UNIVERSIDADE FEDERAL DE CAMPINA GRANDE

CENTRO DE ENGENHARIA ELÉTRICA E INFORMÁTICA

UNIDADE ACADÊMICA DE SISTEMAS E COMPUTAÇÃO

COORDENAÇÃO DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

EDUARDO JERÔNIMO DE MOURA JUNIOR

AVALIAÇÃO DE AMBIENTES DE VIRTUALIZAÇÃO SEGUNDO CARACTERÍSTICAS DA NORMA ISO/IEC 25010 PARA AUXÍLIO NO PROCESSO DE VIRTUALIZAÇÃO DE DATACENTERS

CAMPINA GRANDE - PB

2021

Universidade Federal de Campina Grande
Centro de Engenharia Elétrica e Informática
Coordenação de Pós-Graduação em Ciência da Computação

Avaliação de Ambientes de Virtualização Segundo Características da Norma ISO/IEC 25010 para Auxílio No Processo de Virtualização de Datacenters

Eduardo Jerônimo de Moura Junior

Dissertação submetida à Coordenação do Curso de Pós-Graduação em Ciência da Computação da Universidade Federal de Campina Grande - Campus I como parte dos requisitos necessários para obtenção do grau de Mestre em Ciência da Computação.

Área de Concentração: Ciência da Computação
Linha de Pesquisa: Redes de Computadores

Reinaldo Cézar de Morais Gomes, Anderson Fabiano Batista Ferreira da Costa
(Orientadores)

Campina Grande, Paraíba, Brasil
©Eduardo Jerônimo de Moura Junior, 19/08/2021

M929a Moura Junior, Eduardo Jerônimo de.

Avaliação de ambientes de virtualização segundo características da Norma ISO/IEC 25010 para auxílio no processo de virtualização de datacenters / Eduardo Jerônimo de Moura Junior. – Campina Grande, 2021.

70 f. : il. color.

Dissertação (Mestrado em Ciência da Computação) – Universidade Federal de Campina Grande, Centro de Engenharia Elétrica e Informática, 2021.

"Orientação: Prof. Dr. Reinaldo Gomes; Coorientação: Prof. Dr. Anderson Fabiano Batista Ferreira da Costa".

Referências.

1. Virtualização. 2. Redes de Computadores. 3. Máquinas Virtuais. 4. Datacenters. I. Gomes, Reinaldo. II. Costa, Anderson Fabiano Batista Ferreira da. III. Título.

CDU 004.7(043)

FICHA CATALOGRÁFICA ELABORADA PELA BIBLIOTECÁRIA MARIA ANTONIA DE SOUSA CRB 15/398

MINISTÉRIO DA EDUCAÇÃO
UNIVERSIDADE FEDERAL DE CAMPINA GRANDE
POS-GRADUACAO CIENCIAS DA COMPUTACAO
Rua Aprigio Veloso, 882, - Bairro Universitario, Campina Grande/PB, CEP 58429-900

FOLHA DE ASSINATURA PARA TESES E DISSERTAÇÕES

EDUARDO JERÔNIMO DE MOURA JUNIOR

AVALIAÇÃO DE AMBIENTES DE VIRTUALIZAÇÃO SEGUNDO CARACTERÍSTICAS DA NORMA ISO/IEC 25010 PARA AUXÍLIO NO PROCESSO DE VIRTUALIZAÇÃO DE DATACENTERS

Dissertação apresentada ao Programa de Pós-Graduação em Ciência da Computação como pré-requisito para obtenção do título de Mestre em Ciência da Computação.

Aprovada em: 19/08/2021

Prof. Dr. REINALDO CÉZAR DE MORAIS GOMES - Orientador - UFCG

Prof. Dr. ANDERSON FABIANO BATISTA FERREIRA DA COSTA - Orientador - IFPB

Prof. Dr. THIAGO EMMANUEL PEREIRA DA CUNHA SILVA - Examinador Interno - UFCG

Prof. Dr. PAULO DITARSO MACIEL JÚNIOR - Examinador Externo - IFPB

Documento assinado eletronicamente por **Anderson Fabiano Batista Ferreira da Costa**, **Usuário Externo**, em 07/12/2021, às 13:30, conforme horário oficial de Brasília, com fundamento no art. 8º, caput, da Portaria SEI nº 002, de 25 de outubro de 2018.

Documento assinado eletronicamente por **REINALDO CEZAR DE MORAIS GOMES**, **PROFESSOR DO MAGISTERIO SUPERIOR**, em 07/12/2021, às 14:34, conforme horário oficial de Brasília, com fundamento no art. 8º, caput, da Portaria SEI nº 002, de 25 de outubro de 2018.

Documento assinado eletronicamente por **Paulo Ditarso Maciel Júnior**, **Usuário Externo**, em 07/12/2021, às 18:03, conforme horário oficial de Brasília, com fundamento no art. 8º, caput, da Portaria SEI nº 002, de 25 de outubro de 2018.

Documento assinado eletronicamente por **THIAGO EMMANUEL PEREIRA DA CUNHA SILVA**,

PROFESSOR DO MAGISTERIO SUPERIOR, em 09/12/2021, às 11:33, conforme horário oficial de Brasília, com fundamento no art. 8º, caput, da Portaria SEI nº 002, de 25 de outubro de 2018.

A autenticidade deste documento pode ser conferida no site https://sei.ufcg.edu.br/autenticidade, informando o código verificador **2000846** e o código CRC **A3840B5D**.

Referência: Processo nº 23096.049514/2021-80 SEI nº 2000846

Resumo

O uso de ambientes de virtualização está cada vez mais presente no trabalho de um analista de tecnologia da informação. Porém, com variedade de opções de ambientes e técnicas de virtualização, a tarefa de migrar de um ambiente de ativos dedicados para um ambiente virtualizado que seja adequado às necessidades do usuário se torna complexa. Esse trabalho apresenta uma avaliação de cinco ambientes de virtualização de acordo com as métricas da ISO/IEC 25010, que aborda a qualidade de software, também apresenta um esquema de sugestão de escolha de ambiente de virtualização baseado nas métricas avaliadas e na preferência do administrador de sistema, utilizando o método de tomada de decisão AHP. Adicionalmente, o trabalho propõe um sistema para recomendação de dimensionamento de máquinas virtuais, utilizando o método de tomada de decisão TOPSIS.

Abstract

The use of virtualization environments is increasingly present in the work of an information technology analyst, but with a variety of virtualization environment options and virtualization techniques, the task of migrating from an environment of dedicated assets to a virtualized environment to suit the user's needs becomes complex. This work presents an evaluation of five virtualization environments according to the metrics of ISO/IEC 25010 which addresses software quality, as well as presents a suggested scheme for choosing a virtualization environment based on the evaluated metrics and the preference of the administrator. system using the decision making method AHP, additionally, the work also proposes a system for recommending the sizing of virtual machines, using the decision making method TOPSIS.

Agradecimentos

Primeiramente gostaria de agradecer a Deus por ter proporcionado todas as condições físicas, mentais e espirituais para concluir este trabalho.

Em segundo lugar agradecer a minha família, em especial meus pais Eduardo e Lourdes por toda a orientação, carinho e atenção a mim dedicados durante toda a minha vida.

Em terceiro lugar agradeço a minha noiva Iêzha pelo apoio e incentivo durante todo o tempo que estamos juntos, pela compreensão nos momentos de falta, e por todas as mensagens de incentivo que me ajudaram a continuar seguindo em frente.

O meu muito obrigado a meu orientador Prof. Reinaldo Gomes e meu co-orientador Prof. Anderson Costa, pelo os momentos de ensinamentos e orientação durante todo o período deste trabalho.

Agradeço ao Ponto de Presença da RNP na Paraíba (PoP-PB RNP), em especial a Prof. Pedro Nicolletti (Peter) e Erick, por ter me proporcionado todo o tempo e ajuda necessários para dar continuidade no mestrado.

Ao Laboratório de Sistemas Distribuídos (LSD) e todos os professores e colegas que fazem parte do laboratório, por proporcionar a experiencia necessária no começo da minha carreira.

Obrigado aos meus colegas de mestrado, em especial Marcela,Miqueas e Yngrid, por proporcionar o companheirismo necessário para superar esse trabalho.

Agradeço também a comunidade de amigos do ”Skype do Mike”, que nesse tempo de pandemia conseguiu proporcionar uma forma descontraída de abstrair esse momento tão difícil que estamos vivendo.

E por fim gostaria de agradecer a todas as pessoas que contribuíram direta ou indiretamente para execução deste trabalho.

Sumário

Lista de Figuras

Lista de Tabelas

Capítulo 1

Introdução

1.1 Contextualização e Motivação

Com a ampla utilização de recursos de infraestrutura de redes e o aumento na demanda por aplicações móveis cada vez mais dependentes de conexão, as redes de computadores atuais já não conseguem mais prover serviços que satisfazem todas as necessidades de conexão. Esse cenário ocorre devido a vários fatores, dentre os principais: limitação de uso de equipamentos proprietários com código e protocolos fechados, constante necessidade de uso de tabelas grandes de roteamento, aumento contínuo do tráfego de rede, etc [21]. Considerando todos esses fatores, a gerência de redes atuais se torna uma tarefa muito complexa.

Adicionalmente, nos últimos anos, equipamentos dedicados a prover serviços de rede, tais como firewall, switches, roteadores, balanceadores de carga, *Network Address Translation* (NAT), tornaram-se amplamente utilizados em instituições e empresas para atender às altas demandas de seus clientes.

Os equipamentos de redes supracitados, também conhecidos por *middleboxes*, têm desempenhado um papel essencial para consolidação das redes de computadores, uma vez que com esses equipamentos foi possível chegar ao nível de conexão que temos hoje. Devemos lembrar que alguns *middleboxes* agregam várias funções de rede, como por exemplo os roteadores domésticos disponibilizados pelos *Internet Service Providers*(ISPs). Neles podemos encontrar todos os serviços necessários para uma *Lan Area Network* (LAN) doméstica com poucos usuários.

No entanto, apesar da ampla utilização de equipamentos dedicados, os mesmos apre-

sentam diversas desvantagens [34]. A primeira grande desvantagem é o seu alto custo de aquisição. Equipamentos dedicados geralmente possuem custos maiores devido a seu software proprietário e licenças que devem ser adquiridas em conjunto para atender as demandas mais específicas dos clientes. A segunda grande desvantagem é a necessidade de técnicos especializados para trabalhar com esses equipamentos dedicados, o que dificulta a operação por falta de profissionais com conhecimento adequado, o que consequentemente gera um maior custo de manutenção. O uso de equipamentos dedicados dificulta a realização de tarefas que necessitam maior flexibilidade para satisfazer novas demandas de usuários.

Analisando essas desvantagens, percebe-se que a atual arquitetura da Internet chegou ao ponto de saturação, no qual deixa a desejar na versatilidade [7]. Esse processo é chamado de ossificação da rede, onde é notável que existe uma grande complexidade para testar novos protocolos e funções no núcleo da Internet. Diante desse grande problema, várias pesquisas vêm sendo conduzidas tanto na academia quanto na indústria para buscar métodos que melhorem o processo de planejamento, implantação e manutenção das novas redes de computadores e de seus serviços [12].

A administração de uma infraestrutura de redes ou de um *data center*, pode ser uma tarefa complexa devido a suas diversas configurações de hardware e softwares necessárias para seu funcionamento [42]. Grandes provedores de serviços de internet e nuvem necessitam constantemente de testar novas tecnologias, novos protocolos ou corrigir falhas em seus serviços o que muitas vezes geram *downtime* para seus clientes devido à utilização de *middleboxes* e hardwares dedicados em sua infraestrutura. Esses hardwares dedicados, apesar de apresentarem características robustas para provimento de seus respectivos serviços, na sua grande maioria não são de fácil substituição, o que intensifica a possibilidade de *downtime* nos serviços providos e a complexidade de solução de problemas.

Nesse contexto, a fim de mitigar estes problemas, algumas abordagens foram consideradas [46], nas quais aquelas que usam técnicas de virtualização se destacaram mais [19]. Com o advento da virtualização, foi possível que várias aplicações executassem sobre o mesmo domínio físico compartilhado, resultando numa melhor utilização de recursos computacionais. Logo, ativos de redes como roteadores e switches podem ter suas funções virtualizadas e implementadas em software, eliminando o uso de hardware especializado.

Verificamos, então, a necessidade de investigação de uma solução para evitar a

ossificação da rede, utilizando novos métodos para aumentar a flexibilidade das redes e de seus serviços, ao ponto que o administrador de sistemas consiga realizar sua atividade de forma mais eficiente, economizando recursos e provendo melhores soluções para as suas demandas diárias.

1.2 Delimitação do Problema da Pesquisa

Considerando o uso de virtualização, é importante a escolha do hipervisor a ser utilizado, sabendo das suas vantagens e desvantagens dependendo da necessidade do administrador do sistema [34]. Também é de grande importância a decisão do dimensionamento da máquina virtual, para que os recursos sejam utilizados de forma precisa, evitando subutilização e superutilização [23].

A escolha de hipervisor e o dimensionamento de um ambiente virtualizado são tarefas complexas que o administrador de sistemas deve realizar. Essas decisões devem considerar tanto aspectos objetivos como aspectos subjetivos de cada hypervisor existente, bem como avaliar a quantidade de recursos necessária para que o ambiente virtualizado seja capaz de executar as demandas dos serviços de rede. Dessa forma, uma escolha de ambiente virtualizado sem um estudo prévio pode comprometer o sucesso da migração de um ambiente dedicado e também gerar custos adicionais para o administrador do sistema [22].

Na maioria dos casos, os administradores de sistema escolhem hipervisores com base na sua experiência, sem considerar aspectos objetivos como desempenho ou aspectos subjetivos como operabilidade do hipervisor [19]. Nesse sentido, o administrador de sistema poderia obter melhores resultados se realizasse um estudo de comparação entre hipervisores, ou ainda elabora-se um mecanismo que sugere qual hipervisor utilizar considerando as aplicações e preferências do administrador [32]. Do mesmo modo, o dimensionamento de recursos para uma máquina virtual também é, na maioria dos casos, escolhida através da experiência do administrador, o que pode ocasionar degradação no desempenho da aplicação de rede ou desperdícios de recursos computacionais.

Neste trabalho, vamos propor um processo de avaliação em conjunto com um processo de decisão, que auxilie no levantamento de métricas e requisitos para que a transição entre ambientes não apresente nenhum ônus em relação à performance da rede, se comparada a

seu estado anterior usando equipamentos dedicados. Esse processo receberá dados da atual infraestrutura fornecida pelo administrador de sistema e apresentará escolhas de hipervisor e *flavor* de acordo com a sua necessidade.

1.3 Objetivos da Pesquisa

1.3.1 Objetivos Gerais

O presente trabalho tem por objetivo propor um processo de decisão para auxiliar o administrador de sistemas na transição de uma infraestrutura de redes com equipamentos dedicados para um ambiente virtualizado, com base nas métricas de qualidade de software propostas na ISO 25010 [1].

1.3.2 Objetivos Específicos

Dentro desse contexto, definimos os seguintes objetivos específicos:

- **Realizar testes de desempenho com diferentes hipervisores:** Utilizar softwares de *benchmarks* para aferir o desempenho dos hipervisores considerados neste trabalho.
- **Realizar testes de desempenho com diferentes configurações de máquinas *flavors* e serviços de rede:** Utilizar softwares de *benchmark* para aferir o desempenho de serviços de rede com diferentes *flavors*.
- **Utilizar um método de tomada de decisão para escolha de hipervisor:** Utilizar um método de tomada de decisão para sugerir um hipervisor utilizando preferências do administrador de sistema de acordo com as métricas da ISO 25010.
- **Implementar um processo de sugestão de migração de infraestrutura:** Implementar um processo de sugestão de infraestrutura, usando como base os testes de benchmark feitos previamente no trabalho.
- **Validar processo de sugestão de infraestrutura:** Validar o processo de sugestão para migração de infraestrutura, comparando os resultados obtidos no ambiente virtualizado com o ambiente dedicado.

