `Nutanix AHV`[9], `Red Hat Virtualization`[10], `Proxmox`[11] e `oVirt`[12].

O mercado dispõe de diversas soluções de virtualização, cada uma com características distintas para atender diferentes demandas. No caso da arquitetura vCLASS, as opções Xen Project e KVM destacam-se por serem gratuitas, de código aberto e compatíveis com as necessidades do projeto, oferecendo flexibilidade e suporte a múltiplas plataformas.

2.1.1 Recursos Avançados

Com o crescimento do uso da virtualização, inúmeros recursos avançados foram disponibilizados, aprimorando significativamente o desempenho das máquinas virtuais. O objetivo é tornar a experiência mais fluida e transparente para o usuário, alcançando um desempenho equiparável ao de uma máquina física. A seguir, são descritas algumas dessas técnicas avançadas.

Hard Partitioning: também chamado de *CPU pinning*, vincula processos a núcleos específicos da CPU, é essencial para melhorar o desempenho do sistema. A afinidade da CPU minimiza as perdas de cache e reduz a sobrecarga de comunicação entre núcleos, aumentando a eficiência dos aplicativos (XIA et al., 2024). Esta técnica é particularmente útil em ambientes cuja carga de trabalho exige uso intenso de processamento e tais recursos dedicados podem garantir um desempenho consistente.

memtune: O objetivo do `memtune`[13] é otimizar o uso da memória RAM para melhorar o desempenho e a eficiência de máquinas virtuais, permitindo ajustar parâmetros como alocação de memória para processos, cache de disco, uso de memória compartilhada e configurações de swap.

Virtual I/O Device (VirtIO): O VirtIO é um conjunto de *drivers* paravirtualizados que permite uma comunicação mais eficiente entre máquinas virtuais e o hipervisor. Ele define uma interface padrão para *drivers* de dispositivos em ambientes virtualizados, possibilitando que máquinas virtuais acessem recursos físicos ou virtuais de forma rápida e eficiente (MATOS

[4] https://www.huawei.com/ [5] https://www.oracle.com/ [6] https://www.citrix.com/ [7] https://xcp-ng.org/
[8] https://www.oracle.com/ [9] https://www.nutanix.com/ [10] https://www.redhat.com/
[11] https://www.proxmox.com/en/ [12] https://www.ovirt.org/ [13] https://libvirt.org

et al., 2023). Ao adotar o VirtIO, os sistemas virtualizados podem obter melhor desempenho e flexibilidade, reduzindo a dependência de *drivers* específicos de *hardware* e facilitando a migração entre plataformas de virtualização.

A virtualização é essencial na computação moderna, proporcionando flexibilidade, eficiência e escalabilidade aos recursos de TI. As diferentes abordagens e soluções apresentadas atendem a diversas necessidades, enquanto técnicas avançadas de virtualização garantem um melhor desempenho. Essa evolução permite gerenciar recursos de forma otimizada, aproximando a experiência virtualizada do uso em máquinas físicas.

2.2 Infraestrutura como Código

Segundo a (REDHAT, 2023), Infraestrutura como código (IaC, do inglês *Infrastructure as Code*) refere-se ao gerenciamento e provisionamento da infraestrutura computacional (servidores, serviços, rede, sistemas operacionais etc.) através do desenvolvimento de códigos, ao invés de processos manuais. É como uma abordagem de gerenciamento de Tecnologia da Informação (TI), em que a infraestrutura é provisionada, gerenciada e mantida usando código e ferramentas de automação. Como definido por (MORRIS, 2016), "a infraestrutura como código é uma prática de gerenciamento de infraestrutura que se concentra em automatizar o provisionamento, a configuração, o gerenciamento e a manutenção de infraestrutura de TI usando práticas de desenvolvimento de *software*".

Ao utilizar IaC, podemos descrever e gerenciar a infraestrutura a partir de código, possibilitando criar e implementar ambientes de maneira rápida e automatizada, economizando tempo e reduzindo a probabilidade de erros humanos, tornando todo o processo mais consistente, escalável e seguro.

Adicionalmente, tal processo é beneficiado com uma maior agilidade na implementação de mudanças de infraestrutura, maior consistência e confiabilidade em ambientes de TI, melhor documentação e rastreamento de mudanças, além de uma maior facilidade de colaboração entre equipes. Isso significa que a configuração da infraestrutura é armazenada em um repositório de código e gerenciada usando ferramentas de controle de versão. A seguir, serão descritas algumas soluções de destaque disponíveis para implementação de IaC.

O `Ansible`[14] é uma solução de automação de TI desenvolvida em Python e com grande aceitação entre a comunidade de *software* livre, que permite configurar, gerenciar e orquestrar a infraestrutura e as aplicações de forma eficiente. O acesso aos dispositivos é geralmente feito por meio de SSH e uma interface web chamada `AWX`[15] pode ser usada para gerenciar os *playbooks*.

O `Terraform`[16] é uma solução da HashiCorp que permite criar, modificar e versionar a infraestrutura de maneira segura e eficiente. Utiliza linguagem de configuração declarativa,

[14] https://www.ansible.com/ [15] https://github.com/ansible/awx [16] https://www.terraform.io/

como p.ex. HCL (*HashiCorp Configuration Language*) ou JSON (*JavaScript Object Notation*). Caracteriza-se como uma ferramenta de código livre, mas não possui uma interface web gratuita.

O `Chef`[17] automatiza o provisionamento de infraestruturas de TI em grande escala, a partir da configuração de servidores e aplicativos, além do gerenciamento do ciclo de vida dos serviços oferecidos. Uma linguagem de domínio específico (DSL, do inglês *Domain Specific Language*) baseada em Ruby é usada para definir e executar *recipies* e *cookbooks*.

O `Puppet`[18] foi projetado para ser executado em um modelo cliente-servidor, onde o `Puppet Master` controla os `Puppet Agents`. O servidor armazena as configurações de todos os recursos gerenciados e envia para os clientes, que as aplicam em seus próprios sistemas. Com ele, é possível gerenciar grandes quantidades de servidores e recursos de computação.

O `SaltStack`[19] também é baseado em um modelo de comunicação entre um servidor (denominado *master*) e os clientes (denominados *minions*). Utiliza um arquivo no formato YAML (*YAML Ain't Markup Language*) ou JSON para definir o estado desejado de um recurso e fornece ferramentas para gerenciar a configuração, monitorar o estado e automatizar tarefas.

No contexto de IaC, é importante diferenciar as abordagens de automação e orquestração que, embora relacionadas, possuem objetivos próprios na gerência de uma infraestrutura de TI. Automação refere-se ao processo de automatizar tarefas manuais e repetitivas que geralmente são executadas por administradores de sistemas, usando ferramentas diversas como *scripts* e aplicativos específicos. Por exemplo, é possível automatizar a implantação de um *software*, atualizar sistemas operacionais, gerenciar configurações e monitorar a toda a infraestrutura de TI. Orquestração, por outro lado, refere-se ao processo de gerenciar e coordenar um grande número de recursos de TI, garantindo um funcionamento de maneira eficiente e coordenada. Isso pode envolver a configuração e gerenciamento de vários servidores, serviços e dispositivos de rede.

2.3 Versionamento de Código

Para o (GITLAB, 2023), o controle de versão é uma importante prática no desenvolvimento de programas para rastrear e gerenciar alterações feitas no código e em outros arquivos, também conhecido como controle de origem ou controle de revisão.

É o processo de gerenciamento de alterações em um código-fonte ao longo do tempo, permitindo que pessoas trabalhem em colaboração de forma eficiente e organizada. Existem diversas ferramentas de versionamento de código, com destaque para as três descritas a seguir.

O `Git`[20] foi criado por Linus Torvalds em 2005. Utiliza uma abordagem distribuída para controle de versão, o que significa que cada desenvolvedor possui uma cópia completa do repositório de código-fonte. Isso permite que os desenvolvedores trabalhem de forma independente, sem precisar se preocupar com conflitos ou problemas de sincronização. O Git é uma ferramenta

[17] https://www.chef.io/ [18] https://www.puppet.com/ [19] https://saltproject.io/ [20] https://git-scm.com/

poderosa e flexível, que pode ser usada em vários sistemas operacionais ou plataformas de hospedagem de código-fonte, como `GitHub`[21], `GitLab`[22] e `Bitbucket`[23].

O `Subversion`[24] (`SVN`) é um sistema de controle de versão centralizado que foi lançado pela *Apache Software Foundation* em 2000. Foi uma das primeiras ferramentas populares para versionamento de código, e ainda é amplamente usada em muitos projetos de *software*. Ao contrário do Git, o SVN usa um modelo centralizado em que um servidor mantém o repositório de código-fonte. Os desenvolvedores fazem *check-out* do código-fonte, realizam alterações em seu próprio ambiente de desenvolvimento e, em seguida, as enviam de volta ao servidor central.

O `Mercurial`[25] é um sistema de controle de versão distribuído, assim como o Git. Foi criado em 2005 por Matt Mackall e é escrito em Python. O Mercurial é usado principalmente para gerenciar código-fonte de *software*, mas também pode ser usado para rastrear e gerenciar qualquer tipo de arquivo. Na abordagem distribuída utilizada pelo Mercurial, cada desenvolvedor tem uma cópia completa do repositório de código-fonte.

Tais ferramentas permitem que os desenvolvedores trabalhem em diferentes ramos (ou *branches*) do código, para que possam implementar novas funcionalidades ou correções de *bugs*, sem interferir no código principal. Além disso, o versionamento de código também ajuda a garantir a integridade e a estabilidade do *software* ao longo do tempo, permitindo que os desenvolvedores gerenciem diferentes versões do código e revertam mudanças indesejadas, se necessário. Com isso, é possível manter um histórico completo de todas as alterações feitas no código, bem como as razões por trás dessas mudanças.

2.4 Protocolos e Serviços de Rede

Este projeto pretende automatizar os serviços de gerência dos usuários e de armazenamento. Para tanto, protocolos de perfis de usuário são usados para definir, armazenar e gerenciar informações de acesso/permissões em uma variedade de aplicações e serviços. Normalmente, estes protocolos definem onde os usuários podem se autenticar e o que podem fazer após essa autenticação.

Existem várias soluções diferentes, incluindo as principais descritas a seguir.

O `OpenID`[26] é um protocolo de autenticação na web que permite ao usuário autenticar-se em sites e aplicativos usando uma única identidade. Em vez de criar contas e senhas separadas para cada site, o usuário pode usar suas credenciais de autenticação de um provedor de identidade confiável para acessar vários sites.

O `LDAP`[27] (*Lightweight Directory Access Protocol*) é um protocolo normalmente empregado para acessar e gerenciar informações de usuários e diretórios, incluindo informações do

[21] https://github.com/ [22] https://gitlab.com/ [23] https://bitbucket.org/ [24] https://subversion.apache.org/
[25] https://www.mercurial-scm.org/ [26] https://openid.net/ [27] https://www.rfc-editor.org/rfc/rfc4511

perfil do usuário. Muitos sistemas de autenticação e gerenciamento de usuários utilizam este protocolo, cuja versão de código aberto mais popular é a OpenLDAP.

O `OAuth`[28] é um protocolo de autorização na web que permite que aplicativos e serviços acessem recursos protegidos em nome de um usuário. Com ele, os usuários podem conceder acesso a seus dados em serviços *on-line*, sem precisar compartilhar suas senhas.

O `SAML`[29] (*Security Assertion Markup Language*) é um protocolo de troca de mensagens XML (*Extensible Markup Language*), que permite a autenticação e autorização de usuários em serviços *on-line*. Representa um padrão aberto e amplamente adotado na indústria de tecnologia.

Outro aspecto importante é como os dados dos usuários serão armazenados. Protocolos de armazenamento em rede são usados para permitir que dispositivos de armazenamento, como servidores de arquivos, armazenamento em nuvem e dispositivos de armazenamento em rede, possam ser acessados por outros dispositivos. Existem diversos protocolos de armazenamento em rede, incluindo: `SMB`[30] (*Server Message Block*), `NFS`[31] (*Network File System*), `iSCSI`[32] (*Internet Small Computer System Interface*) e `FTP`[33] (*File Transfer Protocol*).

Também deve ser levado em consideração o uso de um conjunto de protocolos, rotinas e ferramentas que possibilitem a comunicação entre diferentes *softwares* e sistemas, permitindo que eles compartilhem informações e funcionem de forma integrada. Algumas das formas mais comuns de implementar essa comunicação é através do uso de **API** (*Application Programming Interface*) ou **RPC** (*Remote Procedure Call*). A primeira define um conjunto de regras e protocolos que permitem a comunicação entre diferentes sistemas e aplicativos, caracterizada por tipos distintos como: `REST` (*Representational State Transfer*), baseada no protocolo HTTP; `SOAP`[34](*Simple Object Access Protocol*), baseada na troca de mensagens XML; `GraphQL`[35], especialmente eficiente para consulta a grandes quantidades de dados; e APIs tradicionais de linguagens de programação, que permitem o acesso a recursos como bibliotecas de funções, módulos etc.

RPC é um protocolo de comunicação cliente-servidor que permite a execução remota de procedimentos via rede. O seu funcionamento é bastante simples: o cliente envia uma requisição para o servidor com um procedimento a ser executado e os parâmetros necessários, e o servidor executa o procedimento e retorna o resultado para o cliente. Dentre as possíveis soluções de implementação de RPC consideradas nesta pesquisa, destacam-se as descritas a seguir.

O `gRPC`[36] é um protocolo desenvolvido pelo Google que utiliza o formato de serialização denominado de *protobuf* (*Protocol Buffers*) para a comunicação entre os sistemas. Baseado em HTTP/2.0, permite a comunicação entre diferentes linguagens de programação, além de oferecer suporte a *streaming* bidirecional.

[28] https://www.rfc-editor.org/rfc/rfc6749 [29] https://www.rfc-editor.org/rfc/rfc7522 [30] https://www.samba.org/
[31] https://www.rfc-editor.org/rfc/rfc7530.html [32] https://www.rfc-editor.org/rfc/rfc3720
[33] https://www.rfc-editor.org/rfc/rfc959 [34] https://www.rfc-editor.org/rfc/rfc4227 [35] https://graphql.org/
[36] https://grpc.io/

O `Thrift`[37] é um framework de desenvolvimento de aplicativos distribuídos, originalmente desenvolvido pelo Facebook. Suporta diferentes linguagens de programação e permite a comunicação entre diferentes plataformas. Adicionalmente, oferece suporte a *streaming* bidirecional e pode ser utilizado para desenvolver APIs RESTful.

O `JSON-RPC`[38] é um protocolo de comunicação simples e leve, que utiliza o formato JSON para a serialização dos dados. Permite a execução de procedimentos remotos e a comunicação entre diferentes linguagens de programação. Geralmente, é utilizado em aplicações web que precisam de uma comunicação rápida e eficiente.

Em resumo, este projeto utiliza protocolos e serviços de rede, como OpenID, OAuth, LDAP e SAML, para autenticação e autorização, além de soluções como SMB, NFS, REST e gRPC para armazenamento e comunicação. Essa integração visa automatizar a gestão de usuários e dados, garantindo segurança, eficiência e interoperabilidade no sistema.

2.5 *Benchmarking*

Benchmarking é uma ferramenta metodológica que permite comparar e avaliar práticas, processos ou resultados de pesquisa com padrões de excelência ou referências estabelecidas na literatura. Seu objetivo principal é identificar lacunas, aprimorar metodologias e validar hipóteses com base em dados objetivos e comparativos. Essa prática é amplamente utilizada para fundamentar estudos, garantir a relevância científica e posicionar trabalhos dentro do estado da arte em determinada área de conhecimento.

O *benchmarking* desempenha um papel crucial no desenvolvimento de soluções inovadoras, pois permite que pesquisadores analisem criticamente abordagens existentes, avaliem a eficácia de tecnologias ou metodologias e adaptem melhores práticas às necessidades específicas de seus estudos. Ele também é instrumental na avaliação de desempenho de sistemas computacionais, estratégias educacionais, políticas públicas, entre outros temas relevantes para a produção científica.

Um teste de *benchmarking* consiste na avaliação sistemática do desempenho de sistemas, aplicativos ou componentes de infraestrutura tecnológica, através da comparação com padrões estabelecidos ou com outras soluções semelhantes. Essa técnica é amplamente utilizada para medir a eficiência e identificar possíveis gargalos em *hardware*, *software*, redes e servidores, permitindo otimizar o uso de recursos e melhorar o desempenho de sistemas complexos (HENNESSY; PATTERSON, 2019).

O processo de *benchmarking* geralmente inicia-se com a definição clara dos objetivos do teste, que podem variar de acordo com o tipo de recurso avaliado. Entre os parâmetros mais comuns estão a capacidade de processamento (CPU), o desempenho de entrada e saída de dados (IOPS, do inglês (*I/O per second*), a utilização de memória, a latência de rede, entre outros. Para

[37] https://thrift.apache.org/ [38] https://www.jsonrpc.org/

a execução do teste, são utilizadas ferramentas específicas, permitindo a simulação de diferentes cargas de trabalho e a coleta de métricas essenciais (JAIN, 1991).

Durante os testes, um sistema é submetido a uma carga controlada para medir seu desempenho em diferentes condições operacionais. As métricas coletadas, como tempo de resposta, *throughput* e uso de recursos, são comparadas a *benchmarks*, permitindo uma análise objetiva que identifica oportunidades de melhoria, como ajustes na configuração ou na alocação de recursos. Esses testes de *benchmarking* são aplicados em várias áreas da engenharia de *software* e infraestrutura de TI, incluindo a comparação de componentes de *hardware* e a avaliação do desempenho de sistemas operacionais.