1.4 Contribuições

A seguir listamos as principais contribuições deste trabalho:

- Auxílio na migração de um ambiente de redes dedicado para um ambiente virtualizado;
- Compreensão mais aprofundada sobre características de diferentes hipervisores;
- Conhecimento sobre utilização de recursos computacionais em diferentes serviços de rede;
- Auxílio no processo de decisão para escolha de hipervisor;
- Auxílio no processo de decisão para escolha de dimensão de máquinas virtuais.

1.5 Organização da Dissertação

Os demais capítulos desse trabalho estão organizados da seguinte forma:

Capítulo 2 - Neste capítulo são apresentados o referencial teórico que serviu como base da pesquisa, e os conceitos necessários para entendimento do trabalho.

Capítulo 3 - Neste Capítulo são apresentados trabalhos relacionados com a pesquisa efetuada neste trabalho, exaltando as semelhanças e diferenças entre eles.

Capítulo 4 - No Capítulo 4 são apresentados o procedimento de trabalho, a abordagem que foi realizada, além das métricas e mecanismos de decisão utilizados.

Capítulo 5 - Neste Capítulo é apresentada a validação do trabalho utilizando um cenário de simulação, onde foi possível colocar em prática a abordagem proposta.

Capítulo 6 - No Capítulo 6 são apresentados a conclusão do trabalho e as considerações finais, bem como os trabalhos futuros.

Capítulo 2

Fundamentação Teórica

2.1 Virtualização

A virtualização é uma técnica bastante utilizada no mercado atualmente, que de abstrai o hardware para um determinado processo em um sistema operacional e divide logicamente a capacidade computacional em unidades menores independentes umas das outras [24].Essa técnica teve origem no final dos anos 60, desenvolvida pela IBM como uma forma de segregar os recursos computacionais dos mainframes da época para executar aplicações distintas. Após a década de 70, os computadores pessoais com arquitetura x86 se tornaram mais comuns e a técnica de virtualização caiu em desuso [16][40].

Os *data centers* foram criados para agregar conjuntos de recursos de infraestrutura de tecnologia da informação em uma estrutura complexa, abrigando diversos tipos de equipamentos, como servidores, ativos de rede e sistemas de segurança [42]. Na década de 50 e 60, os *data centers* eram compostos com *mainframes* dedicados e sistemas centralizados. Após a popularização do computador pessoal, ocorreu uma mudança nos *data centers* e os sistemas passaram a ser híbridos. Parte dos sistemas continuavam centralizados e começou a surgir aplicações distribuídas e descentralizadas. Na terceira geração de *data centers*, que é a geração atual, todos os sistemas e aplicações passaram a ser virtualizados [42].

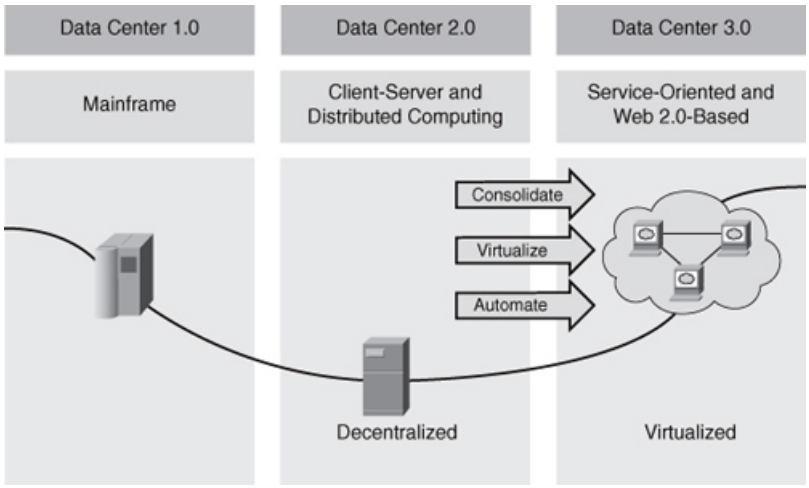

Figura 2.1: Evolução dos data centers.

Fonte [31]

Com a crescente demanda por *data centers* mais robustos com melhor utilização do hardware e maior isolamento de aplicações, a virtualização retornou forte para o mercado, alterando a forma como grandes serviços de provedores de conteúdo na Internet trabalham com recursos computacionais, almejando obter mais segurança, melhor uso de recursos e menores custos para manter a infraestrutura.

2.1.1 Definição

A virtualização de hardware nos permite executar variadas aplicações e sistemas operacionais simultaneamente. Esses sistemas e aplicações executam de forma independente em cima do mesmo hardware compartilhado, proporcionando maior flexibilidade e isolamento [7][11]. O servidor ou máquina física que abriga as máquinas virtuais é chamada de Hospedeiro, e as máquinas virtuais que executam sobre o hospedeiro são chamadas de convidados. A virtualização acontece utilizando uma camada de software entre o hardware do hospedeiro e o seu sistema operacional, disponibilizando um ambiente completo e análogo ao que existe na máquina física [34]. Essa camada de software é conhecida através do acrônimo VMM (*Virtual Machine Monitor*) ou Hipervisor [6][38][33], e é o software responsável por emular o hardware virtual para os sistemas convidados.

Por causa da virtualização temos o que hoje é conhecido como computação em nuvem, onde a flexibilidade da virtualização permite que recursos como processamento, memória,

rede e armazenamento sejam distribuídos por demanda para o usuário, provendo agilidade na entrega dos recurso,s bem como flexibilidade e custo baixo. Os administradores de nuvem podem escalar recursos de forma dinâmica graças a virtualização [38], minimizando a ociosidade de recursos de acordo com as demandas existentes. A virtualização possibilita uma execução de uma máquina virtual completa, com sistemas operacionais diferentes do sistema do hipervisor, o que torna a técnica ainda mais flexível para o uso.Para isso, temos varias opções de hipervisores no mercado, destacando os mais utilizados: KVM, Xen, VMware.

2.1.2 Tipos de Virtualização

A virtualização utiliza os hipervisores como camada intermediaria entre os Hospedeiros e as máquinas virtuais (convidados), mas a forma como é implementada se difere em quatro tipos distintos [38][6], como mostrados a seguir.

Virtualização Completa ou *Hosted Virtualization*

Na virtualização completa apresentada na figura 2.2, o hipervisor atua como um processo no sistema operacional do hospedeiro, não necessitando de modificações e nem de configurações especiais no sistema. O hardware é totalmente emulado e o sistema operacional do hospedeiro gerencia as VMs através de um software instalado em seu sistema que atua como hipervisor. Dessa forma, o sistema operacional da máquina virtual consegue ser executado de forma isolada, sem interferir no sistema operacional do hospedeiro. A principal vantagem desse tipo de virtualização consiste no fato da simplicidade na instalação e utilização do hipervisor, sem a necessidade de modificações. Exemplos de hipervisores que usam esse tipo de virtualização são o Virtual Box e o Vmware Player.

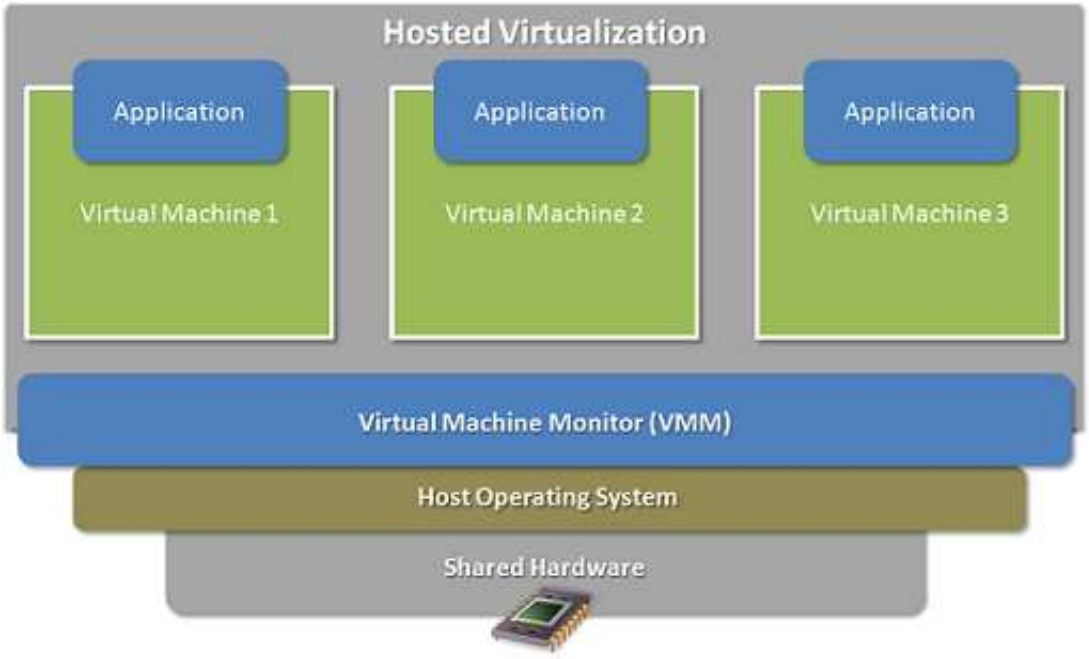

Figura 2.2: Virtualização Completa.

Para-Virtualização ou *Bare-Metal Virtualization*

No cenário da para-virtualização mostrado na figura 2.3, o sistema operacional do hospedeiro é modificado de tal forma que ele trabalha como hipervisor. Ou seja, o hipervisor, que está no mesmo nível do sistema operacional do hospedeiro, é carregado durante o processo de boot do servidor. Esse tipo de hypervisor é capaz de obter acesso direto ao hardware diferente da virtualização completa, e tratar as solicitações de acesso a recursos de hardware do sistema hospedeiro, além de manipular instruções protegidas ou privilegiadas [47]. Essas modificações servem para incluir o hipervisor no sistema hospedeiro e tem como objetivo prover maior desempenho para os sistemas convidados, uma vez que toda as requisições de acesso ao hardware conseguem se comunicar diretamente com o hipervisor. Entretanto, existe a desvantagem do sistema convidado precisar ser modificado para conseguir trabalhar com os drivers do hipervisor do sistema hospedeiro. Exemplo de hipervisor que utiliza para-virtualização é o XEN e o Vmware Enterprise.

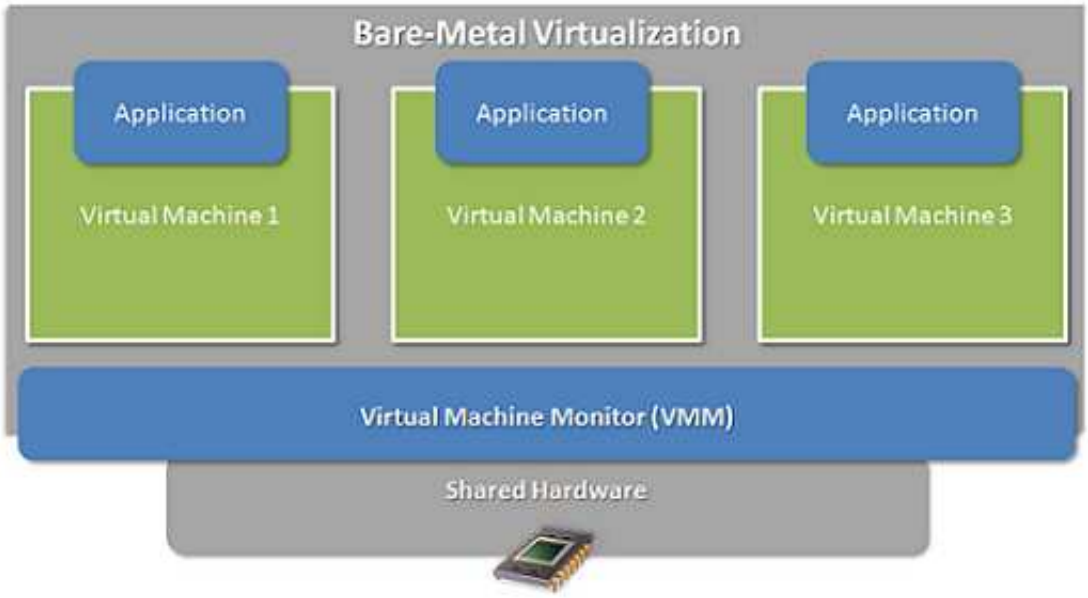

Figura 2.3: Para-Virtualização.

Virtualização Assistida por Hardware

No caso da virtualização assistida por hardware mostrado na figura 2.4, os fabricantes de hardware começaram a projetar seus recursos computacionais para melhorar o desempenho dos seus recursos em ambientes virtualizados onde os maiores exemplos são as tecnologias desenvolvidas pela Intel (Intel VT) e AMD (AMD-V). Esses modelos de chips de processamento foram desenvolvidos com novos modos de execução, permitindo que determinadas chamadas de CPU realizadas pelos sistemas convidados sejam executadas diretamente na CPU, sem a necessidade de passar pelo hipervisor. Desse modo, é possível observar um maior desempenho em aplicações que necessitam de maior poder de processamento em máquinas virtuais que trabalham com essa tecnologia. Porém, essa tecnologia é totalmente dependente do suporte de hardware.

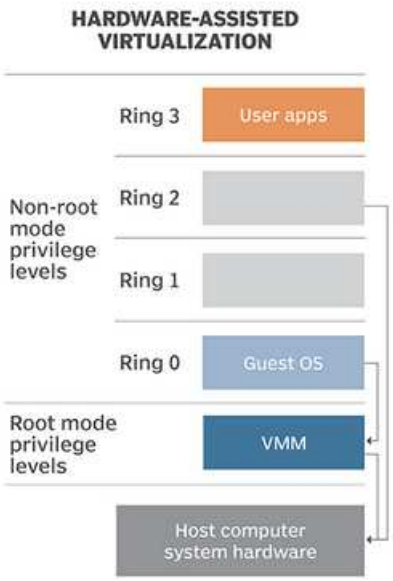

Figura 2.4: Virtualização Assistida por Hardware.

Fonte

https://searchservervirtualization.techtarget.com/definition/hardware-assisted-virtualization

Virtualização no Nível de Kernel

Por fim, temos o tipo de virtualização no nível de kernel, que utiliza o próprio kernel do linux para executar as máquinas virtuais como um processo do sistema operacional comum do usuário [29]. Entretanto, é necessário que o hospedeiro carregue algumas bibliotecas e utilitários no sistema, e que o sistema convidado possua a compatibilidade com o conjunto de instruções da máquina hospedeira. Exemplo de hipervisor que utiliza esse tipo de virtualização é o *Kernel-based Virtual Machine* ou KVM.

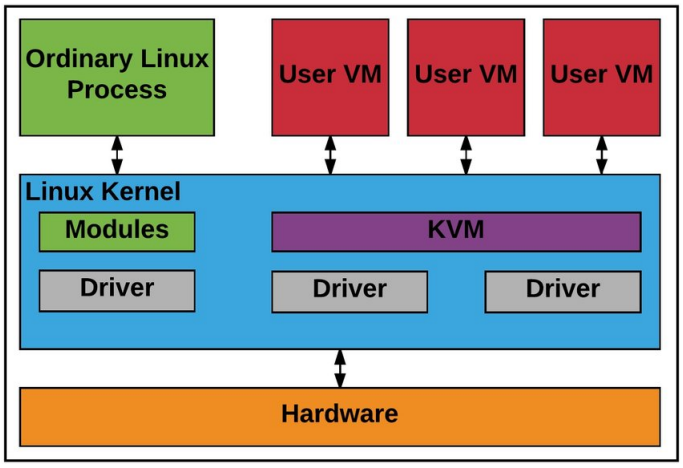

Figura 2.5: Virtualização no nível do Kernel

2.1.3 Benefícios da Virtualização

Dentre inúmeros benefícios provenientes do uso de virtualização, podemos destacar alguns mais significativos.

- Migração de máquinas virtuais: com a virtualização é possível migrar máquinas virtuais entre diferentes hardwares, facilitando a gerencia de recursos e a manutenção da infraestrutura como um todo [14].
- Isolamento lógico: com a virtualização existe o isolamento entre os sistemas operacionais do convidado e do hospedeiro, e também entre os sistemas convidados.
- Rápido provisionamento: a virtualização permite um rápido provisionamento de servidores virtualizados, devido a flexibilidade existente em tais ambientes.
- Redução de *downtime* e resiliência em desastres: devido ao rápido provisionamento e a migração de máquinas virtuais, existe uma redução significativa do *downtime* gerado por desastres.
- Aproveitamento de recursos: com a virtualização, o uso de recursos se torna mais eficiente, minimizando subutilização de servidores físicos.

2.2 *Containers*

O processo de conteinerização é conhecido como a virtualização a nível de sistema operacional, que utiliza pouquíssimos recursos da máquina física que o hospeda, proporciona isolamento e é gerenciada através de bibliotecas e o kernel do sistema operacional do hospedeiro. Essas instâncias chamadas de *container* agregam todo um ambiente de execução de uma aplicação: a aplicação em si, suas dependências, bibliotecas e arquivos de configuração, tudo isso agrupado em um único pacote.

Dessa forma, podemos dizer que os containers virtualizam o sistema operacional enquanto que a virtualização de máquinas abstrai tanto o sistema operacional quanto o hardware. Esse aspecto dos containers possibilita pouca sobrecarga e menor tempo de provisionamento se comparado à virtualização com máquinas virtuais.

A principal vantagem de utilizar containers é o fato de não ser necessário existir um sistema operacional e um hardware virtualizado [20] para executar uma aplicação. Sendo assim, é obtida uma grande otimização no uso dos recursos computacionais se comparado aos modelos tradicionais de virtualização tipo *hosted* e tipo *bare-metal*.

Máquinas virtuais e os *containers* possuem funções muito similares considerando o isolamento [13] e alocação de recursos computacionais. Todavia , a abordagem da arquitetura é diferente e uma comparação entre as duas arquiteturas é apresentada na Figura 2.6

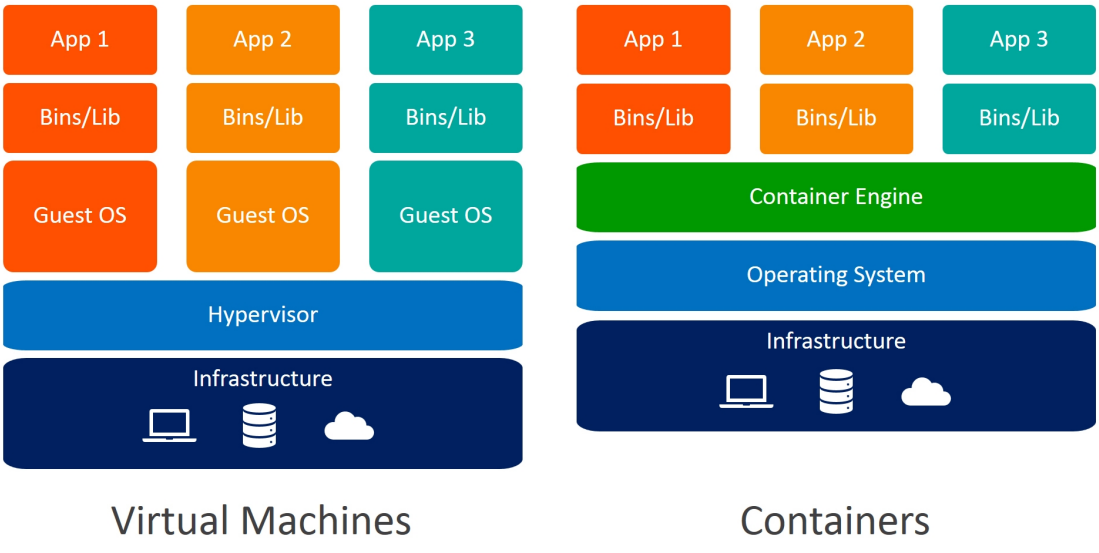

Figura 2.6: Comparação entre máquinas virtuais e *containers*.

Apesar de grande utilização entre desenvolvedores, sempre existem preocupações sobre segurança e privacidade para a execução de diversos *containers*, pertencentes a inquilinos diferentes em um mesmo *kernel* de sistema operacional do hospedeiro . Essa característica dos *containers* pode não garantir segurança caso o *kernel* do sistema hospedeiro apresente algum tipo de vulnerabilidade, então no quesito segurança podemos assumir que o *container* possui um nível menor de isolamento se comparado a uma máquina virtual.

2.3 ISO/IEC 25010 - Padrão de Qualidade de Software

A ISO/IEC 25010 é um padrão para mensurar qualidade de software proposto pelas organizações *International Organization Standardization*(ISO) e *International Electrotechnical Comission* (IEC) [2]. O modelo proposto pela ISO/IEC é base de um sistema de avaliação da qualidade de um produto de software, que determina quais características podem ser consideradas para avaliar as propriedades de um software [1].

A qualidade de um sistema é o grau em que satisfaz as necessidades declaradas e implícitas das partes interessadas no mesmo, fornecendo valor ao produto [30]. As necessidades das partes interessadas no sistema são as variáveis representadas no modelo, que categoriza a qualidade do produto em oito grandes características, sendo elas:

- Funcionalidade: Grau no qual o produto ou sistema provê funções que atendem as necessidades das partes interessadas.
- Confiabilidade: Grau no qual o produto ou sistema desempenha funções sob condições específicas em um determinado período de tempo.
- Usabilidade: Grau no qual o produto ou sistema pode ser usado por usuários específicos para alcançar determinado objetivo com eficiência e satisfação.
- Eficiência: Essa característica representa o desempenho relativo à quantidade de recursos usados nas condições estabelecidas.
- Manutenibilidade: Esta característica representa o grau de eficácia e eficiência com que um produto ou sistema pode ser modificado para melhorá-lo, corrigí-lo ou adaptá-lo às mudanças no ambiente e nos requisitos.

- Portabilidade: Grau de eficácia e eficiência com que um sistema, produto ou componente pode ser transferido de um hardware, software ou de qualquer ambiente operacional para outro.

- Segurança: O grau em que um produto ou sistema protege informações e dados para que pessoas ou outros sistemas tenham o grau de acesso aos dados adequado aos seus tipos e níveis de autorização.

- Compatibilidade: O grau em que um produto, sistema ou componente pode trocar informações com outros produtos, sistemas ou componentes, e/ou executar suas funções necessárias enquanto compartilha o mesmo ambiente de hardware ou software.

As oito características enumeradas acima são compostas por diversas sub-características, dentre elas destacamos as utilizadas neste trabalho para medir o nível de qualidade dos hipervisores:

- Operabilidade: Grau em que um produto ou sistema possui atributos que o torna mais fácil de operar e controlar.

- Proteção contra erros do usuário: O grau em que um sistema protege os usuários para não cometer erros.

- Interoperabilidade: O grau em que dois ou mais sistemas, produtos ou componentes podem trocar informações e usar as informações que foram trocadas.

- Recuperabilidade: Grau em que, em caso de interrupção ou falha, um produto ou sistema pode recuperar os dados diretamente afetados e restabelecer o estado desejado do sistema.

2.4 Mecanismos de Tomada de Decisão

O processo de tomada de decisão significa realizar uma tarefa que faz com que aconteça ou não uma evolução em um processo. Portanto, a decisão pode interferir positivamente ou

negativamente em um fluxo de rotina de uma empresa, de um setor ou até na vida pessoal [15].

Considerando o contexto da decisão, há a necessidade de avaliação do ato de decidir, pois as consequências irão refletir no sucesso da escolha. O processo de decisão implica em coleta e avaliação das informações com o objetivo de buscar alternativas para a solução. Todos os dias no nosso cotidiano nos deparamos com decisões, avaliamos nossos atos baseados em conhecimentos prévios, experiência ou coleta de dados [17] [26].

A tomada de decisão é o estudo da identificação e escolha de alternativas para encontrar a melhor solução com base em diversos fatores, considerando as expectativas dos avaliadores. Cada tomada de decisão é efetuada dentro de um ambiente de decisão, que é definida como a agregação da informação, alternativas, valores e preferências disponíveis no momento da tomada de decisão [26][25].

Segundo Campos [15], no processo de decisão, é necessário avaliar quais quesitos são mais importantes e indispensável para satisfação do resultado final. Em um processo decisório empresarial, as atitudes tomadas podem se tornar uma tarefa complexa. Diferente das decisões de carácter pessoal, decisões empresariais geralmente envolvem muitas variáveis que têm a possibilidade de afetar diversos aspectos da empresa, como também das pessoas e entidades envolvidas.

A parte complexa na tomada de decisão é a multiplicidade dos critérios estabelecidos para julgar as alternativas. O ser humano, devido à sua natureza, tem limitações de absorção das possibilidades existentes quando a decisão envolve múltiplos critérios. Sem um procedimento bem detalhado, a tomada de decisão se torna um processo de difícil entendimento, devido à quantidade de informações a serem consideradas [27].

Os objetivos são frequentemente conflitantes e na maioria dos casos, existem diferentes grupos de tomadores de decisão envolvidos no processo. Com o objetivo de facilitar essa análise, surgiu o MCDA (*Multicriteria Decision Analysis*), dada a necessidade de existir um método formal para auxiliar a tomada de decisão em situações que envolvem múltiplos critérios [8].

o MCDA é um estudo de uma classe geral de modelos de pesquisa operacional, no qual é adequado para abordar problemas complexos que têm como caraterísticas a alta incerteza, objetivos conflitantes, diferentes formas de dados e informações, multi interesse e perspecti-

vas [17] [8].

Dentre as vantagens da aplicação de métodos MCDA, destacam-se: a análise do contexto da tomada de decisão, com a identificação de soluções alternativas e consequências; a organização do processo de decisão, a fim de obter coerência entre objetivo da decisão e a decisão final alcançada; a cooperação e o consenso entre os atores da tomada de decisão; a legitimação da decisão final [10].

Entre os vários métodos existentes de MCDA, os principais podem ser citados:

- AHP - *Analytic Hierarchy Process*;
- ANP - *Analytic Network Process*;
- EVAMIX - *Multi-criteria evaluation with mixed qualitative-quantitative data*;
- TOPSIS - *Technique for Order Preference by Similarity to Ideal Solution*;
- MACBETH - *Measuring Attractiveness by a Categorical Based Evaluation Technique*;
- MAUT - *Multiple Atribute Utility Theory*;
- PROMETHEE - *Preference Ranking Method for Enrichment Evaluation*.

Neste trabalho, foram escolhidos os métodos AHP e TOPSIS para auxiliar no processo de decisão de hipervisores e *flavors*, o AHP foi escolhido por sua capacidade de trabalhar com valores subjetivos de maneira mais eficiente, e o TOPSIS por ser um método mais objetivo e prático para tomada de decisão com valores quantitativos. Nas subseções seguintes, discorremos sobre esses dois métodos escolhidos no trabalho.

2.4.1 AHP

O método AHP [41] foi desenvolvido por Thomas Lorie Saaty em 1970. Esse método se baseia na decomposição e síntese das relações entre os critérios até que se chegue a uma priorização dos seus indicadores, aproximando-se de uma melhor resposta de medição única de desempenho [39]. A ideia principal da teoria da análise hierárquica é a redução do estudo de sistemas a uma sequência de comparações por pares.

A definição do AHP de acordo com o *Decision Suport System Glossary* é ”uma aproximação para tomada de decisão que envolve estruturação de multicritério de escolha numa hierarquia. O método avalia a importância relativa dos critérios, compara alternativas para cada critério e determina um ranking total das alternativas”[45].

De acordo com Saaty e Vargas [41], o principal benefício do método AHP é o fato de que, como os valores dos julgamentos das comparações paritárias são baseados em intuição, experiência e também em dados coletados, o AHP consegue lidar com aspectos qualitativos e quantitativos de um problema de decisão. Sendo assim, o AHP inclui e mede todos os fatores importantes para o tomador de decisão, qualitativamente ou quantitativamente mensuráveis, tangíveis ou intangíveis, para se aproximar de um modelo realista. A Figura 2.7 ilustra exemplo da estrutura hierárquica do AHP.

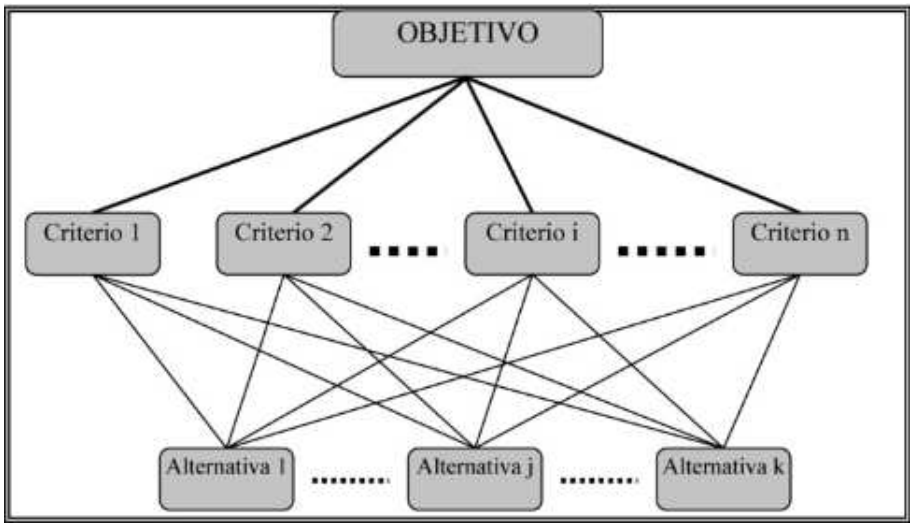

Figura 2.7: Estrutura Hierárquica do AHP.

Para representar a importância de um elemento em relação a outro, uma escala de avaliação foi criada por Saaty e ilustrada na Figura 2.8. Essa tabela apresenta uma escala de valores de 1 a 9 para comparação por pares de variáveis e definição de importância de uma em relação a outra. Para garantir que a decisão seja realizada com consistência, o método AHP dispõe do recurso que calcula as denominadas Razões de Consistência (RC) e Índice de Consistência (IC). Para garantir um nível aceitável de decisão, a razão de consistência deve ficar abaixo dos 20%, considerada a tolerância máxima de inconsistência de valores.

VALOR	DEFINIÇÃO	EXPLICAÇÃO
1	Igual importância	os dois critérios contribuem de forma idêntica para o objetivo
3	Pouco mais importante	a análise e a experiência mostram que um critério é um pouco mais importante que o outro
5	Muito mais importante	a análise e a experiência mostram que o critério é claramente mais importante que o outro
7	Bastante mais importante	a análise e a experiência mostram que um dos critérios é predominante para o objetivo
9	Extremamente mais importante	sem qualquer dúvida um dos critérios é absolutamente predominante para o objetivo
2,4,6,8 valores recíprocos dos anteriores	Valores intermediários	também podem ser utilizados

Figura 2.8: Escala Fundamental de Saaty

Então, de acordo com Saaty [41], o processo do AHP consiste em 3 partes:

- Construir Hierarquias: Nessa etapa é construída uma hierarquia com os objetivo da tomada de decisão, os critérios a serem avaliados e no fim as possíveis alternativas e valores de recomendação.
- Estabelecer prioridades entre critérios: Nessa etapa o tomador de decisão compara par a par os critérios atribuindo valores da escala de Saaty.
- Verificar consistência na comparação dos pares: Após calcular o nível de recomendação de cada alternativa, é necessário checar a razão de consistência se está em um valor aceitável.