Foi realizada uma pesquisa no Google Scholar para identificar as ferramentas de benchmarks que estão sendo utilizadas na academia, empregando o filtro "benchmark"+ "virtualization"+ "windows"+ "linux"+ "guest", com a análise restrita a publicações entre 2020 e 2024, cujos resultados estão apresentados na Tabela 2.

Tabela 2 – Artigos sobre *Benchmarking*

Artigo	**Ano**	**Benchmarking**	**Tipo**	**SO**
(ĐORđEVIC; MARJANOVIć; TIMčENKO, 2020)	2020	Filebench	Disco	Linux
(LI, 2021)	2021	sysbench	Disco CPU Memória	Linux
		fio	Disco	Linux
		mbw	Disco	Linux Windows
		Fritz Chess Benchmark	CPU	Windows
		Crystal Disk Mark	Disco	Windows
(ĐORđEVIć et al., 2021a)	2021	Filebench	Disco	Linux
(ĐORđEVIć et al., 2021b)	2021	Filebench	Disco	Linux
(ĐORđEVIć; KRALJEVIć, 2023)	2023	Bonnie++	Disco	Linux
		LMbench	Memória	Linux
		Postmark	Disco	Linux
		ATTO	Disco	Windows
		AS SSD	Disco	Windows
(ĐORđEVIć; KRALJEVIć; DžUVEROVIć, 2022)	2022	Filebench	Disco	Linux
(ĐORđEVIĆ et al., 2022)	2022	Filebench	Disco	Linux
(ĐORđEVIć; KRALJEVIć; DAVIDOVIć, 2023)	2023	Filebench	Disco	Linux
(DJORDJEVIć et al., 2023)	2023	HD TUNER PRO	Disco	Windows
		ATTO	Disco	Windows
		Bonnie++	Disco	Linux
		Filebench	Disco	Linux
(ĐORđEVIć et al., 2023)	2023	Filebench	Disco	Linux
(ĐORđEVIć; KRALJEVIć; DAVIDOVIć, 2024)	2024	Filebench	Disco	Linux

Com base nos dados apresentados na Tabela 2, é possível verificar que a maioria dos testes

realizados utiliza sistemas Linux, com foco predominante em testes de desempenho relacionados a discos. No entanto, não foram identificados testes que envolvessem interoperabilidade entre sistemas Linux e Windows.

3 TRABALHOS RELACIONADOS

Este capítulo revisa um conjunto de estudos focados em três áreas complementares: a automatização do provisionamento de laboratórios para fins educacionais, virtualização de *desktops* e *benchmarking* de *hardware*, destacando pesquisas que contribuíram para o avanço desses conceitos e identificando lacunas que o presente trabalho busca abordar. Esses tópicos fornecem a fundamentação necessária para entender as soluções e metodologias propostas ao longo desta pesquisa.

3.1 Automação de Laboratórios

A automatização de ambientes de laboratório educacional utiliza tecnologias como virtualização e conteinerização, além de plataformas de orquestração, buscando oferecer soluções escaláveis e acessíveis que facilitem o ensino. Pesquisas nesse campo exploram tais ferramentas e demonstram suas vantagens na criação e gerenciamento de ambientes virtuais eficientes e adaptáveis a diferentes necessidades curriculares.

Para referências mais antigas, (KRASSMANN et al., 2014) apresentam um mapeamento sistemático com foco na identificação de abordagens e metodologias de avaliação deste tipo de implementação. Foram encontradas 1.165 publicações, catorze sendo selecionadas para extração de dados. Os autores concluíram que a área ainda é pouco explorada.

(RANGAVITTALA; SANJAY; SALVI, 2015) apresentam um *framework multi-tenant* para provisionamento de serviços na nuvem com foco em ambientes educacionais construído em código aberto. Segundo os autores, a arquitetura *multi-tenant* é a característica chave do *framework*, que foi capaz de lidar com todas as demandas educacionais levantadas e suportar vinte usuários simultaneamente.

Por sua vez, (RUBIO; CIVERA; HERRAIZ, 2016) avaliam a aplicabilidade de ferramentas de gerenciamento de configuração para automatizar a criação de exercícios práticos no campo da cibersegurança. Utilizando as ferramentas `Puppet` e `Packer`[1], foram desenvolvidos modelos que geram cenários de teste com serviços vulneráveis e medidas de segurança implementadas. Foram avaliadas a flexibilidade da solução e a economia de tempo alcançada. O estudo conclui que o uso de ferramentas DevOps é uma opção viável tanto para o provisionamento de desafios de ensino em pequena escala, quanto para eventos de cibersegurança em grande escala.

(SHUKHMAN et al., 2017) descrevem o CERD (*Cloud Educational Resource Datacenter*), um centro de dados de recursos educacionais em nuvem baseado no modelo DaaS (*Desktop-as-a-Service*), como uma maneira de fornecer, para instituições educacionais, acesso

[1] https://packer.io/

remoto economicamente viável a *softwares* pagos. O trabalho propõe que a implementação real do CERD seja baseada na plataforma `OpenNebula`[2].

Um *framework* para gerência eficiente e em larga escala de laboratórios universitários, com o mínimo de administradores locais, é introduzido por (MASEK et al., 2018). O *framework* foi proposto com forte uso do `Ansible`, uma ferramenta que simplifica tarefas complexas de orquestração e gerenciamento de configuração. Para tornar as funcionalidades mais acessíveis, foi criada uma aplicação *web*. O *framework* foi testado extensivamente na Universidade de Tecnologia de Brno, na República Tcheca, em laboratórios universitários selecionados.

3.2 Virtualização de *Desktops*

(HUANG, 2019) discute como as tecnologias de virtualização podem fornecer exercícios práticos, reduzir amplamente a necessidade de recursos físicos, melhorar a eficácia do ensino e fornecer oportunidades únicas de avaliação que não estariam disponíveis com o uso apenas de computadores físicos.

Na mesma linha, (ROT; CHROBAK; SOBINSKA, 2019) descrevem o potencial da virtualização e suas aplicações em instituições de ensino superior, destacando a otimização da infraestrutura de TI e a redução dos custos relacionados de operação e manutenção. É apresentado um estudo de caso na Universidade de Economia de Wroclaw, na qual foi implementada uma solução virtualizada com mais de 400 terminais em VDI. Foram utilizados seis servidores com dois processadores cada, totalizando 1,7 TB de memória RAM.

Recentemente, (GENKOV; SLAVOV, 2021) apresentaram uma abordagem de VDI utilizada para exercícios de laboratório em cursos de Computação e Engenharia de Software na Universidade Técnica de Gabrovo. O sistema foi construído e implementado durante a pandemia para oferecer ensino a distância, possibilitando a utilização de vários sistemas operacionais simultaneamente, recuperação rápida de danos ao sistema e *backup* simplificado.

Em outro trabalho, (XIONG et al., 2021) analisam e comparam quatro das principais tecnologias de *desktop* na nuvem da atualidade: RDS (*Remote Desktop Service*), VDI, IDV (*Intelligent Desktop Virtualization*) e VOI (*Virtual OS Infrastructure*). Os autores propõem um esquema híbrido integrando VOI e VDI, provendo uma referência para a implementação em universidades. Por fim, na dissertação de (MARTINS, 2022), uma implementação de sistemas de TI em instituições de ensino superior é abordada, destacando as limitações dos paradigmas tradicionais e os benefícios do uso de Computação em Nuvem e IaC para simplificar a configuração dos sistemas, automatizar processos e reduzir custos. Uma arquitetura de nuvem privada usando `Openstack` e `Ansible` é proposta.

[2] https://opennebula.io/

3.3 Benchmarking

Paralelamente, os estudos sobre *benchmarking* de *hardware* se concentram em avaliar a eficácia do uso de recursos em sistemas heterogêneos, com comparações de desempenho entre sistemas operacionais, como Windows e Linux, em cenários de alta demanda computacional. Esses estudos são essenciais para analisar a performance dos componentes de *hardware*, principalmente em ambientes de virtualização e computação em nuvem, onde a alocação otimizada de recursos é fundamental para assegurar uma experiência de uso consistente em contextos multiusuário. Enquanto soluções proprietárias, como VMware ESXi e Microsoft Hyper-V, oferecem uma integração avançada com ferramentas corporativas, as alternativas gratuitas, como Xen e KVM, são frequentemente preferidas por organizações que buscam reduzir custos e manter maior flexibilidade em seus ambientes de virtualização.

O artigo de (GONZáLEZ et al., 2022) compara o desempenho de containers Docker e LXD em sistemas Linux e Windows, analisando aspectos como uso de CPU, armazenamento e rede. Os testes de desempenho foram conduzidos com *benchmarks* como Geekbench para medir a eficiência do processador, iPerf3 para rede e CrystalDiskMark para armazenamento, buscando avaliar o impacto de cada tecnologia em diversas configurações de *hardware*. A análise comparativa de *benchmarks* de CPU entre Windows e Linux mostrou que o Linux tem desempenho superior, com ganhos de 5% em single-core e 6% em multi-core em relação ao Windows. Em tarefas altamente escaláveis, o desempenho do Linux aumentou para 8% em single-core e 9,5% em multi-core. Cargas de trabalho intensivas, como compilação de código e renderização de imagens (Ray Tracer e Clang), foram especialmente beneficiadas, destacando a eficiência do Linux em gerenciar recursos.

Já o artigo (BIS; BARAN; KULAWSKA, 2023), utilizou a ferramenta de *benchmark* OSBench, que tem como objetivo do teste “Criar Arquivos” e avaliar o desempenho do sistema operacional na criação destes. Especificamente, são criados 65.534 arquivos, cada um contendo 32 bytes de dados, que em seguida são excluídos. Para medir o desempenho do sistema de arquivos, abstraindo o meio de armazenamento, o teste utiliza um sistema de arquivos virtual na RAM. Um teste comparativo foi realizado entre o Windows 11 22H2 e o Linux Ubuntu 23.4 Lunar Lobster, com resultados médios que mostraram um desempenho significativamente superior do Linux. O Windows registrou um tempo médio de 1.204,28 microssegundos, enquanto o Linux obteve um tempo médio de apenas 16,03 microssegundos, evidenciando que o Linux é mais de 75 vezes mais rápido.

O artigo (ABDULRAHEEM, 2024) examina o desempenho das plataformas de virtualização Xen e Hyper-V em cenários de computação em nuvem, focando em como cada uma lida com criptossistemas. O estudo avalia diferentes métricas de desempenho, incluindo tempo de processamento e uso de recursos quando tarefas de criptografia e decriptografia são acionadas, comparando a eficiência de ambos em diferentes cenários, como o uso de algoritmos de criptografia simétrica e assimétrica, destacando que o desempenho varia dependendo da complexidade

dos processos criptográficos. Os resultados mostram que o Hyper-V geralmente apresenta maior velocidade em operações de criptografia pesada devido à sua integração com o *hardware* em ambientes Windows, enquanto o Xen é mais eficiente em operações com Linux.

Já no artigo de (KARIM et al., 2024), os autores analisaram o desempenho das plataformas de virtualização Hyper-V e KVM em tarefas criptográficas no contexto de computação em nuvem. A comparação de desempenho entre Hyper-V e KVM mostrou que a melhor solução de virtualização depende do tipo específico de tarefa criptográfica. Em geral, o KVM apresentou um desempenho ligeiramente superior em operações que exigem processamento intensivo, já o Hyper-V mostrou vantagens em cenários onde a latência é crucial, beneficiando-se da integração com ambientes Windows. Dessa forma, o KVM foi apontado como mais eficiente para tarefas com altas demandas de processamento, enquanto o Hyper-V pode ser preferível em ambientes que já utilizam a infraestrutura Microsoft e priorizam compatibilidade e latência.

Nós podemos ver no artigo (DJORDJEVIC et al., 2021) uma comparação de desempenho dos sistemas de arquivos de diversos hipervisores populares: ESXi, KVM, Hyper-V e Xen. Os autores medem o desempenho em operações de entrada e saída (I/O) em diferentes sistemas de arquivos, analisando métricas como taxa de transferência, latência e IOPS sob diferentes cargas de trabalho. Os resultados indicam que cada hipervisor apresenta vantagens específicas em tipos distintos de operações de I/O. O ESXi se destaca pela consistência em operações intensivas, o KVM se mostra eficiente para cargas pesadas com alta demanda de *throughput*, o Hyper-V tem melhor desempenho em ambientes Windows e o Xen, apesar de estável, fica atrás dos demais em alguns cenários de alta concorrência.

Neste outro artigo (FIASE et al., 2024), os autores utilizaram os mesmos hipervisores do estudo anterior. No entanto, em vez de focarem exclusivamente no sistema de arquivos, ampliaram a análise para incluir métricas de desempenho em CPU, memória, taxa de transferência e IOPS. Os resultados mostram que cada hipervisor possui vantagens em cargas de trabalho específicas: o ESXi tem melhor desempenho em *workloads* com I/O intensivo, o KVM se destaca em cargas pesadas que exigem alto *throughput*, o Hyper-V é eficiente em ambientes Windows, e o Xen, embora estável, pode apresentar desempenho inferior em algumas situações de alta demanda.

O impacto da prática de CPU *pinning* (ou afinidade de CPU) em ambientes virtualizados e de contêineres pode ser encontrado no artigo de (SAMANI et al., 2020) . O estudo compara o desempenho de plataformas de virtualização e conteinerização, para diferentes tipos de cargas de trabalho. Os resultados mostram que a pinagem de CPU pode melhorar o desempenho e a previsibilidade em cenários com alta demanda de CPU, mas também possui desvantagens, como o aumento do consumo de energia e o uso de apenas núcleos fixos, o que pode resultar em desperdício de recursos quando esses núcleos não estão em uso total. O artigo conclui que a afinidade de CPU deve ser utilizada de forma estratégica para maximizar o desempenho, dependendo das necessidades específicas de carga de trabalho e do ambiente de execução.

No artigo (RUSSELL; TSIRKIN; HUCK, 2014) detalham como o Virtio otimiza o desempenho e a eficiência dos sistemas virtuais ao permitir que dispositivos de I/O (como discos e interfaces de rede) sejam abstraídos em um formato que pode ser utilizado por diferentes hipervisores e sistemas operacionais, sem precisar de *drivers* específicos. Isso proporciona uma maior interoperabilidade e portabilidade, consolidando o Virtio como um padrão de fato em sistemas virtualizados. O trabalho ainda discute como o Virtio reduz a sobrecarga de virtualização, melhorando a integração de sistemas em nuvens públicas e privadas.

Neste capítulo, foram apresentados os trabalhos relacionados com esta pesquisa, que servem de base para o desenvolvimento e a fundamentação da solução proposta. Algumas das abordagens discutidas aqui são aplicadas diretamente nesta pesquisa, contribuindo para aspectos como eficiência, segurança e escalabilidade. Outras, embora não implementadas diretamente, são mencionadas para contextualizar o avanço da pesquisa e demonstrar alternativas e limitações observadas em estudos anteriores.

4 DESCRIÇÃO DA PROPOSTA

Nesta seção apresentamos a estrutura da pesquisa desenvolvida, descrevendo a arquitetura proposta e seus fluxos de trabalho, a metodologia adotada para avaliar uma prova de conceito e os resultados obtidos.

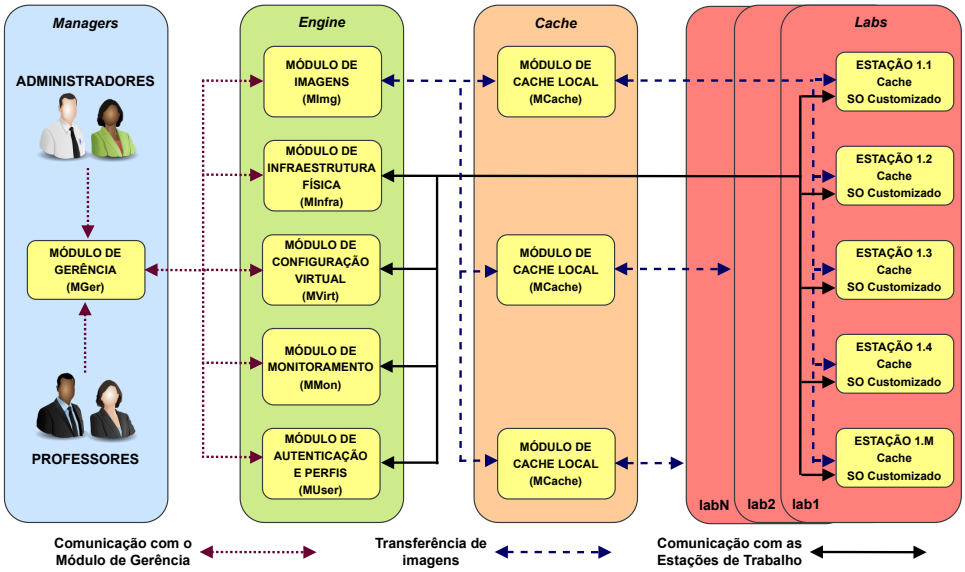
4.1 Arquitetura

Esta subseção está dividida em duas partes. Na primeira, apresentamos os módulos propostos pela arquitetura da solução, detalhando suas funcionalidades e a forma como se integram para alcançar os objetivos do projeto. Na segunda parte, demonstramos possíveis fluxos de trabalho, ilustrando como os módulos interagem entre si para atender aos requisitos funcionais e operacionais da solução.

4.1.1 Módulos

A arquitetura híbrida proposta neste trabalho é composta por diversos módulos que interagem para o provimento dos serviços e que podem ser visualizados a partir de quatro seções: *Managers*, *Engine*, *Cache* e *Labs*.

Figura 4 – Arquitetura `vCLASS`.



Fonte: Próprio Autor (2024)

A Figura 4 ilustra como estes módulos estão dispostos na arquitetura e como é realizada a comunicação entre eles. A seção denominada *Managers* apresenta uma visão geral do sistema aos seus usuários, que podem ser administradores ou professores, através do **Módulo de Gerência**

(`MGer`). A seção *Engine* contempla cinco módulos da arquitetura híbrida, são eles: **Módulo de Imagens** (`MImg`), **Módulo de Infraestrutura Física** (`MInfra`), **Módulo de Configuração Virtual** (`MVirt`), **Módulo de Monitoramento** (`MMon`) e **Módulo de Autenticação e Perfis** (`MUser`). A seção *Cache* é composta por um único componente, que é o **Módulo de *Cache*** (`MCache`), replicado em pontos estratégicos para agilizar a transferência de imagens, quando necessário.

Por fim, a seção *Labs* contém as **estações de trabalho** da arquitetura híbrida, onde serão executadas as aulas virtualizadas. Tais estações estarão habilitadas para opção de inicialização via rede e consistem em um Sistema Operacional (SO) customizado para a virtualização das aulas, com um componente de *cache* previamente configurado.

A interação dos usuários com o módulo `MGer` é realizada por meio de uma interface Web e a comunicação com os demais módulos é implementada com uso de APIs (p.ex. via gRPC ou REST). A interação dos professores com este módulo ocorre estritamente para a elaboração ou atualização de imagens de suas respectivas aulas. Para tanto, pode ser disponibilizado um serviço completo de criação de imagens via interface Web ou mesmo um formulário de requisitos a ser preenchido pelos professores menos familiarizados com ferramentas de virtualização. O objetivo é representar de forma simples suas requisições para as aulas virtualizadas, seja via formulário ou sistema Web.

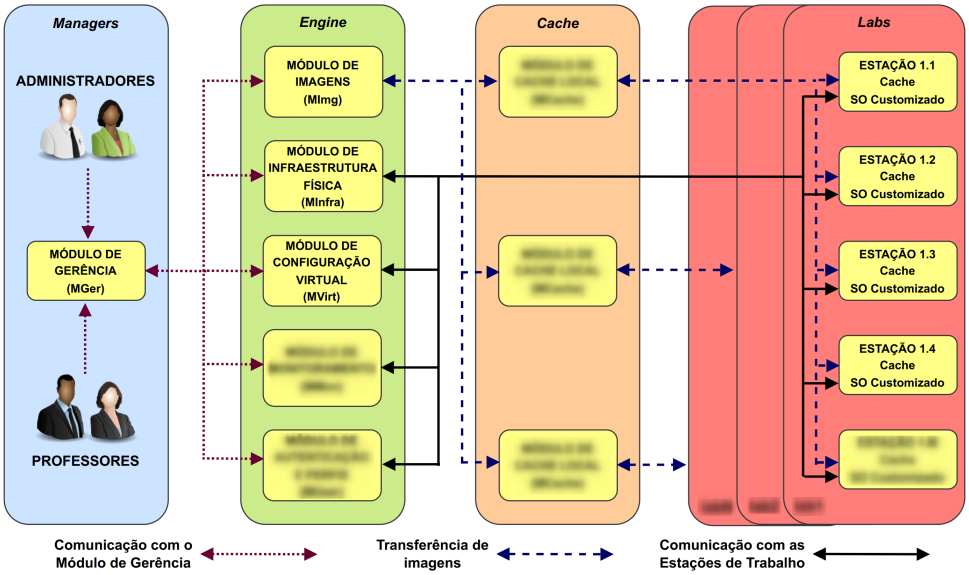
O `MImg` disponibiliza um serviço para armazenar e distribuir as imagens primárias das máquinas virtuais, aquelas que são configuradas ou atualizadas pelos professores e definidas como a versão mais atual de suas aulas. Estas imagens são basicamente uma instanciação de um SO com os requisitos de *software* necessários para a aula. Apesar de não conter dados sensíveis, sugere-se algum mecanismo de criptografia no envio das imagens, pois previne a falsificação das mesmas. O `MInfra` será responsável por realizar a configuração *bare-metal* das estações de trabalho, através da distribuição e atualização do SO customizado para virtualização das aulas. Este último provê o ambiente necessário à execução das aulas virtualizadas, além de pré-configurar um componente de *cache* no dispositivo final. Além disso, deve ter compatibilidade com o `MVirt`, uma vez que este último é responsável por criar a infraestrutura virtual definida pelo `MGer`. Como requisitos funcionais de uma solução para o `MVirt`, espera-se o uso de uma linguagem de fácil entendimento e usabilidade, sem a dependência de agentes de execução e cuja comunicação seja criptografada. O `MMon` coleta informações de utilização das estações de trabalho com o intuito de verificar a "saúde" do *hardware* e exibe as informações através do `MGer`. Ao monitorar os recursos computacionais, como processador, memória, capacidade em disco e tráfego de rede, será possível agregar informações para tentar antecipar problemas que inutilizem as estações de trabalho e, consequentemente, prejudiquem as aulas. O `MUser` é responsável por centralizar as configurações de usuários para acesso às estações e personalização de perfis. Para tanto, deverá interagir com soluções tradicionais de rede como `LDAP`, `CIFS` ou `NFS`. Através deste módulo será possível, dentre outros funcionamentos, mapear unidades de

armazenamento em rede para os arquivos dos usuários ou estabelecer quotas e privilégios de maneira centralizada.

Como mencionado anteriormente, o `MCache` tem o intuito de agilizar a distribuição das imagens armazenando-as mais próximas das estações de trabalho, seja para uma implantação inicial do laboratório ou atualizações do mesmo. Na ponta da arquitetura híbrida, temos as estações de trabalho que executarão as devidas aulas virtualizadas. Um componente local do `MCache` terá a atribuição de verificar se a cópia local das máquinas virtuais é a mais recente, ou se será necessária uma atualização.

Embora a Figura 4 apresente todos os módulos projetados na arquitetura, para simplificar os testes realizados nesta pesquisa, optamos por implementar apenas alguns módulos. Essa escolha visou garantir a execução eficiente e o controle de variáveis, permitindo uma análise mais focada e precisa nos módulos selecionados.

Figura 5 – Módulos implementados da Arquitetura `vCLASS`.



Fonte: Próprio Autor (2024)

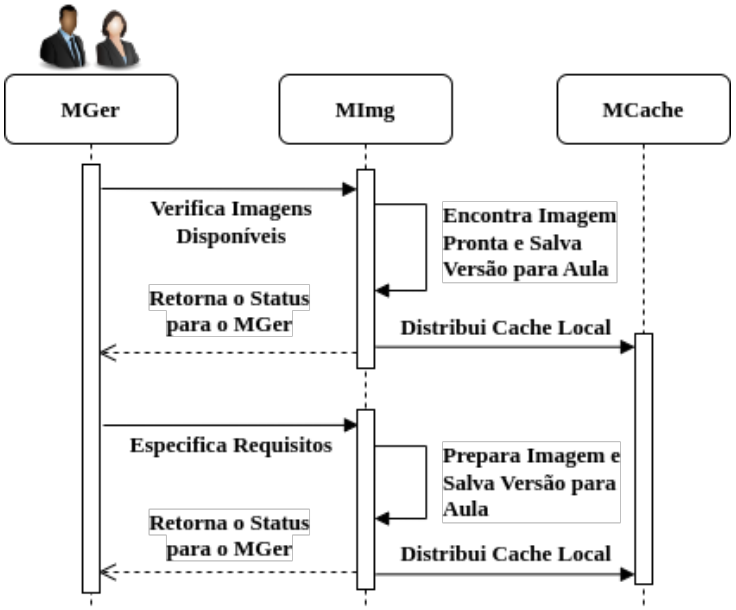
A Figura 5 ilustra os módulos que foram efetivamente implementados e testados durante esta pesquisa. Os módulos restantes, que não foram abordados nesta fase e estão representados de forma borrada na figura, desempenham um papel fundamental na arquitetura e serão incorporados em investigações subsequentes. A inclusão desses módulos em etapas futuras permitirá uma avaliação mais completa do sistema e contribuirá para a validação integral da arquitetura proposta.

A inclusão desses módulos em futuras pesquisas é essencial para validar a arquitetura como um todo, além de possibilitar uma avaliação mais completa e robusta do sistema em cenários variados. O avanço para as próximas fases de implementação permitirá a integração gradual desses módulos, contribuindo para o aprimoramento contínuo do modelo proposto.

4.1.2 Fluxos de Trabalho

Como intuito de esclarecer as interações e execuções de um sistema a partir da arquitetura híbrida proposta, alguns fluxos de trabalho são descritos nesta subseção.

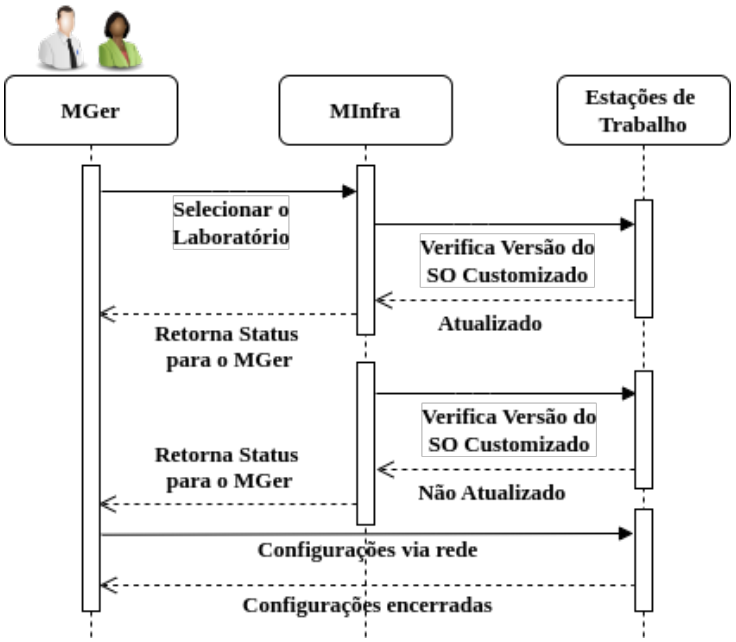
Figura 6 – Criação de imagens.



Fonte: Próprio Autor (2024)

A Figura 6 apresenta o fluxo de trabalho para a criação da imagem da aula virtualizada. Este fluxo está relacionado com a Etapa 1 descrita na Figura 1, na qual um professor é responsável por preparar a aula que será ministrada de maneira virtualizada. Ao interagir com o `MGer`, os professores podem verificar a existência de uma imagem adequada já disponibilizada, sem a necessidade de instalações personalizadas. Por outro lado, é possível especificar os requisitos de *software* necessários à aula, para que o `MImg` possa preparar a imagem desejada e salvar sua versão para distribuição. Em ambos os casos, uma cópia da imagem pode ser distribuída para um ou vários módulos `MCache`.

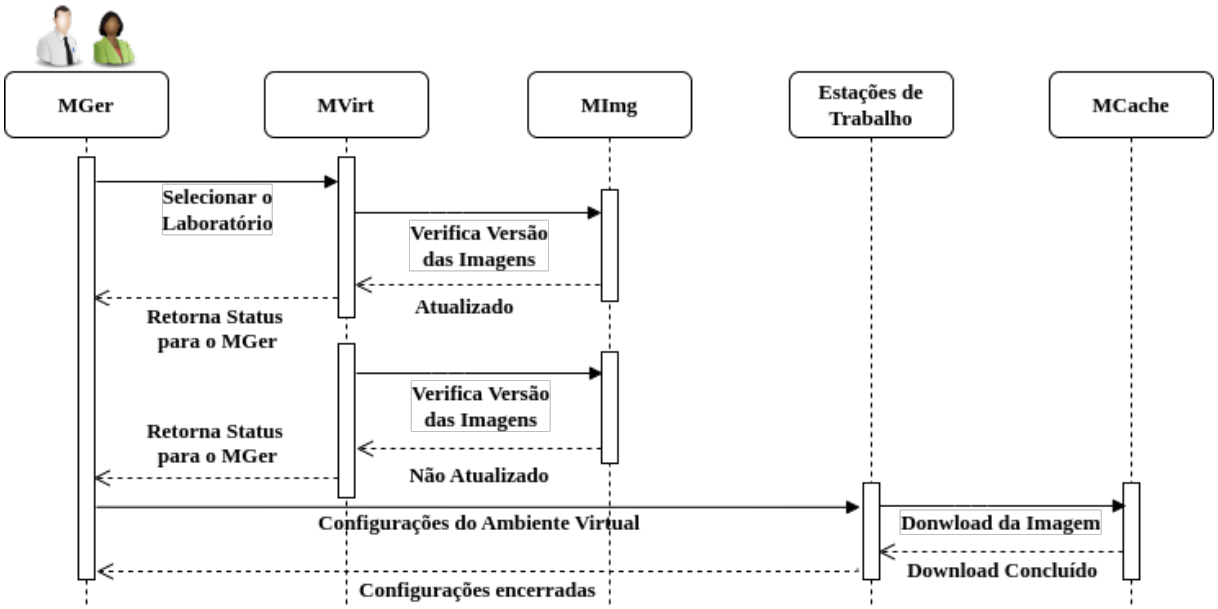
Figura 7 – Configuração *bare-metal*.



Fonte: Próprio Autor (2024)

A Figura 7 apresenta um fluxo de trabalho relacionado à configuração da infraestrutura física, através da instalação dos SOs customizados via `MInfra`. Tal atividade é parte da Etapa 2 descrita na Figura 1, onde os administradores necessitam preparar o ambiente físico para execução das aulas e, quando necessário (p.ex. atualização do SO customizado), reinstalar o SO nos equipamentos. É importante observar que a `vCLASS` visa mitigar a necessidade de reinstalações físicas, uma vez que as aulas serão realizadas de forma virtual. No fluxo em questão, Administradores utilizam o `MGer` para iniciar o processo e selecionar em qual laboratório será realizada a configuração *bare-metal*. Por sua vez, o `MInfra` é acionado para verificar se o SO customizado presente nas estações do referido laboratório está em sua versão mais atual ou não. Caso exista nova versão, o `MGer` comanda a configuração *bare-metal* do laboratório que consiste em formatar e instalar a nova versão do SO.

Figura 8 – Fluxo de configuração do ambiente virtual.



Fonte: Próprio Autor (2024)

A Figura 8 demonstra o fluxo para instalação das aulas virtualizadas nas estações de trabalho. Caso exista, o ambiente será inicializado a partir de uma VM previamente configurada. Caso contrário, um conjunto de instruções será executado para realizar as configurações necessárias ao funcionamento do ambiente de aula. As fases deste fluxo consistem em: selecionar o laboratório específico por parte dos administradores; verificar se as versões das imagens estão atualizadas; e iniciar a configuração do ambiente virtual, onde pode ser preciso fazer o *download* de uma versão mais atual da aula virtualizada. As ações realizadas neste fluxo estão sitiadas na Etapa 3 definida na Figura 1.

A representação dos fluxos descritos acima é uma versão resumida de todas as atividades necessárias para tais configurações, onde detalhes dos processos e das interações com os demais módulos foram omitidos para efeito de simplificação e escopo desta pesquisa.

4.2 Implementação de Referência

Nesta subseção é descrito o ambiente de *hardware* e *software* utilizados na implementação de referência, nós realizamos uma prova de conceito (PoC - *Proof-of-Concept*) da arquitetura `vCLASS`. Mais especificamente, alguns módulos da arquitetura foram instanciados com ferramentas consolidadas em ambientes de IaC, com os quais foi possível validar o processo de configuração do ambiente virtual, bem como analisar o desempenho da solução customizada de virtualização.

4.2.1 Hardware

O *hardware* utilizado para a prova de conceito foi um notebook da marca Lenovo modelo ThinkPad E480, equipado com as seguintes especificações técnicas:

- Processador: Intel® Core™ i5-8250U 1.60 GHz com 4 núcleos e 8 threads
- Placa de Vídeo: Integrada Intel Graphixs 620
- Memória RAM: 32GB DDR4 2400MHz
- Disco: SSD HP EX950 1TB M2 NVME
- Placa de Rede: Realtek® RTL8111GUS

4.2.2 Software

Em um primeiro momento, buscou-se reproduzir um ambiente de configuração das aulas virtualizadas nas estações de trabalho. O módulo de armazenamento e transferência de imagens `MImg` foi instanciado como um servidor Web simples, com o uso do `Apache HTTP Server`[1]. O uso de HTTP para a transferência de imagens também representa um teste de validade de uma futura implementação baseada em API REST. Já o módulo `MVirt`, responsável por configurar o ambiente virtual, foi instanciado a partir da ferramenta `Ansible`. A solução `AWX`[2] foi utilizada para representar as ações de configuração do `MGer` a partir de uma interface Web. Por fim, o SO customizado para as estações de trabalho consiste de um `Linux Debian` versão 11[3], com uma seleção reduzida de pacotes, uma interface gráfica leve[4] e personalizada[5], além do hipervisor `KVM`[6] (*Kernel-based Virtual Machine*).

Ao iniciar este projeto, a escolha de uma plataforma de virtualização adequada foi fundamental para garantir um ambiente estável, eficiente e alinhado às necessidades da pesquisa. Após uma análise criteriosa das opções disponíveis, optamos por utilizar o KVM como base para a implementação. Essa escolha se deve principalmente à sua integração nativa com o kernel Linux, o que proporciona um desempenho robusto e simplifica a configuração e manutenção do ambiente. Além disso, o KVM é amplamente utilizado e conta com uma vasta comunidade de suporte, fator relevante para a resolução de eventuais desafios ao longo do projeto. Outro ponto determinante foi a sua flexibilidade em suportar diversas ferramentas de gerenciamento de infraestrutura como código.

Assim, o KVM desponta como uma escolha estratégica e eficaz para o desenvolvimento deste estudo, alinhando alto desempenho, facilidade de uso e relevância prática na área de virtualização e infraestrutura de TI. Devido à limitação de suporte às tecnologias avançadas de virtualização pelo *hardware*, optamos por restringir os testes a apenas três tecnologias.