2.4.2 TOPSIS

O método TOPSIS foi desenvolvido por Huang e Yoon [44] na década de 1970. Esse método é outra técnica MCDA que se destaca por ser intuitivo e pela simplicidade dos procedimentos matemáticos, facilidade de implementação e aplicação, e permite avaliar uma quantidade não limitada de variáveis quantitativas [18]. Para esse método, a melhor alternativa é aquela que está mais próxima da solução ideal positiva e mais distante da solução ideal negativa [28].

A solução ideal positiva é aquela que maximiza os critérios ”positivos e minimiza os critérios negativos”. O inverso ocorre para buscar a solução ideal negativa, ou seja, maximiza os critérios negativos e minimiza os critérios positivos de acordo com a Figura 2.9.

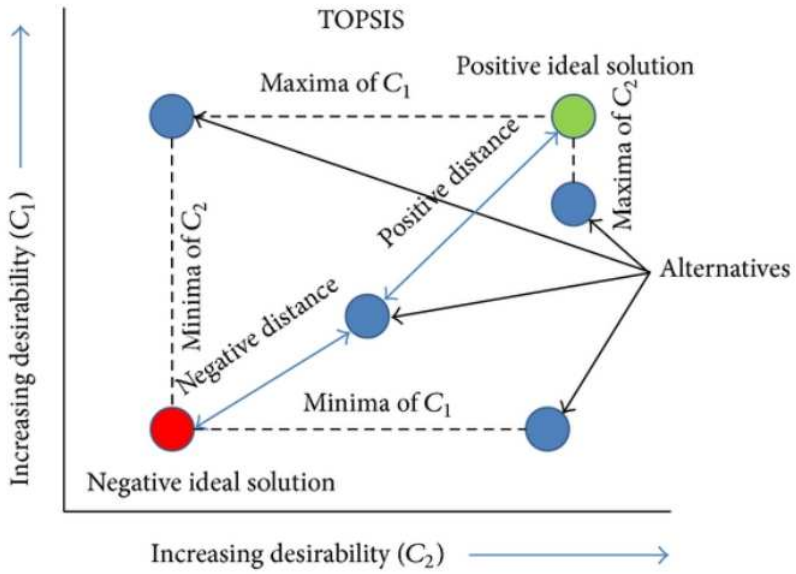

Figura 2.9: Funcionamento do TOPSIS.

Na Figura 2.9 observamos graficamente como se comporta a solução TOPSIS, os marcadores azuis são as alternativas, e os marcadores verde e vermelho são as soluções positiva e negativa ideais respectivamente. Quanto mais uma alternativa se aproxima das soluções ideais, mais essa alternativa vai ser recomendada pelo processo TOPSIS.

A aplicação do método é apresentada em forma de etapas sucessivas, que podem ser calculadas em qualquer ambiente de programação observando a fundamentação matemática do processo. A seguir representamos as etapas do método TOPSIS.

1^a Etapa: Montar a matriz de decisão.

O algoritmo do TOPSIS busca elaborar uma matriz de decisão ilustrada na Figura 2.10 contendo as alternativas viáveis representadas nas linhas da matriz (A1, A2, Ai, ..., Am), e os critérios (C1, C2, Ci, ..., Cm) e opcionalmente, pesos para os critérios W (W1, W2, ..., Wm).

$$M = \begin{array}{c} \\ A_1 \\ \vdots \\ A_i \\ \vdots \\ A_1 \end{array} \begin{array}{c} \begin{array}{ccccc} C_1 & C_2 & .. & C_j & .. & C_m \end{array} \\ \begin{bmatrix} m_{11} & m_{12} & .. & m_{1j} & .. & m_{1m} \\ \vdots & \vdots & & \vdots & & \vdots \\ m_{i1} & m_{i2} & .. & m_{ij} & .. & m_{im} \\ \vdots & \vdots & & \vdots & & \vdots \\ m_{n1} & m_{n2} & .. & m_{nj} & .. & m_{nm} \end{bmatrix} \end{array}$$

$$\widetilde{W} = [\widetilde{w}_1, \widetilde{w}_2, \dots, \widetilde{w}_m]$$

Figura 2.10: Matriz de Decisão do método TOPSIS

2^a Etapa: Calcular a Matriz Normalizada.

A matriz de decisão precisa ser normalizada e o método permite que você escolha qual tipo de normalização. Na maior parte dos casos é utilizada a normalização linear.

$$r_ij = \frac{x_ij}{\sqrt{\sum x_{ij}^2}}$$

Onde ij representa o escore do j-ésimo critério para i-ésima fonte de dados.

3^a Etapa: Calcular matriz com valores de pesos.

Multiplica-se a matriz normalizada pelos respectivos pesos dos critérios. A definição do peso no TOPSIS acontece de acordo com a preferência do decisor ou de um grupo de decisores.

$$v_{ij} = w_{ij} r_{ij}$$

Pnde w_{ij} é o peso definido para cada critério.

4^a Etapa: Identificação da solução positiva ideal (*Positive Ideal Solution*) e da solução negativa ideal (*Negative Ideal Solution*).

Nessa etapa, são determinados os pontos de referência ideal(S+) e anti-ideal(S-) para cada critério analisado no trabalho.

$$S^+ = \{(max v_{ij} | j \in J), (min v_{ij} | j \in J')\}$$

$$S^- = \{(minv_{ij}|j \in J), (maxv_{ij}|j \in J')\}$$

Onde J e J' representam os conjuntos de critérios.

5^a Etapa: Calcular as distâncias entre a situação ideal positiva e cada alternativa (D+), além da situação ideal negativa e cada alternativa (D-).

Esta etapa se resume a calcular a distância entre cada alternativa e a solução ideal e anti-ideal. As distâncias euclidianas entre cada alternativa ideal (D+) e anti-ideal (D-) são calculadas da seguinte forma:

$$D_i^+ = \sqrt{\sum_{j=1}^{n} \left(v_{ij}(x) - v_j^+(x)\right)^2}$$

$$D_i^- = \sqrt{\sum_{j=1}^{n} \left(v_{ij}(x) - v_j^-(x)\right)^2}$$

6^a Etapa: Calcular a similaridade para a posição ideal positiva.

Por fim, obtemos o coeficiente C ou resultado da aproximação da situação ideal (Ci) e a definição da ordenação das alternativas, através da equação:

$$C_i = \frac{D_i^-}{D_i^+ + D_i^-}$$

As alternativas são ranqueadas em ordem decrescente [9], de acordo com os valores obtidos do coeficiente de aproximação que vai de 0 a 1. As alternativas que mais se aproximam do valor 1 são as melhores de acordo com o método TOPSIS.

2.5 Trabalhos Relacionados

Esta seção apresenta um levantamento de trabalhos relacionados, detalhando trabalhos sobre utilização de métricas de qualidade de software com a norma ISO 25010 e trabalhos com avaliações entre diferentes ambientes virtualizados. Na seção 2.3, um sumário de como esse trabalho utiliza os conhecimentos adquiridos e como se diferencia dos trabalhos existentes.

2.5.1 Avaliação de Qualidade de Software ISO/IEC 25010

Existem muitas técnicas disponíveis para medir qualidade de software, mas a norma ISO/IEC [30] do ano de 2011 define métricas para quantificar diversos parâmetros de um software.

Para diferentes tipos de software, existem métodos distintos de medir a qualidade do software. Contudo, não é comum achar práticas de medir qualidade dos softwares que gerenciam infraestrutura de TI. O trabalho de Karnouskos [31], propõe implementar métricas de qualidade de TI em softwares de controle industrial, chamados de agentes industriais. Para realizar essa análise, o autor usa como base as métricas da norma ISO 25010 para atribuir conceitos às diversas características do software.

Já no trabalho de Jung Ji Young et al. [49], é proposto uma análise da possibilidade de migração de uma intranet em servidores locais dedicados para um serviço de nuvem pública. Esse estudo compara as duas abordagens, servidores locais e serviços em nuvem utilizando métricas de qualidade de software da ISO 25010. O objetivo desse trabalho é realizar um estudo de viabilização de transferência da infraestrutura de TI local para nuvem. A utilização das métricas propostas na ISO 25010 como critério de comparação entre os dois ambientes auxilia no entendimento das dificuldades de segurança de uma possível migração ou de um ambiente misto entre nuvem pública e servidores locais.

2.5.2 Avaliação de diferentes ambientes de virtualização

A maior parte dos trabalhos relacionados com análise de ambientes de virtualização é focada no desempenho de cada ambiente. No trabalho de Dejana et al. [46], foram comparados dois ambientes que trabalham com virtualização do tipo completa, Foi executado *benchmarks* em ambos os ambientes de virtualização e foi concluído que o VirtualBox apresentou resultados levemente melhores ao Vmware Player.

No trabalho de Borislav et al. [19], foi feita uma comparação mais ampla entre diversos hipervisores para mensurar a performance de diferentes sistemas de arquivos usando diferentes aplicações conhecidas como aplicações web, e-mail, servidor de arquivos etc. Esse trabalho executou *benchmarks* nos diferentes tipos de cenários utilizando cada hipervisor de tipos diferentes e chegou a conclusão que nenhum hipervisor consegue obter o melhor desempenho em todas as aplicações e *benchmarks* testados no trabalho. Para cada aplicação que utiliza determinado sistema de arquivo, existe um hipervisor que desempenha melhor em relação aos outros.

Em outro trabalho de comparação feito por Kumar et al.[32], foi desenvolvida uma comparação entre os dois hipervisores *open-source* mais utilizados no mercado, KVM e

XEN. No estudo foi comparado varias métricas como desempenho, flexibilidade, custo, segurança e manutenibilidade, Foi constatado que a camada de virtualização afeta fortemente o desempenho dos hipervisores. Após realizarem os testes e observarem as métricas consideradas no trabalho, os autores puderam confirmar que em termos de desempenho, os testes mostraram leve vantagem para o XEN. Entretanto, no restante das métricas avaliadas o KVM conseguiu se destacar positivamente em relação ao XEN.

2.5.3 Considerações Finais

Este capítulo apresentou um levantamento de trabalhos que visam analisar a qualidade de software usando a norma ISO 25010 e trabalhos que propõem uma comparação de ambientes de virtualização de acordo com diferentes métricas, em especial métricas de desempenho. A proposta deste trabalho se difere dos trabalhos analisados neste capítulo, pois toma como base comparar diferentes ambientes de virtualização utilizando a norma padrão ISO 25010 reconhecida internacionalmente. Com os resultados dessa comparação busca-se executar um método de tomada de decisão para auxiliar o decisor na escolha do hipervisor adequado para suas necessidades, bem como auxiliar na decisão de dimensão da máquina virtual que mais se aproxima do desempenho de servidores dedicados.

Capítulo 3

O Processo de Decisório para Auxílio do Administrador de Sistemas

Esse capitulo aborda o mecanismo de decisão para hipervisor e dimensão de máquina virtual, utilizando os conceitos e métricas subjetivas da ISO 25010, e os métodos AHP e TOPSIS.

3.1 ISO 25010

Para que uma melhor avaliação dos hipervisores selecionados neste trabalho seja condizida, utilizou-se alguns conceitos das métricas de qualidade de software propostos na ISO 25010. Dentre todas as métricas listadas na ISO, foram escolhidos cinco métricas que são coerentes para comparação de hipervisores, são elas:

- Operabilidade;
- Proteção contra erros do usuário;
- Interoperabilidade;
- Recuperabilidade;
- Segurança.

3.1.1 Operabilidade

Essa característica mensura o quão operável um software consegue ser, de acordo com a ISO/IEC 25010. Essa métrica foi usada para definir o grau de operabilidade de cada hipervisor considerando variáveis observadas no trabalho que contribuem para a usabilidade.

Para mensurar o seu nível, tivemos que selecionar variáveis para quantificar a operabilidade de um hipervisor. Essas variáveis foram escolhidas a partir da observação do funcionamento do hipervisor, considerando seu efeito na usabilidade do mesmo. Dessa forma, observamos que existem pelo menos duas funções básicas para a interação do usuário com o hipervisor: a *Command Line Interface* (CLI) e a *Graphical User Interface* (GUI). Logo, para cada hipervisor testado neste trabalho, consideramos se existia uma CLI ou GUI nativos e também a possibilidade de instalação externa de alguma das duas interfaces.

Então para avaliarmos cada hipervisor no quesito operabilidade, foi atribuída uma pontuação para cada tipo de interface que o hipervisor dispõe, considerando que se existe nativamente uma CLI ou GUI, o hipervisor está mais adequado na operabilidade do que se existe apenas uma GUI, CLI externa, visto que geralmente CLI e GUI nativas apresentam mais funcionalidades para o hipervisor do que uma ferramente externa.

As variáveis observadas para a métrica de operabilidade, bem como os valores atribuídos para cada uma delas, são listadas na Tabela 3.1. Os valores apresentados na Tabela 3.1 foram obtidos arbitrariamente, observando quais características são mais importantes considerando a métrica operabilidade.

Variável	Pontuação
Existe CLI	0,5
Existe GUI	0,5
Existe CLI externo	0,25
Existe GUI externo	0,25

Tabela 3.1: Tabela de Variáveis de Operabilidade.

Uma vez definidos os scores para as métricas de operabilidade, foi analisado cada hipervisor considerado no trabalho e foi obtido a seguinte tabela 3.2 com os scores dos hipervisores:

Hipervisor	Soma
VBox	1
KVM	0,75
XEN	1
Vmware Player	0,5
Docker	0,75

Tabela 3.2: Somatório de Pontos da métrica Operabilidade por Hipervisor.

Na Tabela 3.2 podemos observar que o Virtualbox adquiriu a nota 1 por possuir GUI e CLI nativos, facilitando a operação do usuário deste hypervisor, já o KVM possui CLI nativa com o software Virsh, e apresenta também a possibilidade de instalar uma GUI externa chamada de virt-manager.No caso do XEN, temos CLI e GUI nativos com o uso do Xencenter e da CLI do Xenserver. O Vmware player obteve a nota 0,5 por possuir apenas GUI nativo, e por fim, o docker obteve 0,75 por apresentar CLI nativa e uma possível GUI externa.

3.1.2 Proteção Contra Erro do Usuário

Outra métrica considerada neste trabalho é a proteção contra erro do usuário, que também é composta por variáveis observadas nos hipervisores comparados.

Para mensurar o nível de proteção contra erros do usuário, consideramos a existência de duas variáveis importantes para o hipervisor. A primeira é a proteção contra erros de criação de máquinas virtuais. Essa proteção garante que o usuário não cometa erros que comprometem os serviços prestados pelo hipervisor durante a criação de uma máquina virtual, como por exemplo checar se existem recursos suficientes para sua criação. A segunda variável é a proteção contra remoção de máquina virtual, que impede o usuário de apagar acidentalmente uma máquina virtual.

Nessa perspectiva, consideramos a proteção contra criação com mesmo peso da deleção devido a seu potencial danoso, visto que a criação de uma máquina virtual sem considerar os aspectos gerais do hipervisor pode afetar negativamente a gerência de recursos do hipervisor como um todo. Já a proteção contra deleção só existe prejuízo para os serviços prestados para aquela máquina virtual específica.

A seguir temos a Tabela 3.3 com os *scores* da cada variável observada para a métrica de

proteção contra erros do usuário:

Variável	Pontuação
Proteção contra criação	0,5
Proteção contra deleção	0,5

Tabela 3.3: Tabela de Variáveis de Proteção contra erros de usuário

Por fim, temos a Tabela 3.4 da soma dos *scores* de proteção contra erros de usuário:

Hypervisor	Soma
VBox	0,5
KVM	1
XEN	1
Vmware Player	1
Docker	1

Tabela 3.4: Somatório de Pontos da métrica Proteção contra erros do usuário

Na Tabela 3.4, todos os hipervisores considerados foram checados em relação a suas capacidades de proteção contra deleção e criação de máquinas virtuais, levando em conta os possíveis potenciais danosos, todas apresentaram tanto proteção para deleção quanto para criação de máquinas virtuais, com exceção do Virtualbox que não apresentou proteção contra criação de máquinas virtuais.

3.1.3 Interoperabilidade

No quesito interoperabilidade, consideramos um hipervisor interoperável com outros quando é possível migrar uma máquina virtual dele para outro, sendo esse processo direto ou indireto. No processo direto, a migração ocorre de forma transparente, apenas importando ou exportando uma máquina virtual entre diferentes hypervisores. Já no processo indireto, a migração necessita algum ajuste ou conversão de alguns parâmetros da máquina virtual para que o hipervisor destino obtenha a capacidade de receber e executar a máquina virtual.

Para mensurar os *scores* de interoperabilidade, comparamos par a par os hipervisores considerados neste trabalho, atribuindo o *score* 2 para o processo de migração direto e para o processo indireto, como mostra a Tabela 3.5 a seguir:

Hipervisor	VirtualBox	KVM	XEN	VMware	Docker
VirtualBox	2	2	1	2	0
KVM	1	2	1	1	0
XEN	1	1	2	1	0
Vmware	2	2	1	2	0
Docker	0	0	0	0	2
Soma de pontos	6	7	5	6	2

Tabela 3.5: Tabela Comparativa de Interoperabilidade

Na Tabela 3.5, podemos observar que todos os hipervisores foram comparados par a par em relação a sua capacidade de migração entre diferentes ambientes virtualizados ou diferente hipervisores. Foi possível notar que o hipevisor KVM conseguiu obter o melhor score em interoperabilidade com 7 pontos, seguido do VirtualBox e Vmware, ambos com 6, logo após o XEN com 5 pontos e por fim o docker com 2 pontos.

3.1.4 Recuperabilidade

A métrica de recuperabilidade é usada para garantir que o hipervisor, após sofrer uma interrupção ou falha, seja capaz de recuperar o estado desejado de seu funcionamento. Neste trabalho, consideramos a recuperabilidade como uma combinação de duas outras métricas subjetivas da ISO 25010, a Operabilidade e a Interoperabilidade. Nós observamos que quando ocorre uma falha em um hipervisor, é importante o sistema prover uma interface de operação para identificar e corrigir a causa da falha, bem como ter a possibilidade de durante o *troubleshooting* conseguir migrar as máquinas virtuais com serviços essenciais para outros hospedeiros ou outros hipervisores. Sendo assim, consideramos o *score* de recuperabilidade como uma soma dos *scores* das outras métricas (Operabilidade e Interoperabilidade).

A Tabela 3.6 a seguir mostra o *score* de cada hipervisor, somando seus respectivos *scores* de operabilidade e interoperabilidade:

Hipervisor	Soma
VBox	7
KVM	7,75
XEN	6
Vmware Player	6,5
Docker	2,75

Tabela 3.6: Somatório de pontos da métrica Recuperabilidade.

Podemos observar que o hipervisor KVM conseguiu o melhor *score* com 7,75 pontos,seguido do Virtualbox com 7 pontos, o Vmware e XEN com 6,5 e 6,0 pontos respectivamente e, por fim, o docker com 2,75 pontos.

3.1.5 Segurança

Na métrica de segurança, consideramos todos os hipervisores com *scores* 1 pelo nível de maturidade e isolamento entre a máquina virtual e o hospedeiro. Porém, o docker ainda sofre com vulnerabilidades quando é preciso executar aplicações de mais baixo nível (*kernel level aplications*).

A tabela abaixo mostra o sumário de scores da métrica de segurança:

Hipervisor	Soma
VBox	1
KVM	1
XEN	1
Vmware Player	1
Docker	0

Tabela 3.7: Pontuação de cada hipervisor na métrica de segurança.

3.1.6 Desempenho

A métrica objetiva de desempenho também é considerada na ISO 25010 e foi abordada em nosso trabalho. Para calcular o desempenho de cada hipervisor, foram realizados testes, *ben-*

chmarks de CPU, disco e memória RAM para cada hipervisor, comparando também com o desempenho da máquina física sem o uso de virtualização.Cada experimento foi realizado com 20 repetições, garantindo um nível de confiança de 95%. Com isso é possível observar o *overhead* [48] gerado por cada hipervisor em relação ao hardware dedicado sem virtualização.

A especificação do hardware utilizado nos testes foi a seguinte:

- CPU: AMD Ryzen 2700x (8 cores, 16 threads)
- RAM: 2 x 8GB RAM DDR4 @ 3000mhz
- DISCO: 500 GB SATA3 7.2k RPM

Capacidade de Processamento

Para mensurar o desempenho de processamento de cada hipervisor utilizamos o software de *benchmark Timed Linux Kernel Compilation* [5]. Esse *benchmark* calcula em quanto tempo o sistema consegue compilar o *kernel* do Linux na sua configuração padrão. Esse *benchmark* é amplamente utilizado em trabalhos acadêmicos e serve como referencia para comparar desempenho de CPU.

Abaixo segue a Figura 3.1 e a tabela 3.8 com os resultados obtidos pelo *benchmark* em cada hipervisor:

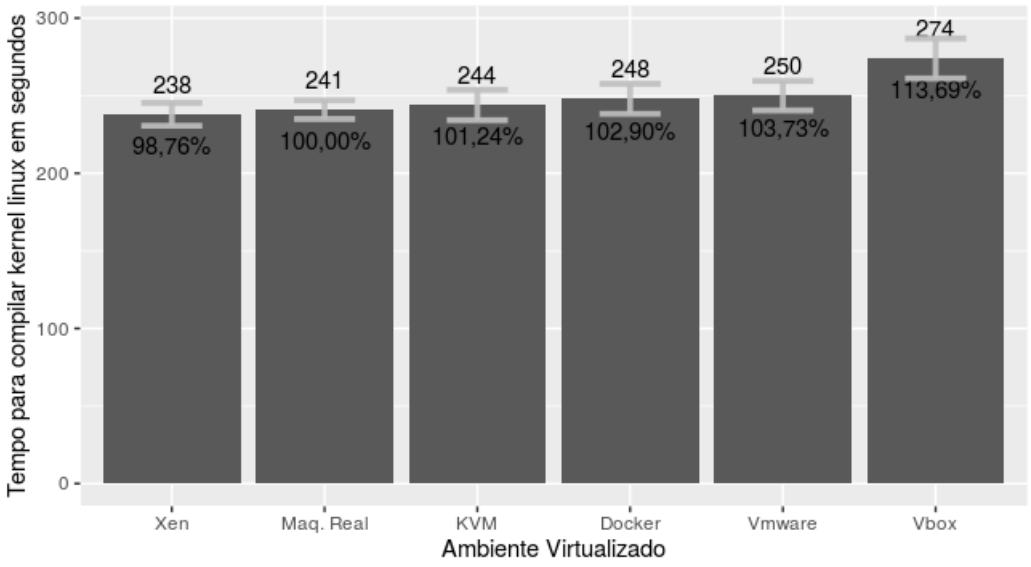

Figura 3.1: *Benchmark* de CPU,Tempo para compilar o *kernel* do linux.

Hipervisor	Tempo em segundos	Intervalo de Confiança
XEN	238	+- 7.37 (230,63 - 245,37)
Maq. Real	241	+- 6.02 (234,98 - 247,02)
KVM	244	+- 9.76 (234,24 - 253,76)
Docker	248	+- 9.67 (238,33 - 257,67)
Vmware Player	250	+- 9.76 (240,24 - 259,76)
VBox	274	+- 7.37 (266,63 - 281,37)

Tabela 3.8: Valores de tempo obtidos do *Benchmark* de CPU com seus limites superiores e inferiores

Podemos observar que na Figura 3.1 que existe valores de *overhead* diferentes para cada hipervisor, sendo o XEN o que apresenta uma tendência com menores valores de *overhead* em relação à máquina com o hardware dedicado. Nós acreditamos que esse resultado se deve ao fato de como o XEN opera suas chamadas de sistema através da sua ferramenta ”*Quick Call Manager*”, possibilitando o sistema convidado fazer chamadas diretas de uma aplicação sem a necessidade de passar através do VMM para cada chamada [32]. Os demais ambientes de virtualização apresentam valores de tempo de compilação do *kernel* superiores ao da máquina real, tendo um aumento sutil no caso do KVM, Docker e VMWare (menos de 4%). Já o VirtualBox apresenta um acréscimo bem mais significativo, de quase 14%.

Taxa de transferência da memória principal

Para avaliar o desempenho da memória RAM em diferentes hipervisores, utilizamos o software de *benchmark* RAMSpeed [4]. Com ele é possível medir a taxa de transferência efetiva da memória principal do sistema. Esse *benchmark* foi criado em 2002, é usado em vários trabalhos acadêmicos e serve como referência para comparar desempenho de módulos de memória RAM. Esse *benchmark* pode ser executado de várias formas diferentes, nesse trabalho optamos por executar cópias de números inteiros e números reais para a memória principal e medir a vazão de bytes por segundo que a memória consegue alcançar.

Abaixo segue a Figura 3.2 e a tabela 3.9com os resultados obtidos por cada hipervisor:

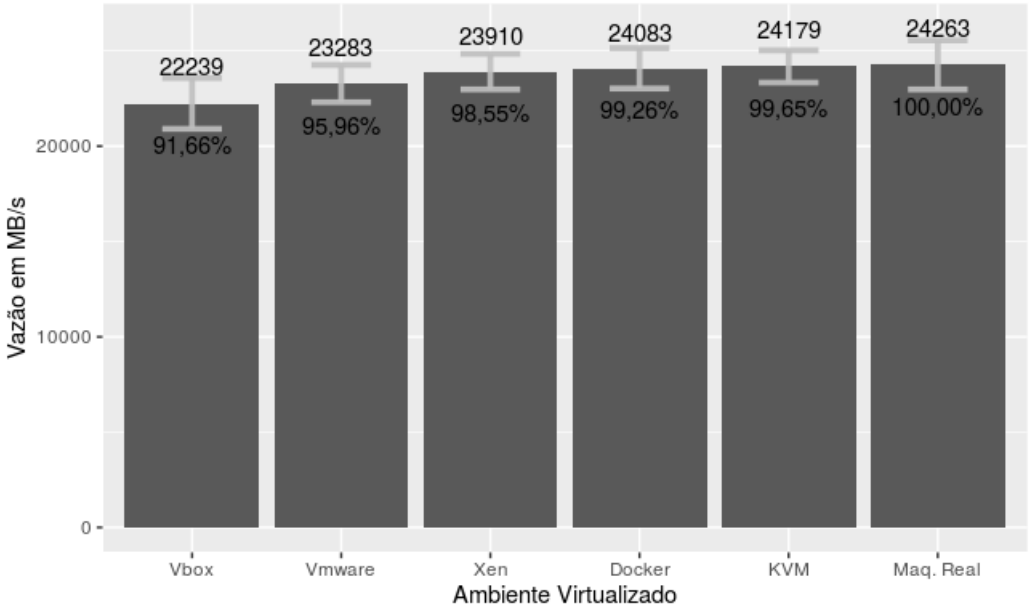

Figura 3.2: *Benchmark* RAM, vazão da memória RAM.

Hipervisor	Vazão em MB/s	Intervalo de Confiança
VBox	22239	+- 1334.34 (20904,66 - 23.573,34)
Vmware Player	23283	+- 977.88 (22305,12 - 24260,88)
XEN	23910	+- 932.49 (22977,51 - 24842,49)
Docker	24083	+- 1059.65 (23023,35 - 25142,65)
KVM	24179	+- 846.26 (23332,74 - 25025,26)
Maq. Real	24263	+- 1285.94 (22977,06 - 25548,94)

Tabela 3.9: Valores de vazão obtidos do *Benchmark* de RAM com seus limites superiores e inferiores

Podemos notar que na Figura 3.2 que estatisticamente não houve grande diferença entre os ambientes de virtualização, entretanto o KVM apresentou uma tendência de desempenho maior no *benchmark* entre os ambientes de virtualização, obtendo uma vazão mais próxima da máquina real, indicando um menor *overhead*. Em seguida temos o Docker e o XEN, que também apresentaram um baixo *overhead* comparado à máquina real. Por fim, tivemos o VMWare Player com um impacto mediano, e o VirtualBox apresentando os piores resultados.

Taxa de leitura/escrita em disco

Para calcular o desempenho do disco nos hipervisores utilizados neste trabalho, utilizamos o software de *benchmark* Postmark [3], que é capaz de medir quantas transações por segundo a máquina consegue executar no disco. O Postmark foi criado em 2010 com o propósito de simular cargas de trabalho em disco semelhantes a serviços comuns de rede. Esse *benchmark* foi projetado para simular leitura e escrita de vários arquivos com tamanhos pequenos, similar ao que a acontece em servidores web e de email. Esse teste executa 25.000 transações com 500 arquivos simultaneamente, de tamanhos variando entre 5 e 512 kilobytes.

A seguir a Figura 3.3 apresenta os *benchmarks* realizados em cada hipervisor.

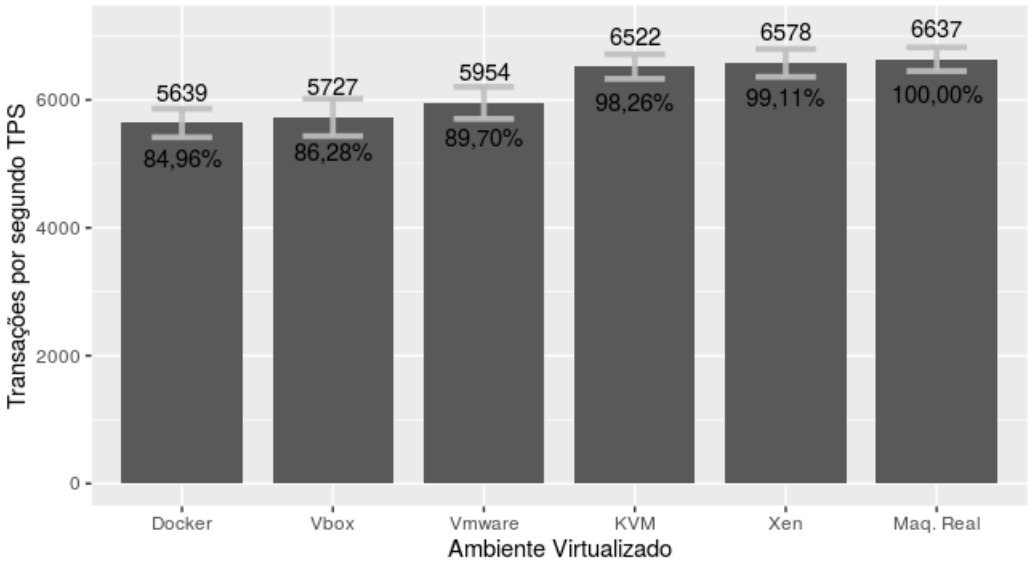

Figura 3.3: *Benchmark* de Disco, transações por segundo.