[1] https://httpd.apache.org/ [2] https://github.com/ansible/awx [3] https://debian.org/ [4] https://xlunch.org/ [5] http://openbox.org/ [6] https://linux-kvm.org/

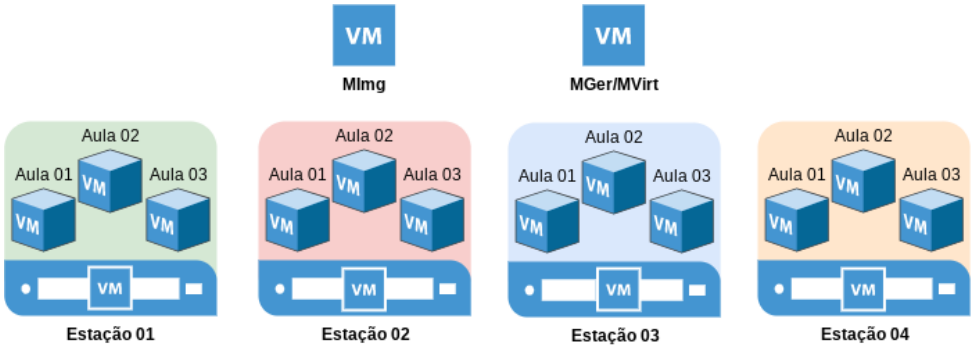
Essa escolha visa maximizar o aproveitamento dos recursos oferecidos e permitir uma análise aprofundada do desempenho das máquinas virtuais, mesmo com as limitações de infraestrutura. Dessa forma, podemos investigar as técnicas avançadas selecionadas, mantendo o foco na transparência e fluidez da experiência do usuário, com o objetivo de alcançar um desempenho próximo ao de uma máquina física.

A implementação de referência foi configurada a partir de dois experimentos.

O primeiro busca demonstrar a viabilidade da arquitetura proposta em configurar o ambiente de aulas virtuais em um tempo satisfatório.

Para efeito de simplificação, o ambiente utilizado neste experimento foi todo virtualizado localmente no hardware mencionado na subseção anterior, onde utilizamos o recurso de virtualização alinhada, também conhecido como **`nested virtualization`**[7], para podermos executar a virtualização dentro de máquinas virtuais. Foram utilizadas ao todo dezoito máquinas virtuais, quatro para as estações de trabalho, três por estação para representar as aulas virtualizadas, uma para o **`MImg`** e outra para o **`MGer/MVirt`**.

Figura 9 – Arquitetura Virtual.



Fonte: Próprio Autor (2024)

A Figura 9 representa a arquitetura do ambiente virtualizado utilizado para o primeiro experimento. Foram utilizadas as soluções **`Apache HTTP Server`**, **`AWX`** e **`Ansible`** para o **`MImg`**, o **`MGer`** e o **`MVirt`** respectivamente. Nas estações foi instalada uma versão customizada do SO **`Linux Debian`**.

O segundo experimento visa apresentar uma análise de desempenho dos ambientes, com o objetivo de comparar o ambiente físico com a versão não customizada, com a versão customizada e com o ambiente ajustado, detalhados a seguir. O intuito é demonstrar o quanto é possível melhorar em termos de desempenho em configurações futuras.

[7] https://www.linux-kvm.org/page/Nested_Guests

4.3 Resultados

Como mencionado anteriormente, o primeiro experimento visa comprovar a viabilidade das transferências de aulas virtualizadas no processo de preparação do laboratório para um semestre letivo. Ou seja, parte do processo de configuração do ambiente virtual representado pelo fluxo da Figura 8 e relacionado à Etapa 3 descrita na Figura 1.

4.3.1 Tempo de Implementação

O experimento consiste em estimar o tempo necessário para transferir aulas virtualizadas, previamente preparadas e armazenadas no `MImg`, para as estações de trabalho. Por ser um experimento simplificado e virtualizado, optou-se por não instanciar o `MCache` e transferir as imagens diretamente do `MImg`. Para representar as aulas virtualizadas, foram utilizadas as versões "*vanilla*" do `Debian 11 (Gnome)`, `Ubuntu Desktop 22.04` e `Windows 10`. A Tabela 3 apresenta o sistema operacional, o tamanho de cada disco e a denominação da aula.

Tabela 3 – Sistema operacional e tamanho das imagens das aulas.

Descrição	Sistema Operacional	Tamanho
Aula 01	`Debian 11 (Gnome)`	4,9G
Aula 02	`Ubuntu Desktop 22.04`	9,3G
Aula 03	`Windows 10`	11,0G

A Tabela 4 apresenta o resultado do tempo de transferência e inicialização das aulas virtuais para cada estação, levando em conta que a execução da implementação é feita de forma simultânea nas quatro estações. Simultaneamente, foi possível configurar as aulas nas quatro estações de trabalho disponíveis em menos de 10 minutos. Como indicado, este tempo contabiliza não apenas a transferência entre o `MImg` e as estações de trabalho, mas também a inicialização de cada VM. Dessa forma, as aulas estarão prontas para uso, sem a necessidade da primeira (e geralmente mais demorada) execução ser realizada pelos estudantes.

Tabela 4 – Tempo de transferência e inicialização das aulas virtualizadas.

Estação 01		**Estação 02**		**Estação 03**		**Estação 04**	
VM	Tempo	VM	Tempo	VM	Tempo	VM	Tempo
Aula 01	01:18	Aula 01	01:23	Aula 01	01:22	Aula 01	01:29
Aula 02	02:53	Aula 02	02:57	Aula 02	02:52	Aula 02	02:55
Aula 03	05:49	Aula 03	05:55	Aula 03	05:47	Aula 03	05:40
Tempo Total	9:20	Tempo Total	9:31	Tempo Total	9:21	Tempo Total	9:24

Os resultados obtidos neste experimento simplificado não objetivam retratar cenários realísticos, uma vez que vários fatores são desconsiderados, p.ex. a influência do tráfego em uma rede real. Contudo, acredita-se que este tempo é um indício satisfatório da viabilidade da arquitetura `vCLASS` e, certamente, representa um tempo muito inferior se comparado à atividade manual de instalar esses três SOs em uma máquina física do laboratório. Conjectura-se, portanto,

que estas mesmas tarefas realizadas manualmente podem levar horas, ou mesmo dias, para finalizar um laboratório. A vantagem oferecida pela arquitetura `vCLASS` é automatizar todo esse processo de forma centralizada e ainda utilizar os recursos distribuídos de forma eficiente. No entanto, uma análise detalhada se faz necessária para demonstrar o uso da arquitetura completa em um ambiente empírico, inclusive utilizando o `MCache` para mitigar os efeitos do tráfego em rede.

4.3.2 Testes de Desempenho

Como uma forma de verificar o desempenho de cada ambiente, foram realizados testes de *benchmark* com a suíte de *software* `OpenBenchmarking`[8]. O principal objetivo é comparar o desempenho em relação a uma versão não customizada do ambiente de virtualização, usando como parâmetro o desempenho dos testes em uma máquina física. O `OpenBenchmarking` fornece uma plataforma aberta de testes, onde foram selecionados com base na popularidade e interoperabilidade (utilização em `Windows` e `Linux`). Para verificar o desempenho de processamento (CPU) foi utilizado o `Blender`[9] (versão 3.2.0), que estima o tempo de renderização de uma imagem a partir de uma modelagem 3D. Adicionalmente, foi utilizado o `t-test1`[10] (versão 1.0.1) para teste de desempenho que mede o tempo de alocação em memória. Por fim, foi selecionado o `Flexible IO Tester`[11] (versão 3.35) para realizar testes de leitura e escrita em disco.

Para um melhor entendimento, os resultados dos ambientes com virtualização (customizado, não customizado e ajustado) estão normalizados (diferença percentual) em relação aos obtidos na máquina física, uma vez que espera-se conseguir um desempenho melhor nesta última. Portanto, considera-se $v_{a,r}$ como o valor obtido no ambiente $a = \{f, c, nc, aj\}$ (físico, customizado, não customizado e ajustado) para o teste de *benchmark* do recurso $r = \{p, m, r, w\}$ (processamento, memória, leitura em disco e escrita em disco). As diferenças percentuais dos ambientes customizado e não customizado para o ambiente físico, considerando todos os recursos avaliados, são:

Processamento

$$d_{c,p} = \frac{v_{c,p} - v_{f,p}}{v_{f,p}} \cdot 100 \quad \text{e} \quad d_{nc,p} = \frac{v_{nc,p} - v_{f,p}}{v_{f,p}} \cdot 100 \quad \text{e} \quad d_{aj,p} = \frac{v_{aj,p} - v_{f,p}}{v_{f,p}} \cdot 100 \qquad (1)$$

Memória

$$d_{c,m} = \frac{v_{c,m} - v_{f,m}}{v_{f,m}} \cdot 100 \quad \text{e} \quad d_{nc,m} = \frac{v_{nc,m} - v_{f,m}}{v_{f,m}} \cdot 100 \quad \text{e} \quad d_{aj,m} = \frac{v_{aj,m} - v_{f,m}}{v_{f,m}} \cdot 100 \qquad (2)$$

[8] https://openbenchmarking.org/ [9] https://blender.org [10] https://locklessinc.com
[11] https://github.com/axboe/fio

Disco Leitura

$$d_{c,r} = \frac{v_{c,r} - v_{f,r}}{v_{f,r}} \cdot 100 \quad \text{e} \quad d_{nc,r} = \frac{v_{nc,r} - v_{f,r}}{v_{f,r}} \cdot 100 \quad \text{e} \quad d_{aj,r} = \frac{v_{aj,r} - v_{f,r}}{v_{f,r}} \cdot 100 \qquad (3)$$

Disco Escrita

$$d_{c,w} = \frac{v_{c,w} - v_{f,w}}{v_{f,w}} \cdot 100 \quad \text{e} \quad d_{nc,w} = \frac{v_{nc,w} - v_{f,w}}{v_{f,w}} \cdot 100 \quad \text{e} \quad d_{aj,w} = \frac{v_{aj,w} - v_{f,w}}{v_{f,w}} \cdot 100 \qquad (4)$$

O experimento consistiu em executar 30 vezes cada um dos quatro testes, em um sistema operacional `Linux Debian 11` e em um `Windows 10`, seguindo práticas estatísticas do design experimental para garantir resultados confiáveis e reduzir a influência de variáveis aleatórias. Inicialmente executamos os testes em uma máquina física (instalação *bare-metal*), depois em VMs executadas a partir do SO customizado e de um SO não customizado, além de executar o SO customizado em um ambiente com ajustes no *hypervisor*. Em cada um dos quatro ambientes (físico, customizado, não customizado e com ambiente ajustado), foram executados os *benchmarks* descritos anteriormente e os resultados estão dispostos nas figuras a seguir.

Os gráficos apresentados nas figuras possuem duas representações distintas, cada uma com sua finalidade específica para análise dos resultados obtidos. Os gráficos **"a"** são gráficos de barras, que exibem os resultados dos 30 testes realizados de forma individualizada. Essa representação detalhada permite identificar tendências, padrões ou variações específicas entre os testes, sendo útil para uma análise mais granular e comparativa dos dados. Já os gráficos **"b"** são gráficos de caixa, que fornecem uma visão sumarizada dos resultados. Essa representação facilita a interpretação geral do comportamento dos dados, permitindo comparações rápidas e a identificação de distribuições ou anomalias nos resultados.

Ao combinar gráficos de barra e de caixa, a análise ganha profundidade e clareza, possibilitando tanto uma visão ampla e resumida quanto uma avaliação detalhada e específica dos resultados dos testes. Essa abordagem oferece uma visão abrangente e complementa a interpretação dos dados com maior precisão.

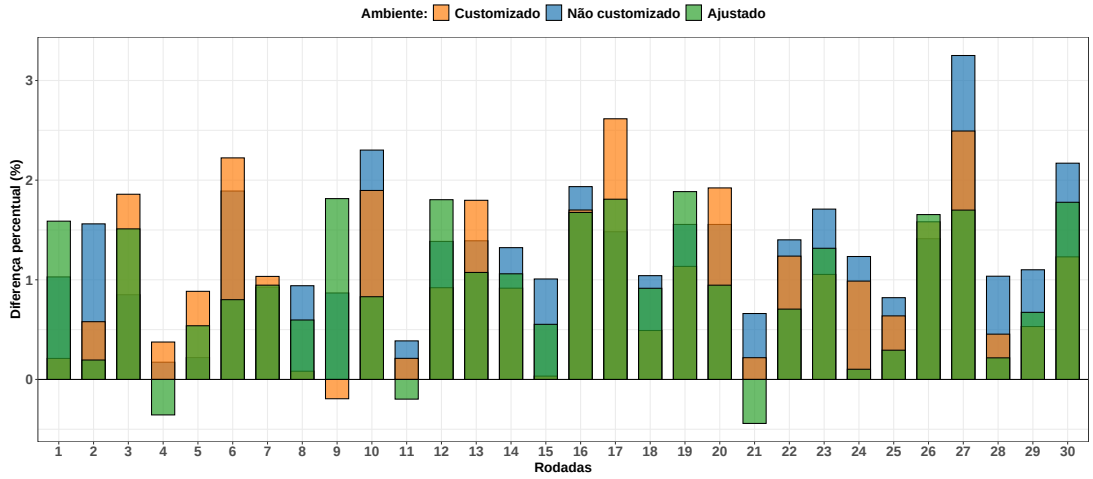
4.3.2.1 CPU

A Figura 10 ilustra a diferença percentual no sistema operacional Linux, entre o tempo de renderização na máquina física e os demais ambientes. Pode-se observar que o ambiente customizado consegue renderizar com um tempo melhor do que o não customizado. Adicionalmente, o ambiente ajustado consegue ser melhor do que o ambiente apenas customizado.

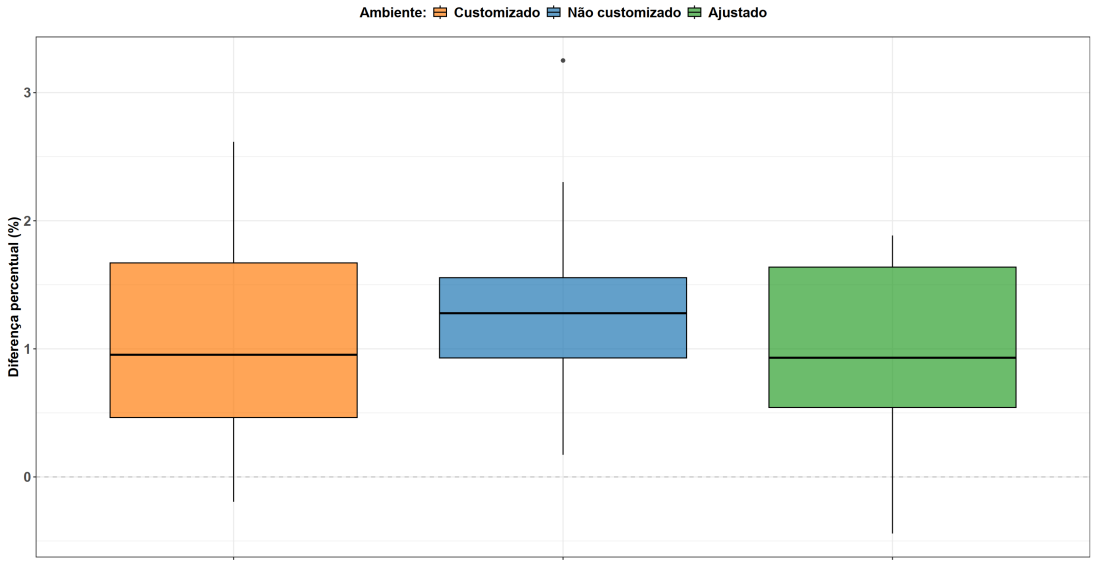
A diferença percentual média é $\overline{d_{c,p}} = 1,04\%(\pm 1,06\%)$ para o SO customizado, de $\overline{d_{nc,p}} = 1,29\%(\pm 1,08\%)$ para o não customizado e $\overline{d_{aj,p}} = 0,93\%(\pm 1,29\%)$ para o ambiente ajustado. Ou seja, em média, o SO customizado com os ajustes no *hypervisor* leva $0,93\%$ a mais de tempo para renderizar, se comparado ao tempo na máquina física.

Figura 10 – CPU Linux

(a) Gráfico de barra para cada execução.



(b) Gráfico de caixa considerando as 30 amostras.

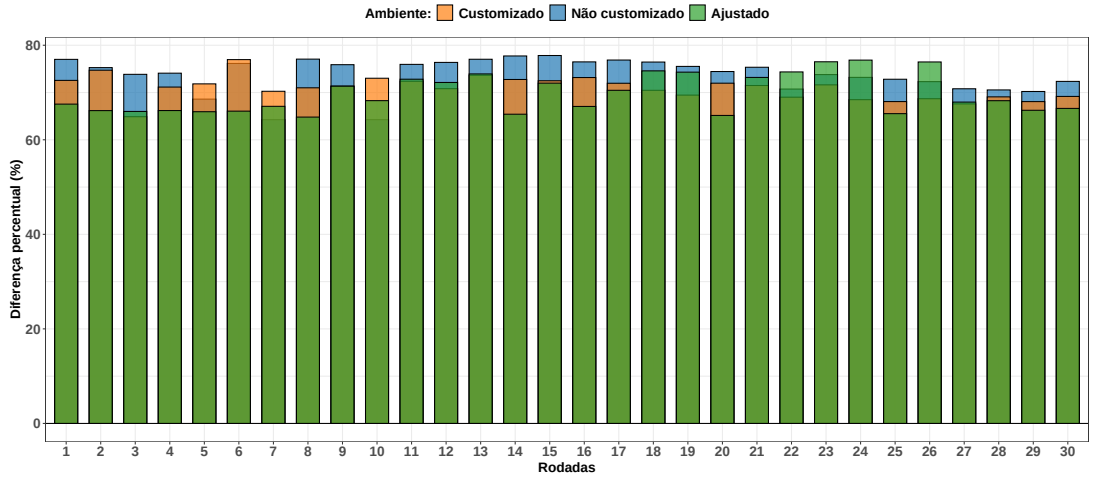


A Figura 11 ilustra a diferença percentual no sistema operacional Windows, e segue o mesmo teste e comportamento do experimento anterior. O ambiente customizado consegue renderizar com um tempo melhor do que o não customizado; novamente, o ambiente ajustado consegue ser melhor do que o ambiente apenas customizado.