No *benchmark* de disco, a discrepância de *overhead* entre a máquina real e os hipervisores foi mais expressiva, chegando a apresentar uma diferença de quase de 20% entre a máquina real e o docker, apresentando o melhor caso o hypervisor XEN com somente 2,11% de *overhead*.

3.2 Processo de Decisão do Virtualizador

Sobre a tomada de decisão da ferramenta de sugestão de hipervisor, nós decidimos utilizar o método AHP [43] para ajudar na escolha de acordo com as preferências do administrador de sistema. Escolhemos o AHP porque permite a ponderação entre as métricas e considera também a preferencia do administrador em relação às métricas consideradas da ISO 25010 para seu ambiente virtualizado.

As variáveis subjetivas e objetivas da norma ISO/IEC 25010 mencionadas na seção 2.4 foram consideradas para montar a matriz de decisão. Logo, o método AHP nos permite analisar as métricas par a par e, de acordo com a entrada fornecida pelo administrador de sistemas, sugerir um ambiente de virtualização que atenda melhor às caraterísticas que o administrador busca em seu novo ambiente. O método apresenta como saída um vetor de decisão que indica, em porcentagem, qual a decisão que mais se adéqua aos pesos fornecidos pelo usuário. Na Figura 4.2 é apresentada a estrutura hierárquica do método AHP com as

variáveis da norma ISO/IEC 25010 consideradas neste trabalho.

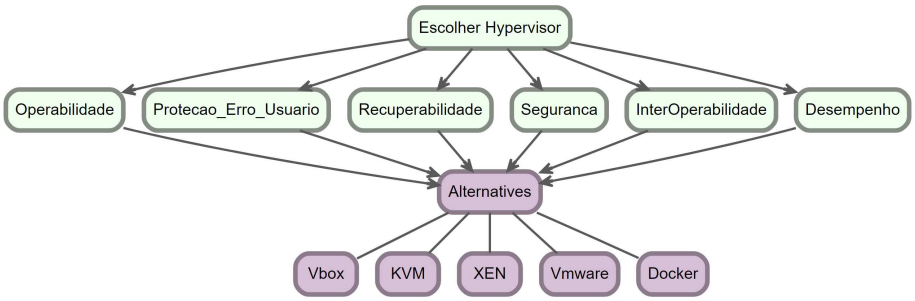

Figura 3.4: Estrutura hierárquica baseada nas métricas da ISO/IEC 25010.

3.3 Processo de Decisão para Dimensionamento de Máquinas Virtuais

Para o processo de decisão de dimensionamento de máquinas virtuais, utilizamos o método TOPSIS. Esse método tem como objetivo avaliar o desempenho de alternativas para a tomada de decisão através da similaridade com a solução ideal. Nesse processo, consideramos os conjuntos **F** com f *flavors* , **S** com s serviços de rede, **T** com t *threads*, e podemos considerar também um custo **C** para cada serviço ou *flavor*. Para um conjunto formado com um *flavor*, um serviço e uma quantidade de *threads*, podemos executar um *benchmark* e obter um resultado R:

$$\text{F} = \{f_1, f_2, ..., f_n\}$$
$$\text{S} = \{s_1, s_2, ..., s_n\}$$
$$\text{T} = \{t_1, t_2, ..., t_n\}$$
$$\text{C} = \{c_1, c_2, ..., c_n\}$$
$$\text{R} = \{f_i, s_i, t_i\} \forall i \in [1, 2, ..., n]$$

Nesse trabalho, consideramos que os valores dos conjuntos de *Flavors*, Serviços e *Threads* são obtidos através do administrador de TI, bem como o custo se o administrador de TI estiver considerando o uso de uma *Cloud* Pública. Esses valores são usados como base para o processo de decisão do tamanho do *flavor*.

Tipicamente em ambientes de produção são escolhidos padrões de *flavors* baseados na quantidade de recursos e na experiência do administrador do sistema, portanto consideramos F um conjunto finito de *flavors*. A escolha dos tipos de *flavor* pelo administrador de TI é uma etapa de grande importância para a execução dessa decisão de *flavor*. Isso determinará quais as possíveis escolhas de flavor que nosso sistema de decisão vai ter que considerar para apresentar sua proposta.

O conjunto de *Threads* T representa uma previsão de demanda do serviço S. Esse valor é fornecido pelo administrador de TI de acordo com a demanda relativa de cada serviço previsto para seu ambiente de TI, podendo representar vários tipos de demanda de acordo com o serviço. Por exemplo, a quantidade de acessos simultâneos a um serviço HTTP ou a quantidade de requisições por segundo a um banco de dados. A demanda influencia diretamente no resultado R do teste/*benchmark* executado.

Para cada Serviço S é escolhido um *benchmark* adequado para executar testes que se assemelham a uso do serviço. Desta forma, o valor de R é um teste realizado utilizando o *flavor* fn , com o serviço de rede sn, um custo cn e uma quantidade de *threads* tn. Com os resultados obtidos dos testes feitos para cada serviço, conseguimos montar uma matriz de decisão M onde cada linha da matriz representa resultados de um *flavor* fn e cada coluna representa os resultados para cada critério/serviço considerado na decisão.

Matrix de decisão M para a quantidade de *threads* Tn:

$$\mathrm{M} = \begin{bmatrix} f1.s1(c1) & f1.s2(c2) & f1.sn(cn) \\ f2.s1(c1) & f2.s2(c2) & f2.sn(cn) \\ fn.s1(c1) & fn.s2(c2) & fn.sn(cn) \end{bmatrix}$$

Por fim, com a matriz de decisão M, executamos o método TOPSIS para apresentar a decisão do *flavor* com base nos resultados obtidos nos testes. O método TOPSIS apresenta na sua saída um *ranking* com as possíveis alternativas de *flavor*. Esse método multi-critério considera todos os *scores* obtidos em cada *flavor* para apresentar o *ranking* com a melhor opção de *flavor* para uma determinada demanda com tn *threads*.

Como exemplo, considere o conjunto de *Flavors*

$$F = \{Small, Medium, Large\}$$

onde:

- Flavor Small = 1 CPU virtual e 512 MB de ram;
- Flavor Medium = 1 CPU virtual, 1GB ram
- Flavor Large, = 2 CPU virtual, 2GB RAM

E o conjunto de serviços $S = \{HTTP, BancoDados\}$ onde:

- Servidor Web HTTP, com desempenho medido através do *benchmark* apachebench. (Unidade em Requisições Por Segundo.)
- Servidor de Banco de dados, com desempenho medido atráves de *benchmark* sysbench. (Unidade em Transações por Segundo.)

Por fim, o conjunto de *threads* ou demanda de acessos/requisições simultâneas $T = \{1, 10\}$ onde:

- 1 *Thread* = Apenas um cliente utilizando/acessando o serviço simultaneamente.
- 10 *Threads* = Dez clientes utilizando/acessando o serviço simultaneamente.

Com os dados do exemplo acima teríamos uma matriz de decisão para cada quantidade de *threads*, ou seja, duas matrizes, uma para 1 *thread* e a segunda para 10 *threads*:

$$M_1 = \begin{bmatrix} small.http & small.BancoDados \\ medium.http & medium.BancoDados \\ large.http & large.BancoDados \end{bmatrix} \tag{3.1}$$

$$M_{10} = \begin{bmatrix} small.http & small.BancoDados \\ medium.http & medium.BancoDados \\ large.http & large.BancoDados \end{bmatrix} \tag{3.2}$$

Com as matrizes de decisão M1 para 1 *thread* e M10 para 10 *threads*, podemos executar o método TOPSIS e obter a recomendação de *flavors* , de acordo com os resultados dos testes de *benchmarks* para cada serviço:

Ranking p/ 1 *Thread*	*Flavor*
1^o	*small*
2^o	*large*
3^o	*medium*

Tabela 3.10: Exemplo *Ranking* TOPSIS para 1 *thread*.

Ranking p/ 10 *Threads*	*Flavor*
1^o	*large*
2^o	*medium*
3^o	*small*

Tabela 3.11: Exemplo *Ranking* TOPSIS para 10 *threads*.

Observando as Tabelas 3.10 e 3.11 podemos concluir quais *flavors* o método TOPSIS recomendou para diferentes *threads* no exemplo apresentado.

Abaixo temos a Figura 4.3 que apresenta o fluxograma de todo o processo de decisão de dimensionamento de máquina virtual citado acima.

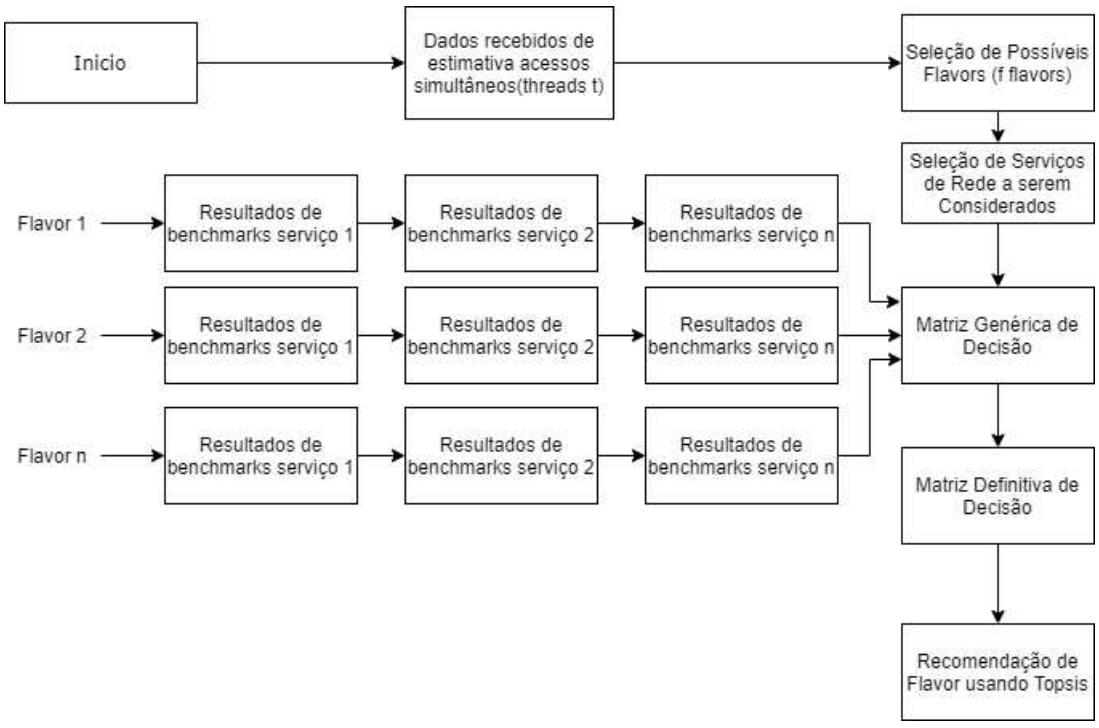

Figura 3.5: Fluxograma do processo de decisão do dimensionamento de VMs

3.4 Processo Completo

O processo completo deste trabalho começa através da necessidade do administrador de sistema de migrar de um ambiente de serviços de rede que é executado em servidores físicos (*bare-metal*) para um ambiente virtualizado, que atenda as mesmas demandas que os servidores físicos já conseguiam atender. Para isso, consideramos que o o administrador de sistemas já tem o conhecimento prévio de quais serviços de rede ele necessita realizar para a migração e qual demanda cada serviço apresentava nos servidores físicos.

Para iniciar a migração, a primeira pergunta a ser respondida será qual hipervisor utilizar dentre tantas opções viáveis no mercado. Este trabalho propõe inicialmente um esquema de sugestão de hipervisor baseando-se nas métricas de qualidade de software ISO/IEC 25010, escolhidas como relevantes para este trabalho. Após o usuário analisar quais métricas da ISO/IEC 25010 são preferenciais em relação às outras consideradas no trabalho, utilizamos o método AHP para apresentar uma pontuação dos hipervisores baseada nas preferências do administrador de sistemas. Com o resultado obtido, o administrador consegue tomar uma decisão mais coerente com o que ele espera do ambiente virtualizado.

O segundo passo do trabalho é definir qual a dimensão da máquina virtual que vai atender a demanda do administrador para determinado serviço de rede. Consideramos que o administrador de sistemas já sabe em média qual a demanda para cada serviço de rede provido no ambiente de máquinas físicas. Sabemos também que existe uma preferência do administrador em relação aos tipos de flavor que ele pretende utilizar no seu ambiente virtualizado.

Com essas informações, conseguimos montar uma matriz de decisão para cada média de demanda esperada pelo administrador de sistemas. Essas matrizes vão conter conjuntos de serviços e *flavors* para obter uma recomendação de *flavor* utilizando o método TOPSIS a partir de cada demanda, facilitando a escolha do administrador para cada serviço utilizado no sistema.

Por fim, o resultado final do trabalho é propor uma recomendação de hipervisor e *flavor* adequada para migração do ambiente de máquinas físicas para um ambiente virtualizado, de modo que essas recomendações apresentem o mínimo de diferenças possíveis em relação a sua execução em ambiente de máquinas físicas. É esperado que as recomendações consigam atender de forma satisfatória as necessidades do administrador de sistemas, agregando a

flexibilidade de um ambiente virtualizado para gerência dos serviços de rede utilizados pelo administrador.

Capítulo 4

Avaliação Experimental e Validação

Neste capítulo apresentamos uma validação do mecanismo proposto no capítulo anterior, utilizando os conceitos abordados para auxílio na decisão de hipervisor e dimensão de máquina virtual, comparando as decisões obtidas do mecanismo com resultados obtidos de um ambiente com servidor dedicado.

4.1 Decisão do Hipervisor

Para validar o sistema de recomendação proposto neste trabalho, executamos um cenário de testes com características especificas nas quais o suposto administrador de sistemas apresentou suas preferências de um ambiente virtualizado, bem como as demandas dos serviços de rede por ele administrados.

No primeiro momento, foi simulado um questionário de preferências das métricas da ISO/IEC 25010 do administrador de sistemas. Esse questionário obteve um resultado par a par de preferência das métricas da ISO 25010 escolhidas pelo administrador, com a escala de valores entre 9 e $\frac{1}{9}$, onde o valor 9 é fortemente favorável a primeira variável do par em relação a segunda, o valor $\frac{1}{9}$ fortemente desfavorável a primeira variável em relação a segunda e no valor 1 ambas variáveis tem a mesma importância. Na Tabela 4.1 mostramos um exemplo de preferência do administrador de sistemas.

Métrica 1	Métrica 2	Valor de preferência
Operabilidade	Proteção Erro Usuário	3
Operabilidade	Recuperabilidade	1
Operabilidade	Segurança	$\frac{1}{3}$
Operabilidade	InterOperabilidade	$\frac{1}{2}$
Operabilidade	Desempenho	$\frac{1}{5}$
Proteção Erro Usuário	Recuperabilidade	$\frac{1}{2}$
Proteção Erro Usuário	Segurança	$\frac{1}{2}$
Proteção Erro Usuário	InterOperabilidade	$\frac{1}{5}$
Proteção Erro Usuário	Desempenho	$\frac{1}{5}$
Recuperabilidade	Segurança	1
Recuperabilidade	InterOperabilidade	1
Recuperabilidade	Desempenho	$\frac{1}{2}$
Segurança	InterOperabilidade	1
Segurança	Desempenho	1
Desempenho	InterOperabilidade	2

Tabela 4.1: Exemplo de tabela com preferência do administrador de sistemas em relação as métricas ISO 25010.

Podemos notar que as preferências do administrador nesse cenário da Tabela 4.1 priorizaram basicamente 3 variáveis, Desempenho, Segurança e Interoperabilidade. Após executarmos o método AHP utilizando as preferências da Tabela 4.1 obtemos o seguinte resultado para a escolha de Hipervisor mostrado na Figura 4.1 a seguir.

	Weight	KVM	XEN	Vmware	Vbox	Docker
Escolher Hypervisor	100.0%	23.4%	22.6%	20.8%	20.5%	12.8%
Desempenho	31.2%	6.6%	7.3%	6.1%	5.1%	5.9%
Seguranca	20.0%	4.8%	4.8%	4.8%	4.8%	1.0%
InterOperabilidade	18.6%	5.4%	3.5%	4.2%	4.2%	1.4%
Recuperabilidade	14.3%	3.7%	3.1%	3.0%	3.2%	1.3%
Operabilidade	10.1%	1.6%	2.6%	1.5%	2.5%	1.9%
Protecao_Erro_Usuario	5.7%	1.3%	1.3%	1.3%	0.6%	1.3%

Figura 4.1: Recomendação AHP do cenário de validação.

De acordo com a Figura 4.2, o Hipervisor KVM obteve maior recomendação na execução do método AHP, obtendo 23.4% de recomendação, seguido do XEN com 22.6%. Logo, de acordo com as preferências do administrador de sistema, o hipervisor que melhor atende às necessidades do administrador considerando os resultados obtidos para as métricas de qualidade de software da ISO/IEC 25010 é o Hipervisor KVM. Esse resultado poderia mudar dependendo dos valores apresentados na tabela de preferências do administrador. O KVM foi escolhido neste cenário devido aos valores escolhidos no exemplo de preferência, onde outro cenário com preferências distintas, o resultado poderia variar e o método recomendar outro hipervisor. Com base nesse resultado obtido no exemplo, vamos utilizar durante o restante do cenário de validação o Hipervisor KVM para os testes a seguir de validação apresentados neste capítulo.

4.2 Decisão de Dimensão das Máquinas Virtuais

No processo de decisão da dimensão de máquinas virtuais, consideramos que no nosso teste de validação a escolha do conjunto de *flavors* do administrador se resume a três *flavors* diferentes. Esses *flavors* executarão 3 *benchmarks* que representam 3 tipos de serviços diferentes: o primeiro *benchmark* representa um servidor web básico testado a partir de software do teste Apache bench,esse serviço de web básico pode ser considerado como uma requisição a uma *API*, o segundo serviço representa um banco de dados convencional testado a partir do *benchmark* Sysbench Mysql e, por fim, o terceiro serviço representa uma página completa que chamamos de web avançado. Assim a validação envolve tecnologias web mais recentes

e o uso de banco de dados para armazenar as informações do serviço web. Nesses testes rodamos o Apache bench em uma instalação limpa do software wordpress.

Com o conjunto de *flavors* e o conjunto de serviços, conseguimos montar uma matriz de decisão para cada valor de demanda (*threads*). Na Tabela 4.2 temos os três tipos de *flavors* que servirão para nossos testes de validação, lembrando que todos os testes utilizarão o hipervisor KVM que foi recomendado através do método AHP na seção anterior.

Flavor	*vpcus*	RAM(Gb)
1	1	1
2	2	2
3	4	4

Tabela 4.2: Pontuação de cada hipervisor na métrica de segurança.

4.2.1 Avaliação de Desempenho dos *Flavors*

O primeiro serviço considerado no teste de validação é o Web Básico, que consiste em uma instalação limpa do servidor web Apache 2 e apresenta a página html básica de boas vindas ao software. Esse serviço pode ser comparado a serviços de páginas web estáticas, nos quais o servidor web trabalha sem depender de outros softwares. Para mensurar o desempenho obtido nesse teste, utilizamos o software Apache Bench para medir quantas requisições por segundo o *flavor* consegue atender. Executamos o teste 20 vezes com um nível de confiança de 95%. Na Figura 4.2 mostramos os resultados do *benchmark* usando o apache bench em um serviço de web básico.

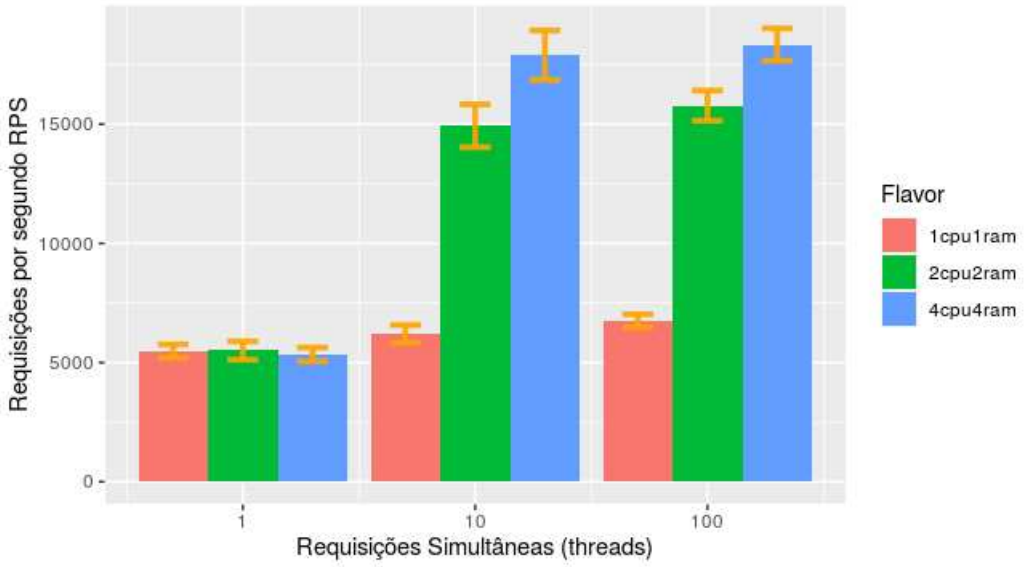

Figura 4.2: Apache Bench para Web Básico

Podemos observar na Figura 4.2 que para requisições de 1 *thread*, representando a requisição de 1 usuário simultâneo, o desempenho foi semelhante para os 3 *flavors*, ficando próximo das 5 mil requisições por segundo. Já para 10 *threads* é possível notar que o *flavor* de 1 vpcu e 1Gb de RAM entrega um desempenho significativamente inferior em relação aos outros dois *flavors*. Esse comportamento se repete também para 100 *threads*. Entretanto o flavor de 2 vcpu e o flavor de 4 vCPU entregam desempenho semelhante para 10 e 100 threads, essa semelhança de desempenho se deve ao fato que mesmo aumentando a concorrência para requisições simultâneas, o hardware testado com 10 *threads* chegou no seu limite de uso de *IO*, logo o aumento de capacidade de processamento e memória principal não atinge melhores resultados.

O segundo serviço avaliado por cada um dos três *flavors* foi o de banco de dados, utilizando a ferramenta de *benchmark* Sysbench Mysql. Esse benchmark realiza operações de consulta e escrita utilizando uma tabela com tamanho de 10 mil linhas e mede quantas transações por segundo a máquina consegue realizar. Também é possível configurar o número de *threads* que acessam o sistema simultaneamente. Abaixo segue a Figura 4.3 com os *benchmarks* do Sysbench Mysql para mensurar a performance do uso de banco de dados em cada *flavor*.

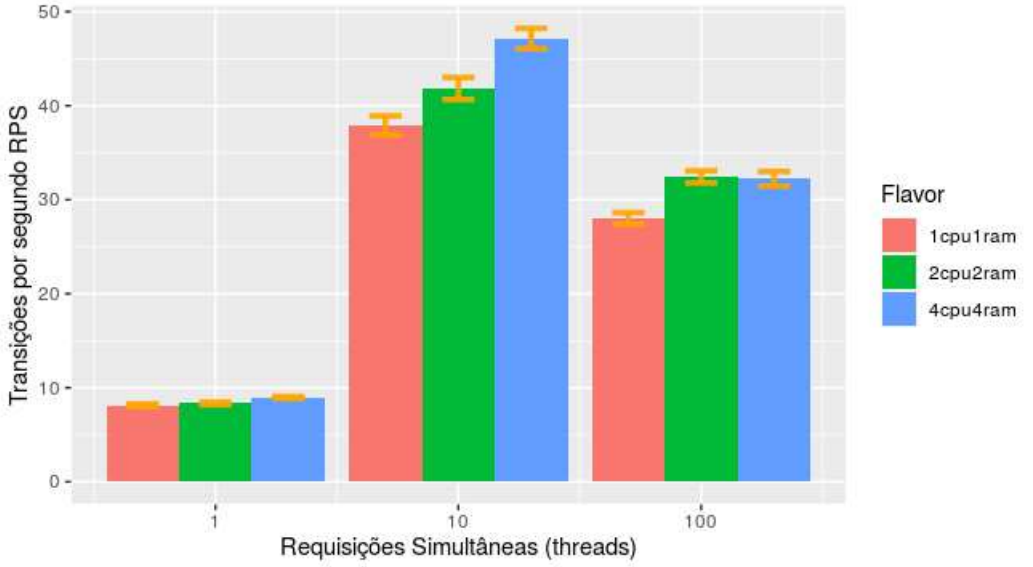

Figura 4.3: Sysbench Mysql para serviço banco de dados

Na Figura 4.3 podemos notar que para 1 requisição simultânea (1 *thread*), os valores de transação por segundo aparecem muito próximos, com tendência de maiores valores para o flavor de 4vcpu e 4gb de ram, já para 10 requisições simultâneas (10 *threads*) podemos observar que existe sim uma melhora nas transações por segundo quando aumentamos a quantidade de recursos. Para 100 requisições simultâneas (100 *threads*), podemos ver que os valores de transações por segundo ficam menores com relação aos valores de 10 requisições simultâneas. Isso se deve pelo mesmo fato de limitação da capacidade de *IO* citado na 4.2, mas, como esperado, o *flavor* com mais recursos conseguiu obter melhores valores de transações por segundo em relação aos outros *flavors* com 100 requisições por segundo.

Por fim, consideramos o serviço web avançado, que é composto de servidor web, tecnologias web dinâmicas como CSS e PHP e um banco de dados. Esse cenário é o que representa melhor os serviços web atuais. Para esse teste usamos novamente o *benchmark* Apache Bench, porém é executado em cima de uma instalação do Wordpress limpa com sua página padrão. O Wordpress é um sistema gerenciador de conteúdo de código aberto, utilizado por mais de 40% de todos os sites web da Internet [37].

Para o teste do apache bench usando uma página do wordpress, utilizamos os mesmos parâmetros do teste web básico, executamos 10 mil requisições para o servidor web e mensuramos quantas requisições por segundo cada flavor consegue lidar.

Na Figura 4.4 apresentamos os resultados de *benchmark* do apache bench para cada flavor:

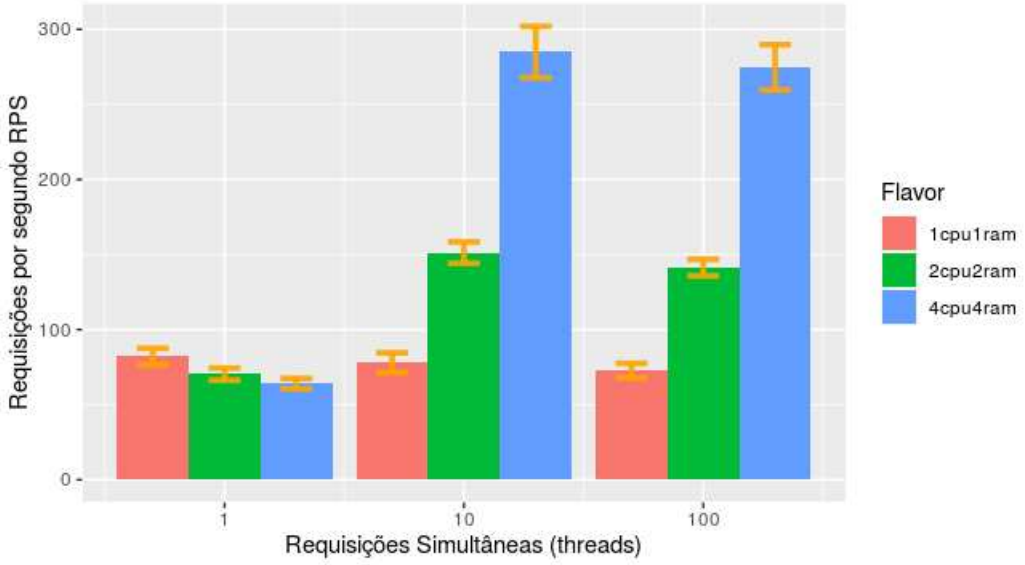

Figura 4.4: Apache Bench para web avançado(wordpress)

Na Figura 4.4 podemos notar que, para 1 requisição simultânea (1 *thread*), o *flavor* com menos recursos conseguiu um desempenho maior em relação aos outros *flavors* por causa da única requisição simultânea aproveitar melhor o uso de uma só cpu em relação a divisão de carga de um acesso simultâneo para 2 e 4 cpus. Já para 10 requisições simultâneas podemos observar que o *flavor* com mais recursos conseguiu alcançar o melhor desempenho e, por fim, com 100 requisições simultâneas (100 *threads*) o mesmo acontece. O flavor com 4cpu e 4gb de RAM consegue alcançar um melhor desempenho. É possível notar também que o desempenho dos *flavors* 1 vcpu 1GB e 2v cpu 2GB diminuem em relação a 10 requisições simultâneas, pelo fato de existir maior concorrência para requisitar a página do wordpress.

4.2.2 Desempenho vs. Custo

Para complementar o aspecto do desempenho na decisão, também foi considerada a métrica do custo do *flavor*. Essa métrica de custo vai colaborar com a tomada de decisão da dimensão do *flavor*, sendo considerada como mais uma métrica de avaliação para a escolha. Para obter valores de exemplo, consideramos as *clouds* públicas mais utilizadas do mercado e escolhemos a Azure [35] como base de preços para o custo dos nossos *flavors* escolhidos

[36].

Após conseguir todos esses resultados dos 3 serviços selecionados pelo administrador de sistemas em conjunto com os valores de custos, ficamos então com a tarefa de decidir qual *flavor* se encaixa melhor para cada demanda apresentada , como mostrado no capítulo anterior. Optamos por tomar a decisão de *flavor* utilizando o método TOPSIS, uma vez que nos permite tomar decisões de forma mais objetiva, comparando os resultados obtidos nos *benchmarks* de cada serviço e apresentando uma escolha de *flavor* que atenda melhor o administrador de sistema. Considerando ainda que tanto o valor do desempenho do serviço quanto o valor do custo possuem o mesmo peso na tomada de decisão.

Nessa etapa, foram geradas 9 matrizes de decisão, três para cada demanda (quantidade de *threads*). Para cada valor de serviço de rede e demanda usados no teste de validação, teremos uma recomendação de *flavor* diferente.

Para a primeira demanda de 1 requisição simultânea no serviço web, montamos a matriz de decisão do método TOPSIS utilizando os valores obtidos de *benchmark* para 1 *thread* do serviço web básico e os seus valores de custo. Essa informações são apresentadas de maneira sumarizada na Tabela 4.3.

Flavor	Web básico	Custo
1Cpu1Ram	5480.06 RPS	0,079 $/hora
2Cpu2Ram	5502.51 RPS	0,168 $/hora
4Cpu4Ram	5339.72 RPS	0,351 $/hora

Tabela 4.3: Matriz de decisão para um acesso simultâneo (1 *thread*) no serviço web básico.

A partir desses valores, o método TOPSIS foi utilizado para o cálculo da matriz de decisão para 1 acesso simultâneo no serviço web básico. Os resultados obtidos são apresentados na Tabela 4.4.