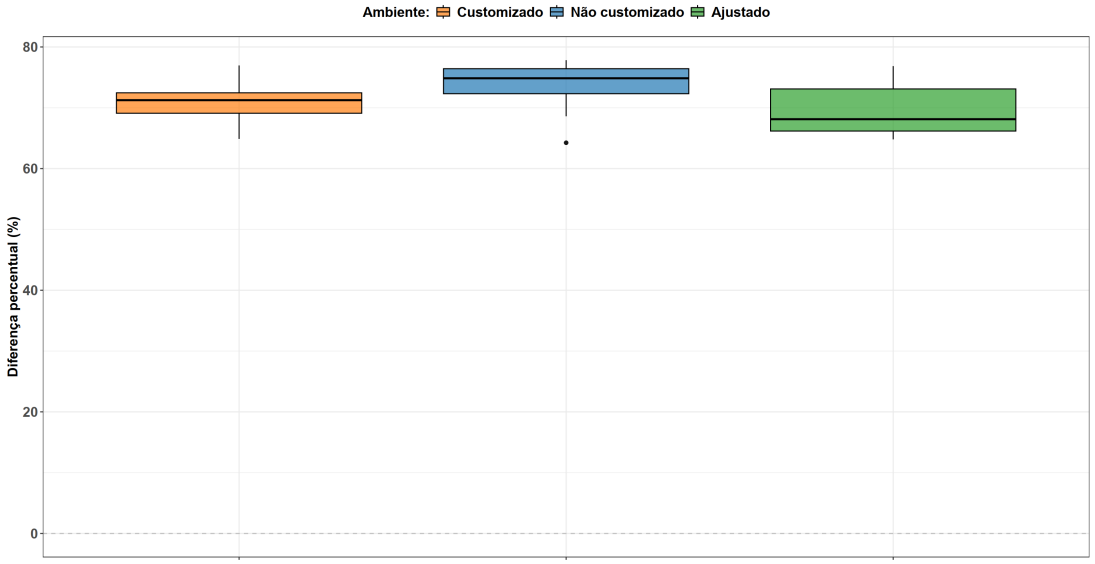
A diferença percentual média é $\overline{d_{c,p}} = 70,94\%(\pm 6,97\%)$ para o SO customizado, de $\overline{d_{nc,p}} = 73,77\%(\pm 9,96\%)$ para o não customizado e $\overline{d_{aj,p}} = 69,64\%(\pm 10,55\%)$ para o ambiente ajustado. Aqui também o SO customizado com os ajustes no *hypervisor* se sobressai ligeiramente aos demais, levando $69,64\%$ a mais de tempo para renderizar, se comparado ao tempo na máquina física.

Figura 11 – CPU Windows

(a) Gráfico de barra para cada execução.



(b) Gráfico de caixa considerando as 30 amostras.



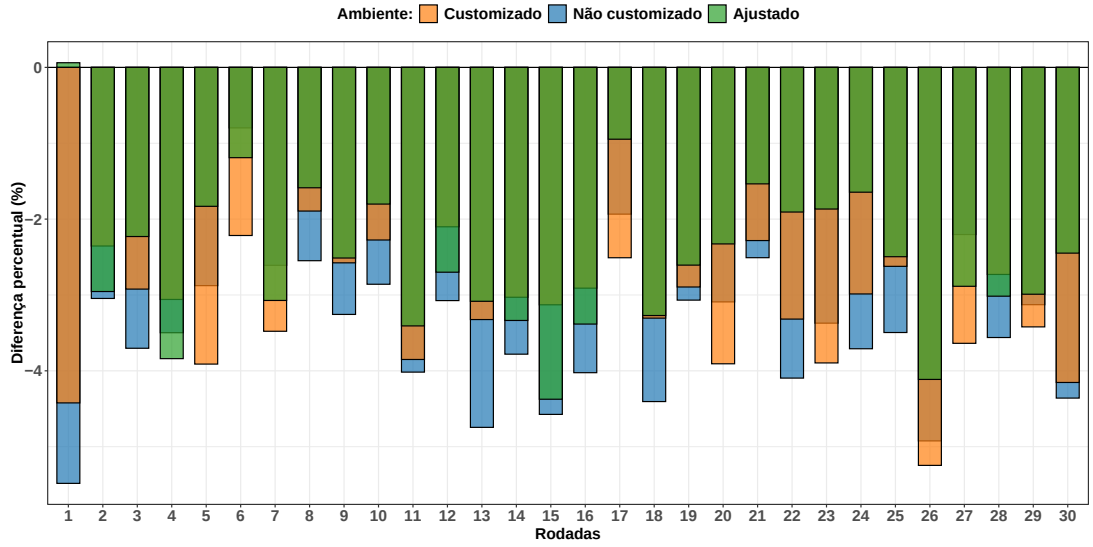
Para o teste de desempenho da CPU, observamos resultados significativamente diferentes entre os ambientes Linux e Windows.

Para investigar essa variação, decidimos realizar novos testes e os resultados são exibidos na Figura 12 e na Figura 13. Foi utilizada a ferramenta `7-Zip`[12], que permite a execução de *benchmarks* consistentes em ambos os sistemas operacionais. Essa escolha se baseou na habilidade do 7-Zip de avaliar a capacidade de processamento dos sistemas, medindo a velocidade de execução de algoritmos de compressão e descompressão, realizando testes intensivos na CPU, oferecendo uma pontuação em MIPS (milhões de instruções por segundo) que permite comparar o desempenho entre diferentes sistemas ou configurações, onde a pontuação mais alta indica um melhor desempenho. Portanto, as diferenças percentuais são negativas, apontando que a máquina física é y% mais veloz do que os demais ambientes.

[12] https://www.7-zip.org/

Figura 12 – CPU Linux (7zip)

(a) Gráfico de barra para cada execução.



(b) Gráfico de caixa considerando as 30 amostras.

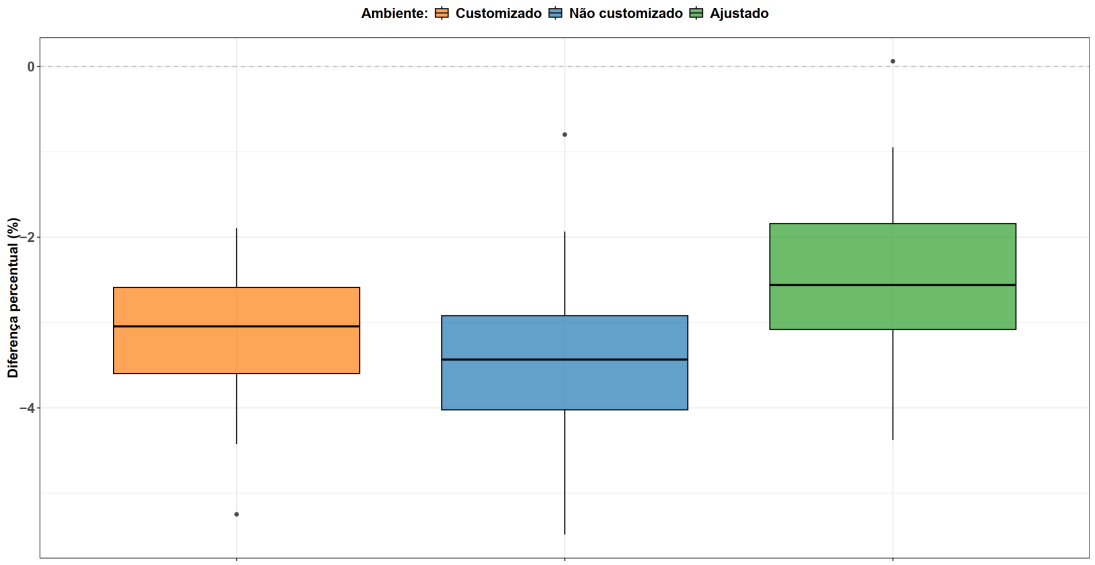
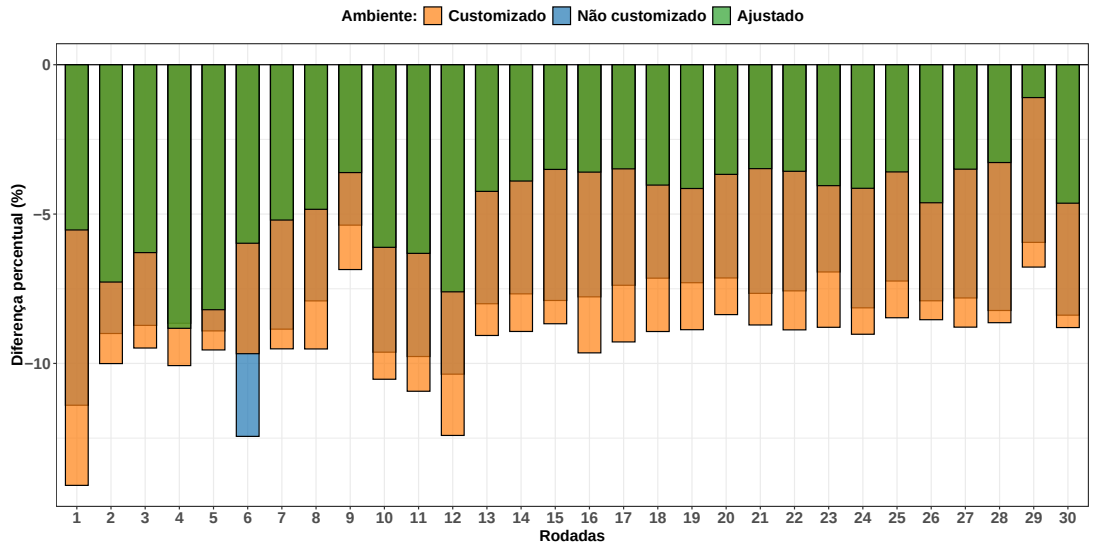
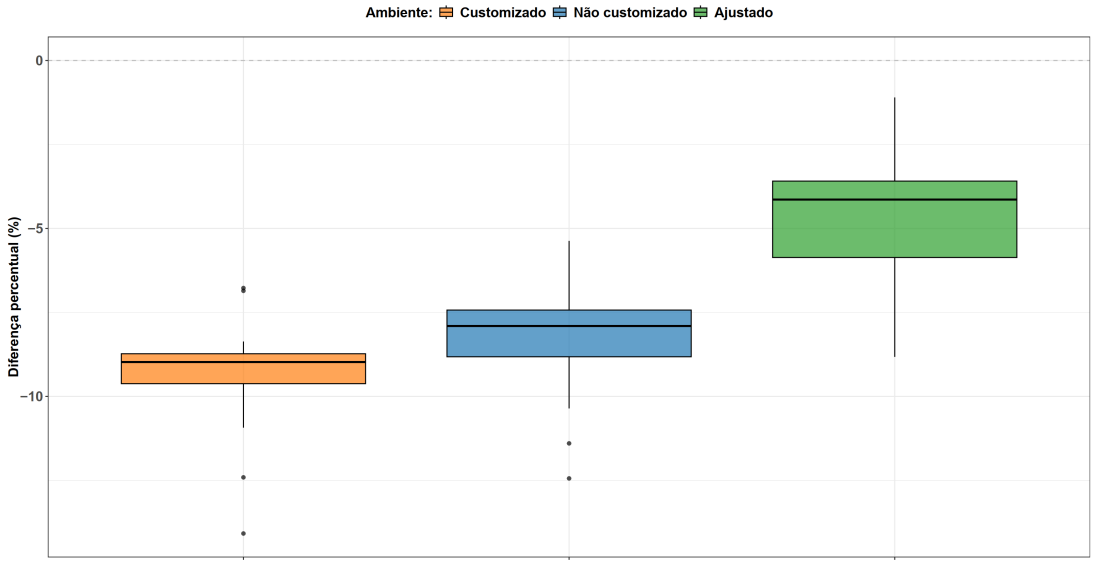


Figura 13 – CPU Windows (7zip)

(a) Gráfico de barra para cada execução.



(b) Gráfico de caixa considerando as 30 amostras.



Em paralelo, buscamos fontes que fornecessem indícios sobre o comportamento do Blender em diferentes sistemas operacionais, visando compreender as razões por trás das discrepâncias significativas observadas nos resultados de desempenho entre as plataformas testadas. A literatura existente sugere que diversos fatores podem contribuir para essas diferenças, incluindo a otimização de *drivers*, a gestão de recursos e a forma como o Blender utiliza os núcleos de CPU em cada ambiente.

Um estudo da (TECHGAGE, 2021) revela que o Linux, com seus *drivers* atualizados e um Kernel otimizado, pode oferecer um desempenho superior em tarefas de renderização em comparação ao Windows, especialmente com *hardware* de múltiplos núcleos.

Uma análise do (BLENDERNATION, 2019) discute como a implementação de otimiza-

ções específicas no Linux pode beneficiar o desempenho do Blender, evidenciando a importância de um ambiente bem configurado para maximizar a eficiência da renderização. Nos testes realizados, observou-se que o Linux foi, em média, 49% mais rápido do que o Windows, corroborando a ideia de que um ambiente otimizado pode ter um impacto significativo no desempenho.

Adicionalmente, um Gist do (GITHUB, 2019) que compilou dados de *benchmark* sobre o tempo total de renderização em várias cenas do Blender também confirma a tendência de que o Linux pode superar o Windows em desempenho, apresentando vantagens de até 22% em algumas configurações.

No (BLENDERNATION, 2019), um usuário relata nos comentários sua experiência ao migrar do Windows para o Ubuntu 20.04. Após seis anos utilizando o Blender 2.7 no Windows, ele instalou o Blender 2.80, apenas para descobrir que sua placa de vídeo (Intel HD Graphics 2000) não suportava OpenGL 3.3 devido a problemas de *drivers*. Após a mudança para o Ubuntu, o Blender 2.83 foi executado sem problemas, sem necessidade de modificações ou atualizações. O autor ficou surpreso com a diferença de desempenho, sugerindo que a capacidade do *hardware* de suportar OpenGL 3.3 sempre esteve presente, mas os *drivers* deficientes no Windows impediam sua utilização plena.

Também encontramos estudos técnicos de alguns fabricantes que apresentam comparações detalhadas do uso de VDI com processamento apenas em CPU versus ambientes com suporte a GPU, demonstrando o impacto significativo que a carga de trabalho gráfica pode ter sobre a CPU.

No artigo Intel (2023), é demonstrada uma carga de trabalho com aplicações baseadas em WebGL e reprodução de mídia. São comparadas duas sessões VDI: uma configurada com GPU virtual (vGPU) e outra utilizando apenas CPU, ambas executando o mesmo aplicativo WebGL em um navegador. As máquinas virtuais foram configuradas de maneira idêntica. Os resultados mostram que a VM com apenas CPU atinge quase 100% de utilização de processamento, tendo dificuldades em manter uma sessão VDI de 1080p, conseguindo apenas 13 quadros por segundo (fps). Em contraste, a VM equipada com vGPU oferece uma experiência mais estável para o usuário final, mantendo uma taxa de aproximadamente 30 fps, além de proporcionar uma significativa margem de recursos de CPU, com menos de 10% de utilização. Esses dados evidenciam o impacto positivo da vGPU na eficiência e na qualidade da experiência VDI, especialmente em aplicações com alta demanda gráfica, onde o desempenho é ampliado e a carga de processamento sobre a CPU é substancialmente reduzida.

O artigo da NVIDIA (2015) descreve como sua tecnologia de vGPU oferece *desktops* virtuais com gráficos de alta escalabilidade. A solução vGPU permite que várias máquinas virtuais compartilhem uma única GPU física, proporcionando aos usuários acesso a recursos gráficos avançados que antes eram limitados a *desktops* físicos ou soluções com GPU dedicada, tornando-se possível executar aplicativos gráficos intensivos, como CAD, modelagem 3D e simulação, em ambientes VDI.

Esses *insights* nos ajudam a contextualizar e interpretar os resultados obtidos nos nossos testes, contribuindo para uma análise mais aprofundada das diferenças de desempenho entre os sistemas operacionais.

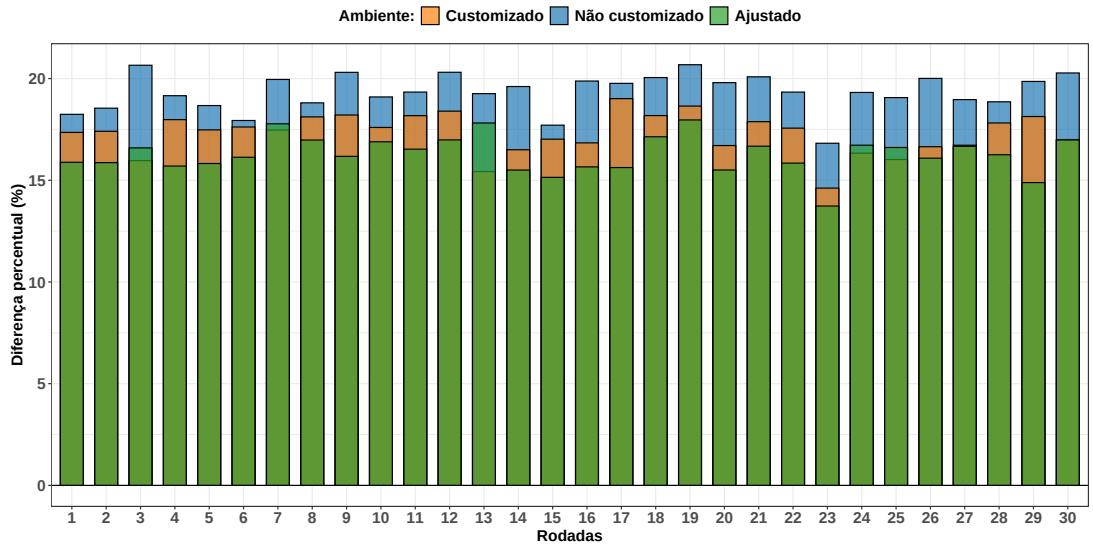
4.3.2.2 Memória

Os resultados dos testes de desempenho de memória revelaram uma melhoria significativa na versão customizada do sistema operacional, otimizada com ajustes específicos no *hypervisor*, em comparação com os demais ambientes testados.

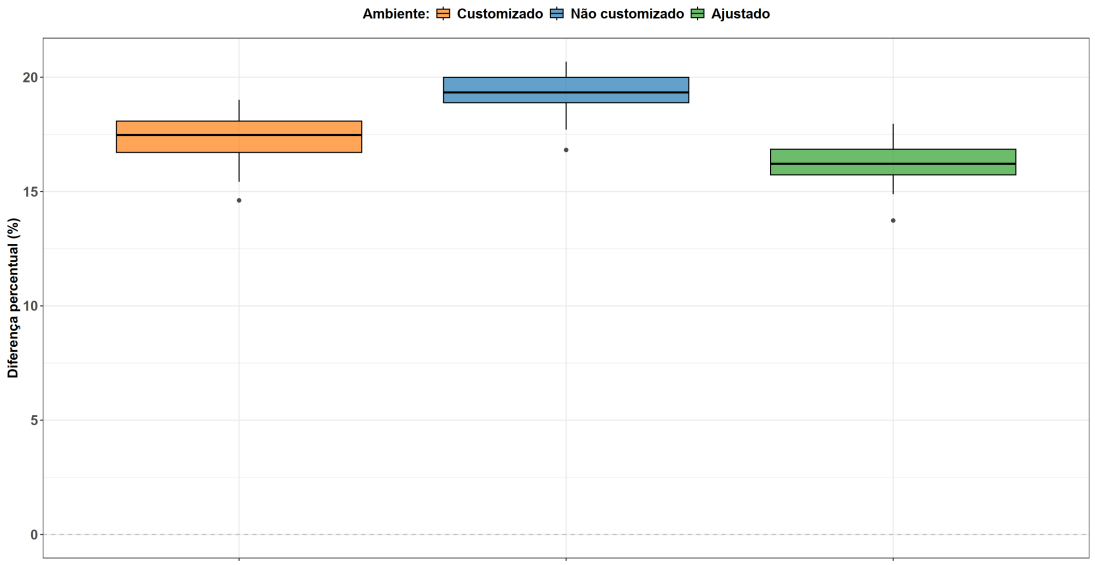
A Figura 14 ilustra a diferença percentual no sistema operacional Linux, cuja diferença média do ambiente customizado em relação à máquina física é $\overline{d_{c,m}} = 17,29\%(\pm 0,03\%)$. Já nos ambientes não customizado e ajustado obtivemos os respectivos valores $\overline{d_{nc,m}} = 19,35\%(\pm 0,03\%)$ e $\overline{d_{aj,m}} = 16,27\%(\pm 0,02\%)$, provando que os ajustes no *hypervisor* também contribuíram para uma redução no uso de memória, proporcionando um ambiente mais estável e com melhor desempenho.

Figura 14 – Memória Linux

(a) Gráfico de barra para cada execução.



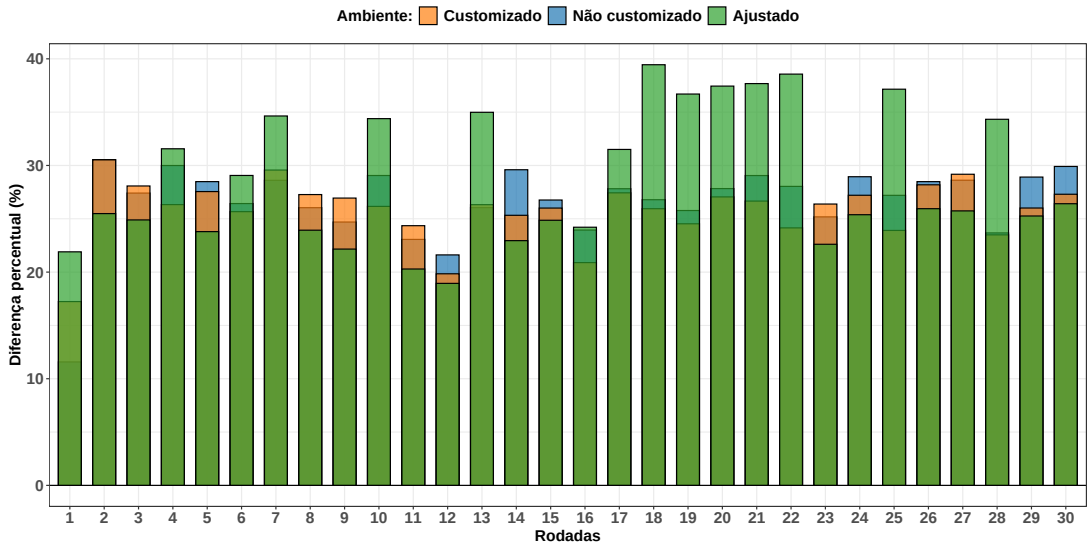
(b) Gráfico de caixa considerando as 30 amostras.



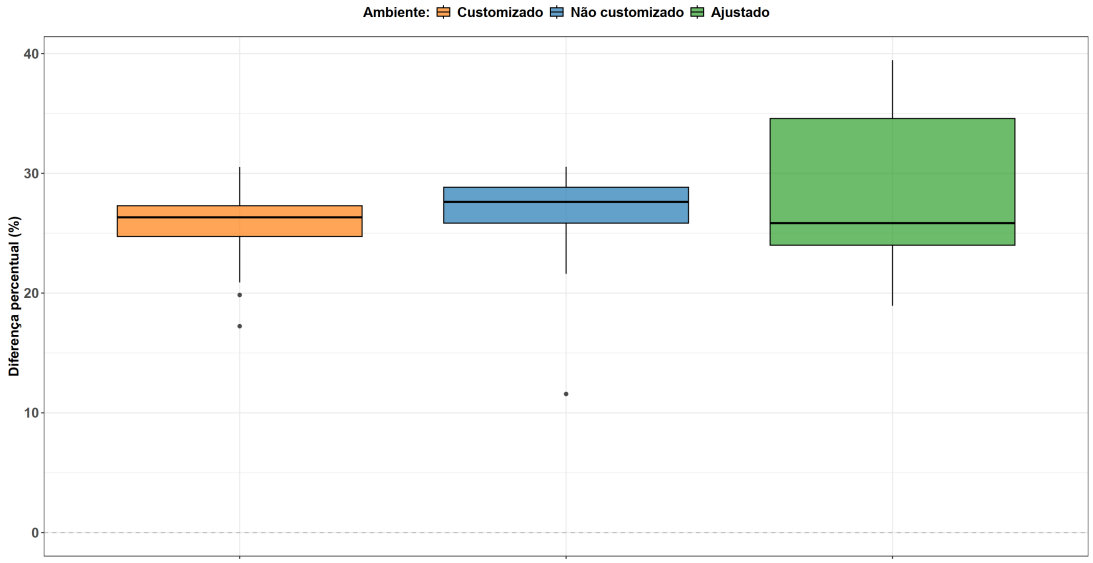
A Figura 15 ilustra a diferença percentual no sistema operacional Windows, cujas diferenças entre os ambientes seguem o mesmo padrão do Linux, apesar do primeiro consumir mais memória. No ambiente customizado a média em relação à máquina física foi de $\overline{d_{c,m}} = 25,85\%(\pm 0,12\%)$, no ambiente não customizado foi de $\overline{d_{nc,m}} = 26,67\%(\pm 0,07\%)$ e no ambiente ajustado foi $\overline{d_{aj,m}} = 28,74\%(\pm 0,38\%)$.

Figura 15 – Memória Windows

(a) Gráfico de barra para cada execução.



(b) Gráfico de caixa considerando as 30 amostras.



Assim como para o processamento, percebe-se que a gerência de memória utilizando o Windows virtualizado é menos eficiente do que no Linux, suprimindo inclusive os efeitos da customização e ajustes realizados.

4.3.2.3 Disco

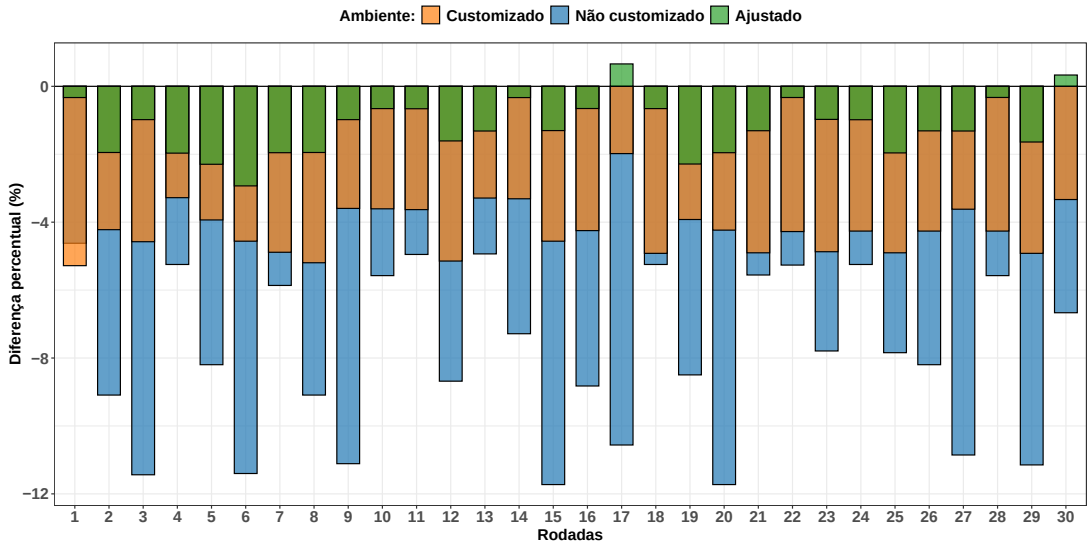
Por fim, temos os resultados dos testes de leitura e escrita em disco que, assim como os demais testes, se mostraram com um desempenho superior no ambiente ajustado.

A Figura 16 e a Figura 17 medem a velocidade com que a ação é executada no sistema operacional Linux e, portanto, as diferenças percentuais são negativas, apontando que a máquina física é *y%* mais veloz do que os demais ambientes, salvo alguns poucos resultados onde o

ambiente ajustado foi ainda melhor do que o físico. No ambiente customizado a média em relação à máquina física foi de $\overline{d_{c,m}} = 4,20\%(\pm 0,74\%)$, no ambiente não customizado foi de $\overline{d_{nc,m}} = 7,94\%(\pm 2,63\%)$ e no ambiente ajustado foi $\overline{d_{aj,m}} = 1,20\%(\pm 0,75\%)$.

Figura 16 – Disco Leitura Linux

(a) Gráfico de barra para cada execução.



(b) Gráfico de caixa considerando as 30 amostras.

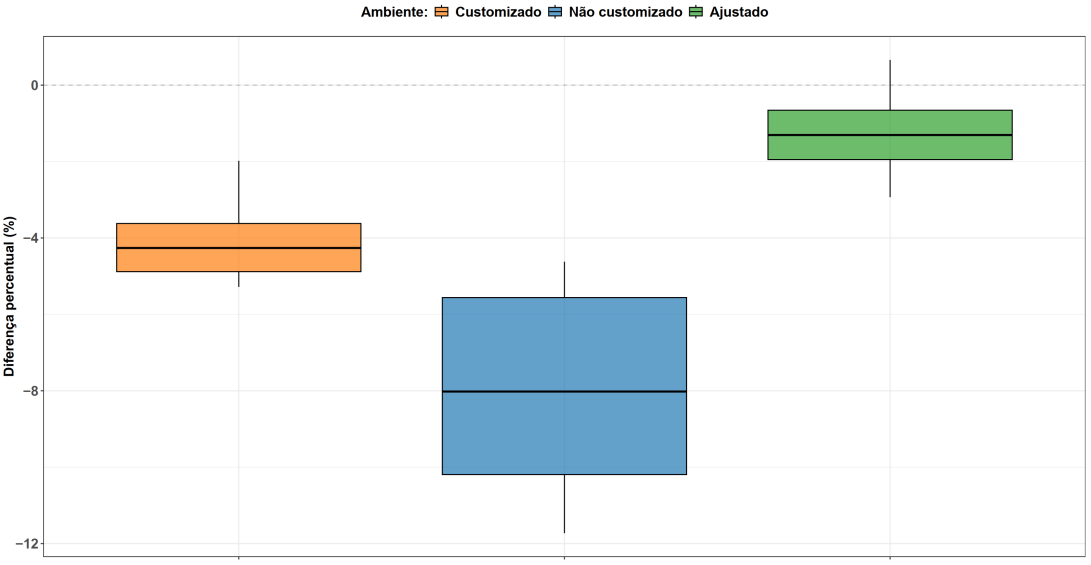
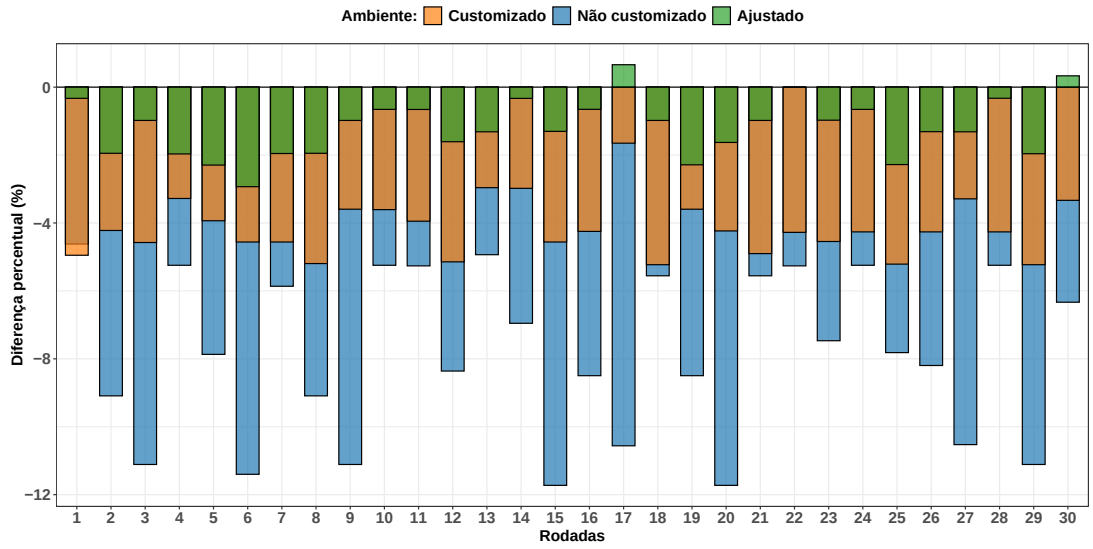
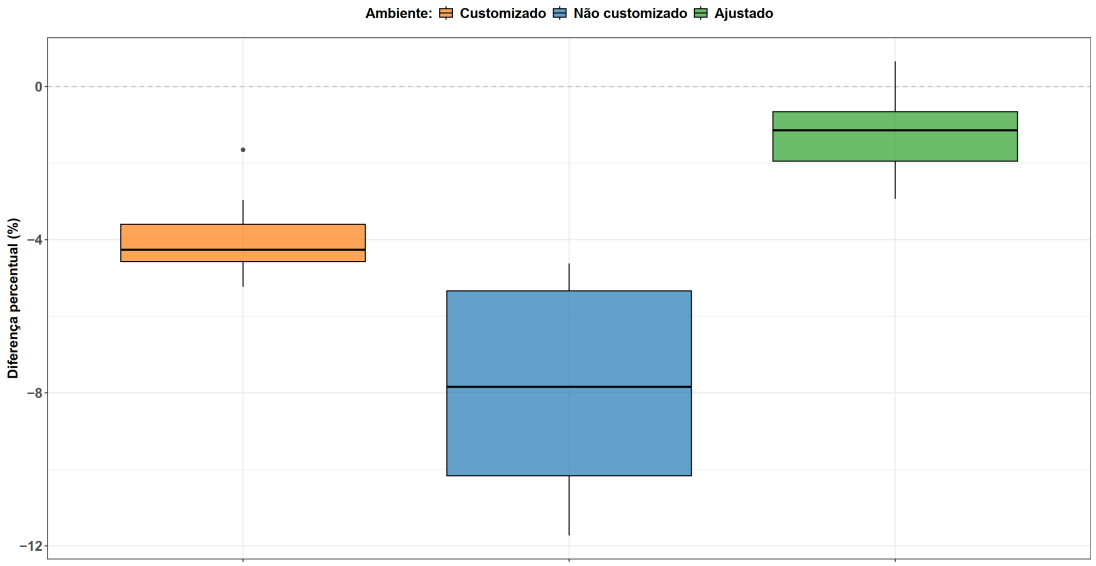


Figura 17 – Disco Escrita Linux

(a) Gráfico de barra para cada execução.



(b) Gráfico de caixa considerando as 30 amostras.



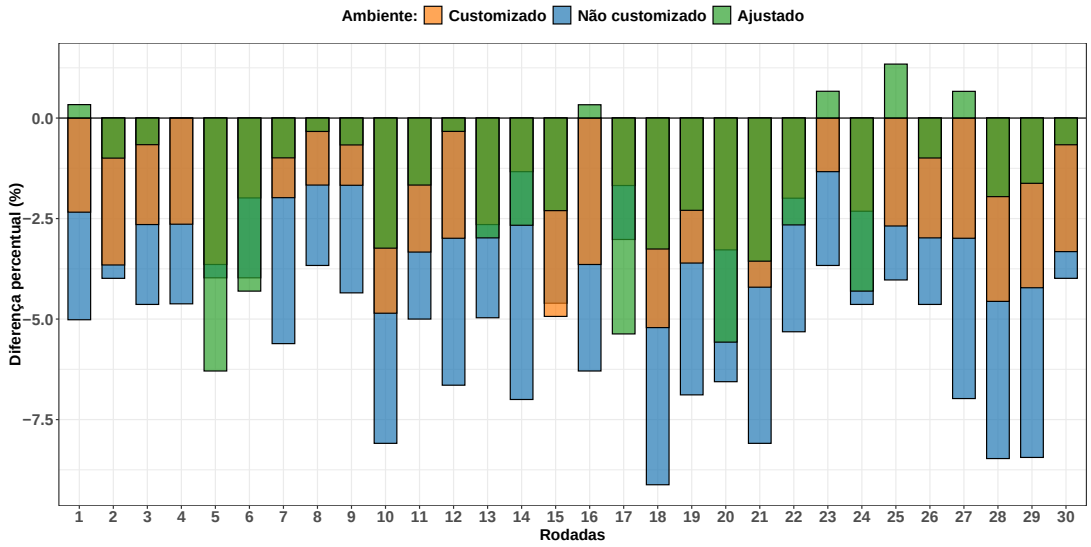
Ao analisarmos as figuras de desempenho de leitura e escrita em cada ambiente, observamos que os valores obtidos para ambas as operações são essencialmente equivalentes. Essa paridade decorre da configuração específica do teste, que foi ajustado para realizar 50% das operações em leitura e 50% em escrita. Esse balanceamento intencional permite uma avaliação equilibrada do subsistema de armazenamento em ambos os aspectos, garantindo que o impacto das otimizações seja avaliado de forma uniforme entre as operações de I/O.

A Figuras 18 e a Figura 19 medem a velocidade com que a ação é executada no sistema operacional Windows. As diferenças percentuais também são negativas, apontando que a máquina física é y% mais veloz do que os demais ambientes, novamente, salvo alguns resultados com o ambiente ajustado. No ambiente customizado, a média em relação à máquina física foi de

$\overline{d_{c,m}} = 3,01\%(\pm 0,82\%)$, no ambiente não customizado foi de $\overline{d_{nc,m}} = 5,54\%(\pm 1,27\%)$ e no ambiente ajustado foi $\overline{d_{aj,m}} = 1,88\%(\pm 2,14\%)$.

Figura 18 – Disco Leitura Windows

(a) Gráfico de barra para cada execução.



(b) Gráfico de caixa considerando as 30 amostras.

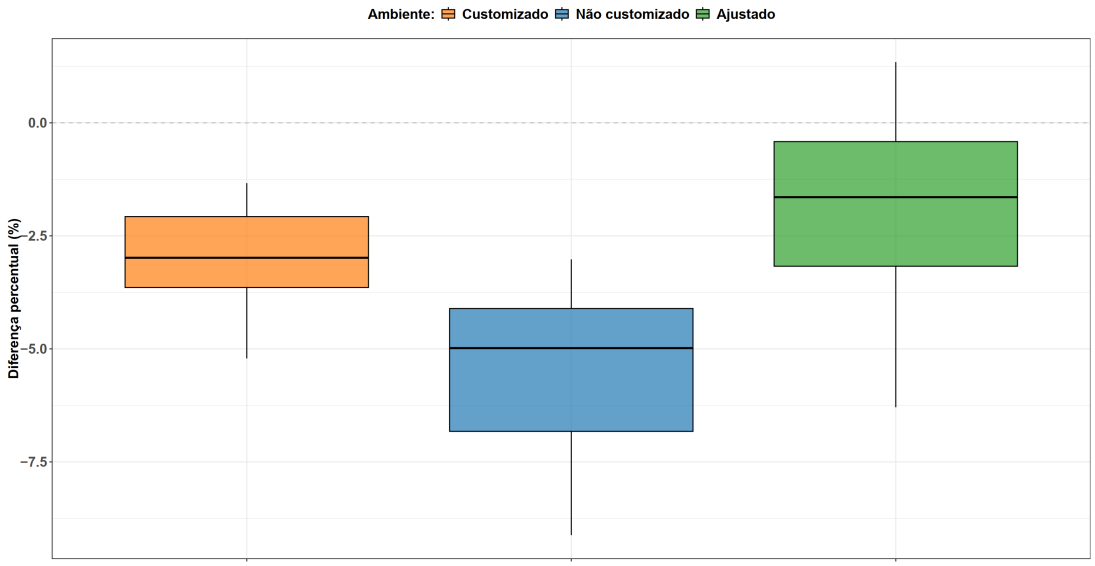
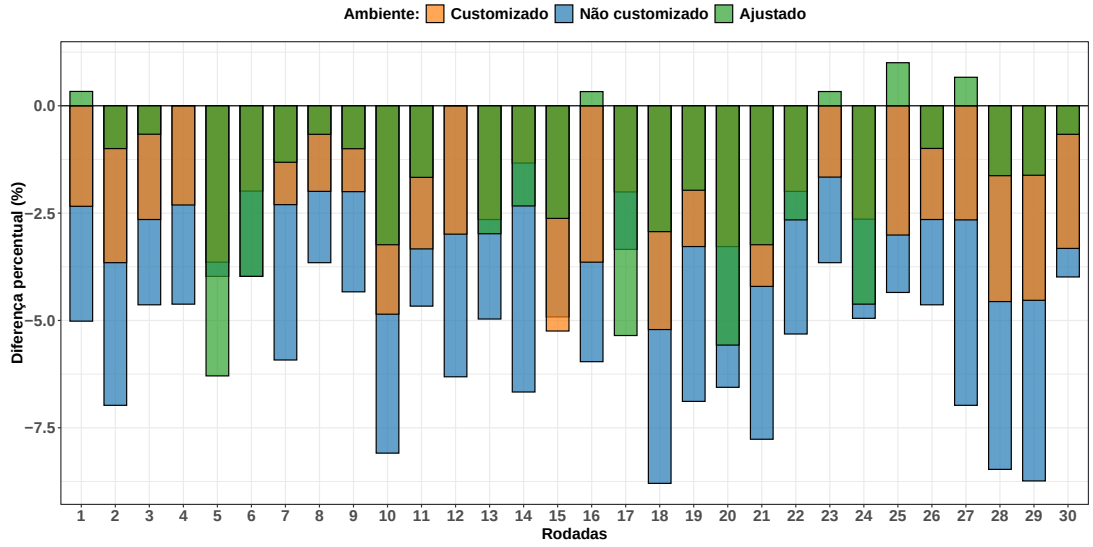
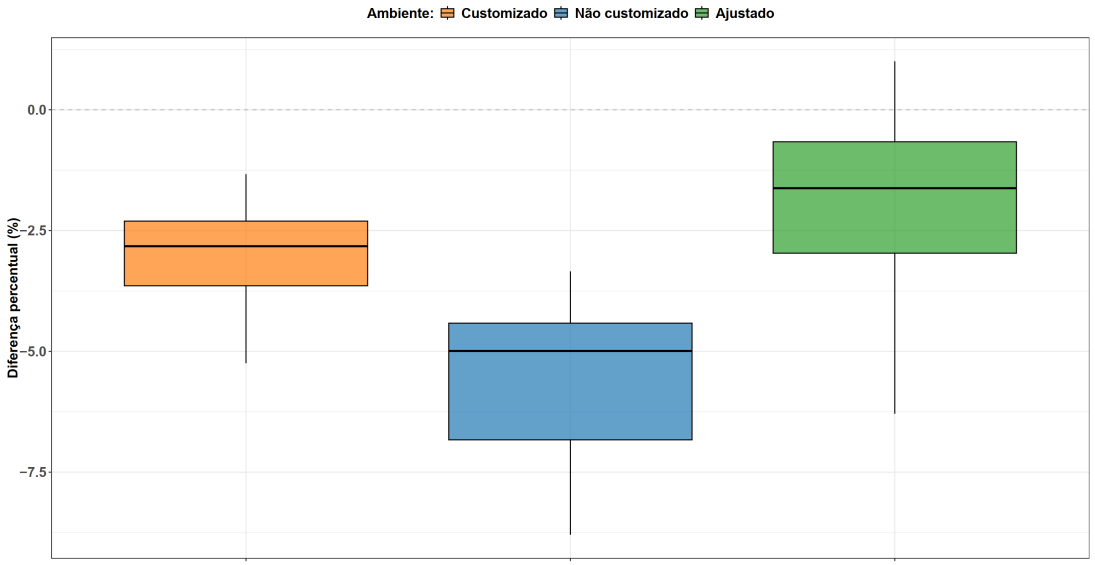


Figura 19 – Disco Escrita Windows

(a) Gráfico de barra para cada execução.



(b) Gráfico de caixa considerando as 30 amostras.



A Tabela 5 apresenta uma análise abrangente dos resultados dos diferentes testes de desempenho realizados nos três ambientes avaliados: "Customizado", "Não Customizado"e "Ajustado". Os resultados são expressos em termos de valores médios e desvios padrão, permitindo uma comparação detalhada do desempenho entre as configurações para cada tipo de teste.

Tabela 5 – Testes de Desempenho

Teste	Ambiente					
	Customizado		Não Customizado		Ajustado	
	Média	Desvio Padrão	Média	Desvio Padrão	Média	Desvio Padrão
CPU Linux	1,04%	0,76%	1,29%	0,63%	**0,93%**	0,70%
CPU Windows	70,94%	2,42%	73,77%	3,59%	**69,64%**	3,97%
CPU Linux 7zip	-3,15%	0,76%	-3,42%	0,96%	**-2,51%**	0,97%
CPU Windows 7zip	-9,33%	1,38%	-8,24%	1,44%	**-4,74%**	1,69%
Memória Linux	17,29%	0,99%	19,35%	0,90%	**16,27%**	0,90%
Memória Windows	**25,85%**	2,78%	26,67%	3,60%	28,74%	6,26%
Disco Leitura Linux	-4,20%	0,74%	-7,94%	2,45%	**-1,20%**	0,82%
Disco Escrita Linux	-4,15%	0,83%	-7,85%	2,43%	**-1,19%**	0,85%
Disco Leitura Windows	-3,01%	1,09%	-5,54%	1,70%	**-1,88%**	1,99%
Disco Escrita Windows	-3,06%	1,06%	-5,64%	1,62%	**-1,88%**	1,93%

Como podemos observar, o ambiente ajustado apresenta desempenho superior na maioria dos testes realizados, destacando-se como a configuração mais eficiente entre as opções avaliadas. No entanto, uma exceção notável ocorre no teste de memória no sistema operacional Windows, onde ele não supera os outros ambientes. Essa discrepância pode estar associada a características específicas da implementação do ambiente ajustado ou particularidades no gerenciamento de memória do Windows.

Em síntese, os resultados dos testes realizados confirmam melhorias significativas no desempenho do ambiente ajustado em comparação com as configurações dos demais ambientes.

4.4 Considerações sobre os Resultados

Embora os testes realizados com o Blender no sistema operacional Windows não tenham alcançado o desempenho esperado, os resultados obtidos no teste com o 7zip evidenciam melhorias substanciais no ambiente com os ajustes no *hypervisor* em comparação com os demais. Esses resultados indicam que, apesar das limitações observadas em algumas aplicações específicas, as otimizações implementadas foram eficazes em proporcionar ganhos de desempenho, refletindo um ambiente mais eficiente e responsivo.

Nos testes de memória, o ambiente com ajustes no *hypervisor* apresentou menor latência e maior largura de banda, permitindo tempos de resposta mais rápidos e uma melhor eficiência no uso, o que é particularmente vantajoso para aplicações de alta demanda de memória.

Adicionalmente, os testes de disco evidenciaram um aumento na taxa de leitura e escrita; essa otimização no subsistema de armazenamento garante maior estabilidade e fluidez em operações de I/O intensivas, proporcionando um ambiente confiável e de alto desempenho para aplicações críticas.

De maneira geral, esses resultados validam a eficácia das otimizações aplicadas na configuração “ajustada”, destacando ganhos substanciais em eficiência, estabilidade e desempenho em ambientes virtualizados. No entanto, os resultados obtidos neste estudo, embora promissores e

alinhados com as expectativas iniciais, demonstram que ainda há espaço para melhorias significativas. Foram identificadas limitações que sugerem oportunidades de refinamento, especialmente no desempenho da memória no SO Windows virtualizado. Esse aspecto ainda carece de avanços no estado da arte e indica um promissor caminho de pesquisa. Testes empíricos em ambientes reais de aula servirão para ratificar os benefícios dos ajustes realizados e também serão objeto de trabalhos futuros de extensão da pesquisa.

5 CONSIDERAÇÕES FINAIS

Está pesquisa fundamenta-se na premissa de que laboratórios voltados ao ensino e à pesquisa podem ser geridos de forma eficiente com o uso de tecnologias de virtualização de código aberto. O objetivo é não apenas promover a economia no uso de recursos públicos, mas também maximizar a utilização de equipamentos legados, considerando que soluções proprietárias são geralmente caras e exigem infraestrutura específica. Esse cenário é observado em algumas instituições públicas no Brasil, onde a gestão desses recursos é frequentemente realizada de maneira não otimizada.

Na prova de conceito apresentada no capítulo anterior, de toda a arquitetura proposta, foram utilizados apenas os módulos **MImg**, como um servidor web para hospedagem das imagens; **MVirt** para configurar o ambiente virtual e **MGer** com uma solução web para representar a centralização da solução em uma única interface. Além disso, foram escolhidas algumas ferramentas para a PoC por critério de simplicidade, como `KVM`, `Ansible`, `AWX`, `Apache HTTP Server`.

Para a continuação deste projeto de pesquisa, uma investigação completa de soluções alternativas às ferramentas já utilizadas é pretendida, bem como de novas ferramentas para compor a arquitetura do projeto. Deverá ser realizada uma comparação entre as ferramentas investigadas, citando os pontos positivos e negativos de cada uma no contexto da arquitetura `vCLASS`.

Embora o KVM tenha sido escolhido como plataforma de virtualização principal para este projeto, o Xen também pode ser considerado para testes e comparações em uma fase posterior da pesquisa. Essa abordagem permitirá uma análise comparativa entre as duas tecnologias, possibilitando uma avaliação mais abrangente de seus desempenhos, funcionalidades e adequações a diferentes cenários de uso. A inclusão do Xen em um momento posterior possibilita uma visão mais completa dos prós e contras de cada plataforma, permitindo observar como suas arquiteturas e abordagens impactam o desempenho e a eficiência do ambiente virtualizado. Essa análise adicional pode enriquecer os resultados do projeto, contribuindo para uma avaliação mais profunda e fundamentada das opções de virtualização disponíveis.

A escolha por limitar o estudo a três tecnologias avançadas de virtualização foi motivada pela necessidade de otimizar os recursos disponíveis e superar as limitações de suporte ao *hardware*. Através da implementação dessas técnicas, foi possível observar ganhos significativos na alocação de recursos, no gerenciamento de memória e na eficiência de comunicação entre máquinas virtuais e o hipervisor.

Embora os testes tenham sido restritos, os resultados foram satisfatórios, melhorando a performance dos ambientes virtualizados. Há a possibilidade de expandir os testes para incluir

outros recursos e tecnologias de virtualização. Com essa abordagem, será possível explorar e comparar novas técnicas e funcionalidades que surgirem, obtendo uma visão ainda mais abrangente e detalhada sobre o desempenho e a eficiência das máquinas virtuais. Essa ampliação futura permitirá uma adaptação contínua do projeto, contribuindo para uma pesquisa mais completa e alinhada com as necessidades do projeto.

E também permitirá uma evolução significativa do projeto, não apenas mantendo-o alinhado com as necessidades acadêmicas e tecnológicas, mas também ampliando seu impacto ao proporcionar uma base sólida para futuras implementações em instituições públicas e privadas. Além disso, contribuirá para o avanço do conhecimento na área de virtualização, oferecendo soluções inovadoras e otimizadas para a gestão de recursos em ambientes educacionais e de pesquisa. A incorporação de novas tecnologias e a realização de comparativos aprofundados garantirão que o projeto permaneça relevante e atualizado, atendendo às demandas crescentes por eficiência, economia e sustentabilidade no uso de recursos computacionais.

A pesquisa também resultou na submissão de um artigo ao `SEMISH'23`[1] (Seminário Integrado de Software e Hardware), um dos principais fóruns científicos promovidos pelo `CSBC`[2] (Congresso da Sociedade Brasileira de Computação). O artigo foi aceito, reforçando o alinhamento do trabalho com os requisitos e normas do Programa de Pós-Graduação em Tecnologia da Informação do IFPB. Esse reconhecimento ressalta a relevância da pesquisa e sua contribuição para a área de tecnologias de virtualização e gestão de recursos em ambientes acadêmicos.

[1] https://csbc.sbc.org.br/2023/semish/ [2] https://csbc.sbc.org.br

REFERÊNCIAS BIBLIOGRÁFICAS

ABDULRAHEEM, W. K. Performance comparison of xen and hyper-v in cloud computing while using cryptosystems. *Journal of Cloud Computing*, Springer, 2024. Citado na página 35.

BIS, D.; BARAN, K.; KULAWSKA, O. Performance comparison of different versions of windows and linux operating systems. *Advances in Web Development Journal*, Patryk Organiściak, v. 1, p. 107–119, 2023. Citado na página 35.

BLENDERNATION. *Windows vs Linux - In-depth Blender Benchmark Comparison*. 2019. Disponível em: <https://www.blendernation.com/2019/09/23/windows-vs-linux-in-depth-blender-benchmark-comparison/>. Acesso em: 02 novembro 2024. Citado 2 vezes nas páginas 52 e 53.

BOAVENTURA, R. et al. Analise de desempenho de algoritmos em ambientes virtualizados em cloud computing. *Conferência IADIS Ibero-Americana WWW/Internet*, 01 2013. Citado na página 21.

DJORDJEVIC, B. et al. File system performance comparison in full hardware virtualization with esxi, kvm, hyper-v and xen hypervisors. *Advances in Electrical and Computer Engineering*, v. 21, p. 2021, 03 2021. Citado na página 36.

DJORDJEVIć, B. et al. Comparison of vmware workstation, virtualbox and ms hyper-v hypervisor performance with ms windows os based guests. In: *2023 22nd International Symposium INFOTEH-JAHORINA (INFOTEH)*. [S.l.: s.n.], 2023. p. 1–5. Citado na página 31.

FIASE, D. et al. Evaluation and comparison of the performance of different hypervisors under various workloads of giadec. *International Journal of Innovative Science and Research Technology (IJISRT)*, p. 2870–2874, 09 2024. Citado na página 36.

GENKOV, D.; SLAVOV, M. Implementation of a virtual laboratory for computer oriented disciplines. In: IEEE. *2021 29th Telecommunications Forum (TELFOR)*. [S.l.], 2021. p. 1–4. Citado na página 34.

GITHUB. *Benchmarking Windows vs. Linux with Blender Benchmark*. 2019. Disponível em: <https://gist.github.com/sethgoldin/e1a393cda096f37b140fc4ea595a7ae6>. Acesso em: 02 novembro 2024. Citado na página 53.

GITLAB. *What is version control?* 2023. Disponível em: <https://about.gitlab.com/topics/version-control>. Acesso em: 22 abril 2023. Citado na página 27.

GONZáLEZ, J. A. et al. Toward optimal virtualization: An updated comparative analysis of docker and lxd container technologies. *MDPI*, v. 12, n. 4, p. 198, 2022. Disponível em: <https://www.mdpi.com/1999-5903/10/4/198>. Citado na página 35.

HENNESSY, J. L.; PATTERSON, D. A. *Computer Architecture: A Quantitative Approach*. 6th. ed. [S.l.]: Morgan Kaufmann, 2019. Citado na página 30.

HUANG, A. Teaching, learning, and assessment with virtualization technology. *Journal of Educational Technology Systems*, SAGE Publications Sage CA: Los Angeles, CA, v. 47, n. 4, p. 523–538, 2019. Citado na página 34.

INTEL. *Overview and Performance of VDI on Intel® Data Center GPU Flex Series*. 2023. Disponível em: <https://www.intel.com/content/www/us/en/products/docs/discrete-gpus/data-center-gpu/flex-series/vdi-performance-white-paper.html>. Acesso em: 09 novembro 2024. Citado na página 53.

JAIN, R. *The Art of Computer Systems Performance Analysis: Techniques for Experimental Design, Measurement, Simulation, and Modeling*. [S.l.]: John Wiley & Sons, 1991. Citado na página 31.

KARIM, N. A. et al. Performance comparison of hyper-v and kvm for cryptographic tasks in cloud computing. *Computers, Materials and Continua*, v. 78, n. 2, p. 2023–2045, 2024. ISSN 1546-2218. Disponível em: <https://www.sciencedirect.com/science/article/pii/S1546221824000894>. Citado na página 36.

KRASSMANN, A. L. et al. Avaliação de implementações de computação em nuvem para ambientes educacionais: um mapeamento sistemático. *Novas Ideas em Informática Educativa. TISE*, p. 194–205, 2014. Citado na página 33.

LI, Z. Comparison between common virtualization solutions: Vmware workstation, hyper-v and docker. In: *2021 IEEE 3rd International Conference on Frontiers Technology of Information and Computer (ICFTIC)*. [S.l.: s.n.], 2021. p. 701–707. Citado na página 31.

MARTINS, M. A. T. *Infrastructure as Code: Automatização do aprovisionamento e configuração de Infraestrutura e Serviços no Ensino Superior*. Tese (Masters Dissertation) — Instituto politécnico de Viseu, 2022. Citado na página 34.

MASEK, P. et al. Unleashing full potential of ansible framework: University labs administration. In: IEEE. *2018 22nd conference of open innovations association (FRUCT)*. [S.l.], 2018. p. 144–150. Citado na página 34.

MATOS, E. de et al. Integrating virtio and qemu on sel4 for enhanced devices virtualization support. In: *2023 IEEE 22nd International Conference on Trust, Security and Privacy in Computing and Communications (TrustCom)*. [S.l.: s.n.], 2023. p. 1076–1085. Citado na página 26.

MORENO, F.; BARBOSA, C. de; MANFIO, E. Visual tahs: software para auxiliar o ensino de tabelas hash na disciplina de estrutura de dados. In: *Anais do XLVI Seminário Integrado de Software e Hardware*. Porto Alegre, RS, Brasil: SBC, 2019. p. 33–44. ISSN 2595-6205. Disponível em: <https://sol.sbc.org.br/index.php/semish/article/view/6565>. Citado na página 16.

MORRIS, K. *Infrastructure as Code: Managing Servers in the Cloud*. [S.l.]: O'Reilly Media, Inc, 2016. v. 1. Citado na página 26.

NVIDIA. *NVIDIA GRID vGPU: DELIVERING SCALABLE GRAPHICS-RICH VIRTUAL DESKTOPS*. 2015. Disponível em: <https://images.nvidia.com/content/pdf/grid/whitepaper/NVIDIA-GRID-WHITEPAPER-vGPU-Delivering-Scalable-Graphics-Rich-Virtual-Desktops.pdf>. Acesso em: 09 novembro 2024. Citado na página 53.

RANGAVITTALA, S.; SANJAY, H.; SALVI, S. Enhanced multi-tenant architecture for daas, paas, iaas and saas in edu-cloud: Simplifying the service provisioning in edu-cloud by multi-tenant architecture. In: *Proc. of the Sixth International Conference on Computer and Communication Technology*. [S.l.: s.n.], 2015. p. 51–56. Citado na página 33.

REDHAT. *O que é infraestrutura como código (IaC)*. 2023. Disponível em: <https://www.redhat.com/pt-br/topics/automation/what-is-infrastructure-as-code-iac>. Acesso em: 22 abril 2023. Citado na página 26.

ROT, A.; CHROBAK, P.; SOBINSKA, M. Optimisation of the use of it infrastructure resources in an institution of higher education: A case study. In: IEEE. *2019 9th International Conference on Advanced Computer Information Technologies (ACIT)*. [S.l.], 2019. p. 171–174. Citado na página 34.

RUBIO, M. S.; CIVERA, G. L.; HERRAIZ, J. J. M. Automatic generation of virtual machines for security training. *IEEE Latin America Transactions*, IEEE, v. 14, n. 6, p. 2795–2800, 2016. Citado na página 33.

RUSSELL, R.; TSIRKIN, M. S.; HUCK, C. Virtio: Towards a De-Facto Standard For Virtual I/O Devices. In: *Proceedings of the Ottawa Linux Symposium*. Linux Foundation, 2014. p. 3–20. Disponível em: <https://www.kernel.org/doc/ols/2014/ols2014-virtio.pdf>. Citado na página 37.

SAMANI, D. G. et al. The art of cpu-pinning: Evaluating and improving the performance of virtualization and containerization platforms. *CoRR*, abs/2006.02055, 2020. Disponível em: <https://arxiv.org/abs/2006.02055>. Citado na página 36.

SAMPAIO, F. et al. Labvad: Uma proposta de consórcio nacional para laboratórios didáticos a distância de ciências e robótica. In: *Anais do XLII Seminário Integrado de Software e Hardware*. Porto Alegre, RS, Brasil: SBC, 2015. p. 119–129. ISSN 2595-6205. Disponível em: <https://sol.sbc.org.br/index.php/semish/article/view/10204>. Citado na página 16.

SHUKHMAN, A. et al. Cloud educational resource datacenter simulator. *Procedia Computer Science*, Elsevier, v. 103, p. 543–548, 2017. Citado na página 33.

TECHGAGE. *Blender 2.92 Linux & Windows Performance: Best CPUs & Graphics Cards*. 2021. Disponível em: <https://techgage.com/article/blender-2-92-linux-windows-performance/>. Acesso em: 02 novembro 2024. Citado na página 52.

VERAS, M. *Virtualização de Servidores*. Rio de Janeiro, Brasil: RNP/ESR, 2011. 422 p. Il. ; 28 cm. Citado na página 23.

VERAS, M. *Virtualização: Componente Central do Datacenter*. [S.l.]: Brasport Livros e Multimidia Ltda, 2016. v. 2. Citado 2 vezes nas páginas 21 e 22.

XIA, X. et al. Optimizing multi-path quic protocol performance in cloud-edge collaboration through cpu affinity and kernel-level resource scheduling. In: *2024 IEEE/CIC International Conference on Communications in China (ICCC)*. [S.l.: s.n.], 2024. p. 1269–1274. Citado na página 25.

XIONG, N. et al. Design and research of hybrid cloud desktop scheme in colleges and universities. In: EDP SCIENCES. *MATEC Web of Conferences*. [S.l.], 2021. v. 336, p. 05004. Citado na página 34.

ĐORđEVIC, B.; MARJANOVIć, M.; TIMčENKO, V. Performance comparison of native host and hyper-based kvm virtualization. In: *2020 28th Telecommunications Forum (TELFOR)*. [S.l.: s.n.], 2020. p. 1–4. Citado na página 31.

ĐORđEVIć, B.; KRALJEVIć, N. Hyper-v as type-2 hypervisor virtualization: guest file system performance examination. In: *2023 31st Telecommunications Forum (TELFOR)*. [S.l.: s.n.], 2023. p. 1–4. Citado na página 31.

ĐORđEVIĆ, B. et al. Comparison of file system performance in full virtualization with ms hyper-v and kvm hypervisors. In: . [s.n.], 2022. Disponível em: <https://api.semanticscholar.org/CorpusID:254587358>. Citado na página 31.

ĐORđEVIć, B. et al. Performance comparison of different hypervisor versions of the type-2 hypervisor virtualbox. In: *2023 10th International Conference on Electrical, Electronic and Computing Engineering (IcETRAN)*. [S.l.: s.n.], 2023. p. 1–6. Citado na página 31.

ĐORđEVIć, B.; KRALJEVIć, N.; DAVIDOVIć, N. Guest file system behavior for type-2 hypervisor-based virtualization in vmware workstation. In: *2023 22nd International Symposium INFOTEH-JAHORINA (INFOTEH)*. [S.l.: s.n.], 2023. p. 1–6. Citado na página 31.

ĐORđEVIć, B.; KRALJEVIć, N.; DAVIDOVIć, N. Performance comparison of cpu hardware-assisted features for the type-2 hypervisors. In: *2024 23rd International Symposium INFOTEH-JAHORINA (INFOTEH)*. [S.l.: s.n.], 2024. p. 1–5. Citado na página 31.

ĐORđEVIć, B.; KRALJEVIć, N.; DžUVEROVIć, B. Optimal guest file system for type-2 hypervisor-based virtualization in virtual box. In: *2022 30th Telecommunications Forum (TELFOR)*. [S.l.: s.n.], 2022. p. 1–4. Citado na página 31.

ĐORđEVIć, B. et al. Comparing performances of native host and virtualization on esxi hypervisor. In: *2021 20th International Symposium INFOTEH-JAHORINA (INFOTEH)*. [S.l.: s.n.], 2021. p. 1–4. Citado na página 31.

ĐORđEVIć, B. et al. Performance comparison of native host and hyper-based virtualization virtualbox. In: *2021 20th International Symposium INFOTEH-JAHORINA (INFOTEH)*. [S.l.: s.n.], 2021. p. 1–4. Citado na página 31.

Apêndices

APÊNDICE A – LISTA DE COMANDOS

Este apêndice apresenta uma coletânea de comandos utilizados ao longo do desenvolvimento deste trabalho de mestrado, oferecendo um guia prático e de fácil consulta para os leitores que desejam replicar, adaptar ou ampliar as soluções descritas.

Esse apêndice visa não apenas fornecer uma referência rápida, mas também fomentar o entendimento e a prática dos comandos, de forma a contribuir para a autonomia dos leitores na realização de atividades relacionadas à linha de comando.

Todos os experimentos realizados utilizando a suíte OpenBenchmark passaram por uma configuração personalizada para garantir maior precisão e confiabilidade nos resultados. Nos arquivos de configuração, especificamente no arquivo **test-definition.xml**, foi ajustado o parâmetro **<TimesToRun>** de seu valor padrão **3** para **30**. Essa modificação visa aumentar a repetição dos testes de 3 para 30, reduzindo a variabilidade dos resultados e permitindo uma análise estatística mais robusta.

Para o teste de desempenho da CPU, utilizamos o Blender versão 3.2.0, o teste foi realizado com a cena de referência "BMW". Esse teste foi conduzido em ambientes Windows e Linux, mantendo-se o mesmo comando para execução em ambos os sistemas operacionais, seja por meio de bash no Linux ou PowerShell no Windows.

```
# phoronix-test-suite benchmark blender-3.2.0
```

Para o teste de desempenho de memória, utilizamos o T-Test1, que é amplamente empregado para avaliar a eficiência e a estabilidade do uso de memória pelos sistemas. Assim como no teste de CPU com o Blender, o teste de memória foi conduzido tanto em sistemas Windows quanto Linux.

```
# phoronix-test-suite benchmark t-test1-1.0.1
```

O comando a seguir foi utilizado para realizar testes de leitura e escrita de disco no sistema operacional Windows, empregando a ferramenta **fio** (Flexible I/O Tester). A execução ocorreu no console PowerShell com permissões de administrador, garantindo acesso adequado aos recursos do sistema para medir o desempenho de entrada e saída de maneira precisa e confiável.

```
> for ($i = 1; $i -le 30; $i++) {
    fio --name=fiotest1 --filename=.\test1 --size=16Gb --bs=1M \
    --direct=1 --rw=read --numjobs=8 --iodepth=16 --group_reporting \
```

```
    --runtime=60 | Select-String -Pattern "READ: bw="
}
```

No ambiente Linux, o comando é quase idêntico, com a principal diferença sendo a adaptação do loop for para a sintaxe do Bash, o comando foi executar com o usuário root do Linux.

```
# for i in {1..30}; do
    fio --name=fiotest1 --filename=.\test1 --size=16Gb --bs=1M \
    --direct=1 --rw=read --numjobs=8 --iodepth=16 --group_reporting \
    --runtime=60 | grep "WRITE: bw=";
done
```

Conforme mencionado, realizamos um novo teste de CPU com o objetivo de validar os resultados obtidos com o Blender. Para isso, utilizamos o 7-Zip, cuja ferramenta de benchmark é amplamente reconhecida, proporcionando uma avaliação adicional do desempenho da CPU. O teste foi executado de maneira idêntica tanto no ambiente Windows quanto no Linux, garantindo a consistência dos parâmetros de avaliação em ambas as plataformas.

```
> for ($i = 1; $i -le 30; $i++) {
    .\7z.exe b -mmt8 -md26 | Select-String -Pattern "Tot: "
}
```

Aqui também, o comando é quase idêntico, com a principal diferença sendo a adaptação do loop for para a sintaxe do Bash, o comando também foi executado utilizando o usuário root.

```
# for i in {1..30}; do
    7z b -mmt8 -md26 | grep -i "Tot: ";
done
```

Os exemplos a seguir ilustram os ajustes realizados no ambiente, denominado "ambiente ajustado". Estes exemplos mostram as modificações feitas na CPU, memória e disco de cada máquina virtual utilizada nos testes. Essas alterações foram implementadas para otimizar o desempenho e garantir a eficácia durante a execução dos testes.

Para começar, criamos todas as máquinas virtuais copiando as configurações de CPU do host físico, nós também definimos a topologia de modo a replicar o host físico, com 1 soquete, e 4 cores com dois núcleos cada. Abaixo, temos um exemplo de configuração de CPU no arquivo XML de uma máquina virtual. Este campo especifica a quantidade de vCPUs que serão utilizadas pela VM.

```
<vcpus>8</vcpus> (antes)
<vcpus cpuset='0-7'>8</vcpus> (depois)
```

Ao configurarmos o parametro **cpuset**, estamos informando ao hipervisor para fixar essas CPUs físicas a máquina virtual.

Na configuração de mémoria, definimos todas as opções como unlimited quando passamos o paramétro "-1" no comando, dessa forma a máquina virtual não terá nenhuma limitação.

```
# virsh memtune nome_vm -1 -1 -1
```

Jás nas configurações de disco, além de utilizarmos o VirtIO, também passamos alguns parametrós de forma a ajustar e melhorar o desempenho de leitura e escrita do disco. Todos os discos virtuais foram criados com o formtato "qcow2"e com a opção "-o cluster_size=2M" para um melhor desempenho, além disso, todas as máquinas virtuais tinham o mesmo tamanho de disco, 60 GigaBytes.

```
# qemu-img create -f qcow2 -o cluster_size=2M vmdisk.qcow2 60G
```