Flavor	*Score* Topsis	Posição
1Cpu1Ram	0.9927451	1^o
2Cpu2Ram	0.6714199	2^o
4Cpu4Ram	0.0000000	3^o

Tabela 4.4: Recomendação do método TOPSIS para um acesso simultâneo no serviço web básico

Podemos observar na decisão apresentada na Tabela 4.4 que os *flavors* que foram indicados como mais adequados foram os com menor quantidade de recursos. Isso ocorreu devido à ponderação realizada entre o desempenho e o custo através do TOPSIS. Como os 3 *flavors* obtiveram resultados muito próximos nos *bechmarks* para aferir seu desempenho, possuem uma diferença bastante significativa de custo, o uso de soluções com menor quantidade de recursos foi mais indicado pelo método de decisão.

Para a demanda de 1 requisição simultânea no serviço banco de dados, montamos a matriz de decisão do método TOPSIS utilizando os valores obtidos de *benchmark* para 1 *thread* do serviço banco de dados e os seus valores de custo. Essa informações são apresentadas de maneira sumarizada na tabela 4.5.

Flavor	Banco de dados	Custo
1Cpu1Ram	8.14 TPS	0,525 $/hora
2Cpu2Ram	8.33 TPS	0,66 $/hora
4Cpu4Ram	8.95 TPS	0,935 $/hora

Tabela 4.5: Tabela com matriz de decisão para um acesso simultâneo (1 *thread*) no serviço de banco de dados.

A partir desses valores, o método TOPSIS foi utilizado para o cálculo da matriz de decisão para a decisão do sistema utilizando 1 acesso simultâneo no serviço de banco de dados. Os resultados obtidos são apresentados na Tabela 4.6.

Flavor	*Score* Topsis	Posição
1Cpu1Ram	0.8551714	1^o
2Cpu2Ram	0.6550246	2^o
4Cpu4Ram	0.1448286	3^o

Tabela 4.6: Recomendação do método TOPSIS para um acesso simultâneo no serviço banco de dados.

Podemos observar na decisão apresentada na Tabela 4.6 que acontece o mesmo cenário da recomendação do serviço web básico com 1 requisição simultânea. *Flavors* indicados como mais adequados foram os com menor quantidade de recursos. Mais uma vez, isso ocorreu devido à ponderação realizada entre o desempenho e o custo através do TOPSIS. Como o desempenho obteve valores aproximados nos 3 *flavors*, o critério do peso influenciou mais a decisão.

Para a demanda de 1 requisição simultânea no serviço web avançado,foi considerado o custo total desse serviço como a soma dos outros dois serviços (web básico + banco de dados),a partir dessa consideração montamos a matriz de decisão do método TOPSIS utilizando os valores obtidos de *benchmark* para 1 *thread* do serviço web avançado e os valores de custo apresentados na Tabela 4.6. Essa informações são apresentadas de maneira sumarizada na Tabela 4.7.

Flavor	Web Avançado	Custo
1Cpu1Ram	82.07 RPS	0,604 $/hora
2Cpu2Ram	70.31 RPS	0,828 $/hora
4Cpu4Ram	64.01 RPS	1,286 $/hora

Tabela 4.7: Tabela com matriz de decisão para um acesso simultâneo (1 *thread*) no serviço web avançado.

A partir desses valores, o método TOPSIS foi utilizado para o cálculo da matriz de decisão para a decisão do sistema utilizando 1 acesso simultâneo no serviço web avançado. Os resultados obtidos são apresentados na Tabela 4.8.

Flavor	Score Topsis	Posição
1Cpu1Ram	1.0000000	1^o
2Cpu2Ram	0.6312817	2^o
4Cpu4Ram	0.0000000	3^o

Tabela 4.8: Recomendação do método TOPSIS para um acesso simultâneo no serviço banco de dados.

Podemos notar na decisão apresentada na Tabela 4.8 que acontece novamente o mesmo cenário da recomendação do serviço web básico e do serviço de banco de dados com 1 requisição simultânea. O *flavor* com menos recurso foi o mais recomendado, levando em consideração a ponderação igual para o desempenho e custo.

Iniciando as demandas de 10 requisições simultâneas, temos a Tabela 4.9 que apresenta os resultados sumarizados dos *benchmarks* para o serviço de web básico, em conjunto com os valores de custo para cada *flavor*.

Flavor	Web básico	Custo
1Cpu1Ram	6200.96 RPS	0,079 $/hora
2Cpu2Ram	14933.20 RPS	0,168 $/hora
4Cpu4Ram	17893.83 RPS	0,351 $/hora

Tabela 4.9: Matriz de decisão para dez acessos simultâneos (10 *threads*) no serviço web básico.

A partir desses valores obtidos na Tabela 4.9, o método TOPSIS foi utilizado para o cálculo da matriz de decisão para 10 acessos simultâneos no serviço web básico. Os resultados obtidos são apresentados na Tabela 4.10.

Flavor	*Score* Topsis	Posição
1Cpu1Ram	0.5855582	2^o
2Cpu2Ram	0.6963616	1^o
4Cpu4Ram	0.4144418	3^o

Tabela 4.10: Recomendação do método TOPSIS para dez acessos simultâneos no serviço web básico.

É possível observar na Tabela 4.10 que o *flavor* mais indicado foi o intermediário com 2 vcpu e 2 GB RAM, seguido do *flavor* com menos recursos e por último o *flavor* com mais recursos.Nessa etapa os dois *flavors* com mais recursos conseguiram obter um resultado de *benchmark* significativamente superior ao *flavor* com menos recursos de 1 vcpu e 1 GB RAM, porém a ponderação considerando o custo resultou nessa classificação de recomendação apresentada na Tabela 4.10.

Continuando com as demandas de 10 requisições simultâneas, temos a tabela 4.11 que apresenta os resultados sumarizados dos benchmarks para o serviço de banco de dados,em conjunto com os valores de custo para cada flavor.

Flavor	Banco de Dados	Custo
1Cpu1Ram	37.89 TPS	0,525 $/hora
2Cpu2Ram	41.83 TPS	0,660 $/hora
4Cpu4Ram	47.14 TPS	0,935 $/hora

Tabela 4.11: Matriz de decisão para dez acessos simultâneos (10 *threads*) no serviço banco de dados.

A partir dos valores obtidos na Tabela 4.11, o método TOPSIS foi utilizado para o cálculo da matriz de decisão para a decisão do sistema utilizando 10 acessos simultâneos no serviço banco de dados. Os resultados obtidos são apresentados na Tabela 4.12.

Flavor	*Score* Topsis	Posição
1Cpu1Ram	0.7213409	1^o
2Cpu2Ram	0.6349927	2^o
4Cpu4Ram	0.2786591	3^o

Tabela 4.12: Recomendação do método TOPSIS para dez acessos simultâneos no serviço banco de dados.

Na Tabela 4.12 podemos observar que , para o serviço de banco de dados com 10 requisições simultâneas, a recomendação continuou sendo o *flavor* com menos recursos, 1 vcpu e 1GB RAM, seguido do *flavor* intermediário com o segundo lugar de recomendação e o *flavor* com mais recursos em terceiro lugar. Nessa etapa podemos perceber que o desempenho melhorou com o aumento de recursos nos diferentes *flavors*, mas o preço também

aumenta consideravelmente, o que faz com que a ponderação escolhida colabore com a decisão apresentada pelo método TOPSIS.

Finalizando com as demandas de 10 requisições simultâneas, temos a Tabela 4.13 que apresenta os resultados sumarizados dos *benchmarks* para o serviço de web avançado,em conjunto com os valores de custo para cada *flavor* retirados da Tabela 4.13.

Flavor	Web Avançado	Custo
1Cpu1Ram	77.93 RPS	0,604 $/hora
2Cpu2Ram	151.33 RPS	0,828 $/hora
4Cpu4Ram	284.88 RPS	1,286 $/hora

Tabela 4.13: Matriz de decisão para dez acessos simultâneos (10 *threads*) no serviço web avançado.

A partir dos valores obtidos na Tabela 4.13, o método TOPSIS foi utilizado para o cálculo da matriz de decisão para 10 acessos simultâneos no serviço web avançado. Os resultados obtidos são apresentados na Tabela 4.14.

Flavor	*Score* Topsis	Posição
1Cpu1Ram	0.3994171	3^o
2Cpu2Ram	0.4556672	2^o
4Cpu4Ram	0.6005829	1^o

Tabela 4.14: Recomendação do método TOPSIS para dez acessos simultâneos no serviço web avançado.

Para o serviço de web avançado, conseguimos avaliar a recomendação na Tabela 4.14, é possível observar que o *flavor* mais recomendado foi aquele com mais recursos, seguido do *flavor* intermediário e por fim o *flavor* com menos recursos. Na tabela 4.13 podemos notar um ganho significativo de desempenho quando aumentamos a quantidade de recursos, portanto o método TOPSIS mesmo considerando o custo, apresentou o *flavor* com mais recursos como melhor opção para o serviço de web avançado.

Em seguida, iniciamos as demandas de 100 requisições simultâneas, Temos a tabela 4.15 que apresenta os resultados sumarizados dos *benchmarks* para o serviço de web básico, em conjunto com os valores de custo para cada *flavor* retirados da tabela 4.14.

Flavor	Web básico	Custo
1Cpu1Ram	6754.89 RPS	0,079 $/hora
2Cpu2Ram	15775.18 RPS	0,168 $/hora
4Cpu4Ram	18338.84 RPS	0,351 $/hora

Tabela 4.15: Matriz de decisão para cem acessos simultâneos (100 *threads*) no serviço web básico.

A partir dos valores obtidos na Tabela 4.15, o método TOPSIS foi utilizado para o cálculo da matriz de decisão para 100 acessos simultâneos no serviço web básico. Os resultados obtidos são apresentados na Tabela 4.16.

Flavor	*Score* Topsis	Posição
1Cpu1Ram	0.5976198	2^o
2Cpu2Ram	0.7034727	1^o
4Cpu4Ram	0.4023802	3^o

Tabela 4.16: Recomendação do método TOPSIS para cem acessos simultâneos no serviço web básico.

Observando a Tabela 4.16, podemos notar que o *flavor* mais recomendado foi o intermediário, seguido do *flavor* com menos recursos e por fim o *flavor* com mais recursos.É possível perceber que na Tabela 4.15 o desempenho do *flavor* com menos recursos para o *flavor* intermediário mais que dobrou. Porém, do intermediário para o *flavor* com mais recursos não houve a mesma proporção de crescimento de desempenho. Considerando a ponderação entre desempenho e custo, o método TOPSIS recomendou o *flavor* intermediário.

Continuando com as demandas de 100 requisições simultâneas, temos a Tabela 4.17 que apresenta os resultados sumarizados dos *benchmarks* para o serviço de banco de dados,em conjunto com os valores de custo para cada *flavor* retirados da Tabela 4.16.

Flavor	Banco de dados	Custo
1Cpu1Ram	28.01 TPS	0,525 $/hora
2Cpu2Ram	32.43 TPS	0,660 $/hora
4Cpu4Ram	32.22 TPS	0,935 $/hora

Tabela 4.17: Matriz de decisão para cem acessos simultâneos (100 *threads*) no serviço banco de dados.

A partir dos valores obtidos na Tabela 4.17, o método TOPSIS foi utilizado para o cálculo da matriz de decisão para a decisão do sistema utilizando 100 acessos simultâneos no serviço banco de dados. Os resultados obtidos são apresentados na Tabela 4.18.

Flavor	*Score* Topsis	Posição
1Cpu1Ram	0.7979654	1^o
2Cpu2Ram	0.6852708	2^o
4Cpu4Ram	0.1942894	3^o

Tabela 4.18: Recomendação do método TOPSIS para cem acessos simultâneos no serviço banco de dados

Na Tabela 4.18 podemos observar que , para o serviço de banco de dados com 100 requisições simultâneas, a recomendação continuou sendo o *flavor* com menos recursos, seguido do *flavor* intermediário com o segundo lugar de recomendação e o *flavor* com mais recursos em terceiro lugar. Nessa etapa, podemos perceber que o desempenho degradou quando aumentamos a demanda de 10 para 100 requisições simultâneas e também diminuiu do *flavor* intermediário para o *flavor* com mais recursos para 100 requisições simultâneas. Essa degradação de desempenho se deve pelo fato do aumento da concorrência para escrita e leitura no disco, onerando o valor de transações por segundo que o serviço consegue atender. Devido a essa degradação de desempenho, o método TOPSIS considerando o custo e o desempenho baixo nas 100 requisições simultâneas, recomendou novamente o *flavor* com menos recursos.

Finalizando com as demandas de 100 requisições simultâneas, temos a Tabela 4.19 que apresenta os resultados sumarizados dos *benchmarks* para o serviço de web avançado, em conjunto com os valores de custo para cada *flavor* retirados da Tabela 4.18.

Flavor	Web básico	Custo
1Cpu1Ram	72.74 RPS	0,604 $/hora
2Cpu2Ram	141.33 RPS	0,828 $/hora
4Cpu4Ram	274.71 RPS	1,286 $/hora

Tabela 4.19: Matriz de decisão para cem acessos simultâneos (100 threads) no serviço web avançado.

A partir dos valores obtidos na Tabela 4.19, o método TOPSIS foi utilizado para o cálculo da matriz de decisão para 100 acessos simultâneos no serviço web avançado. Os resultados obtidos são apresentados na Tabela 4.20.

Flavor	*Score* Topsis	Posição
1Cpu1Ram	0.3945698	3^o
2Cpu2Ram	0.4438182	2^o
4Cpu4Ram	0.6054302	1^o

Tabela 4.20: Recomendação do método TOPSIS para cem acessos simultâneos no serviço web avançado.

Para o serviço de web avançado, conseguimos avaliar a recomendação na Tabela 4.20. É possível observar que o *flavor* mais recomendado foi o flavor com mais recursos, seguido do *flavor* intermediário e por fim o *flavor* com menos recursos. Na Tabela 4.19, podemos notar novamente um ganho significativo de desempenho quando aumentamos a quantidade de recursos, assim como aconteceu na demanda de 10 requisições por segundo no serviço web avançado. Portanto, o método TOPSIS mesmo considerando o custo, apresentou o *flavor* com mais recursos como melhor opção para o serviço de web avançado.

O resultado das sugestões do método TOPSIS pode variar de acordo com a matriz de peso utilizada, nesse cenário utilizando a matriz de peso considerada neste trabalho, os resultados das sugestões permanecerão os mesmos.

4.3 Comparação de Desempenho de Ambiente Virtualizado Baseado na Tomada de Decisão vs. Servidor Dedicado

Esta seção tem como propósito apresentar uma comparação entre as recomendações obtidas na seção anterior com o desempenho da máquina real, executando os mesmos *benchmarks* que simulam cargas de serviços de redes diferentes. O objetivo dessa comparação é verificar qual a perda de desempenho que o administrador de sistemas pode esperar em comparação com os servidores dedicados, uma vez que ele seguiu as recomendações de hypervisor utilizando o método AHP e a dimensão do *flavor* utilizando o método TOPSIS. Como apresentado antes neste capítulo, o hipervisor recomendado para o cenário hipotético foi o KVM e os *flavors* para cada serviço serão utilizados conforme recomendação do método TOPSIS.

Foram executados os *benchmarks* dos três serviços considerados no trabalho: web básico, banco de dados e web avançado. Para cada serviço foram comparados os resultados das sugestões de máquina virtual da seção anterior com uma máquina dedicada com a mesma quantidade de recursos. Na Figura 4.5, temos os resultados de desempenho do serviço web básico para 1 requisição simultânea. Foi comparada a máquina virtual com hipervisor KVM e com *flavor* de menor quantidade de recursos com a máquina dedicada também com 1 cpu habilitada de 1GB de RAM. Para a demanda de 10 e 100 requisições simultâneas, foi utilizado o *flavor* intermediário com 2 vcpu e 2 GB RAM no hipervisor KVM, assim como na máquina dedicada.

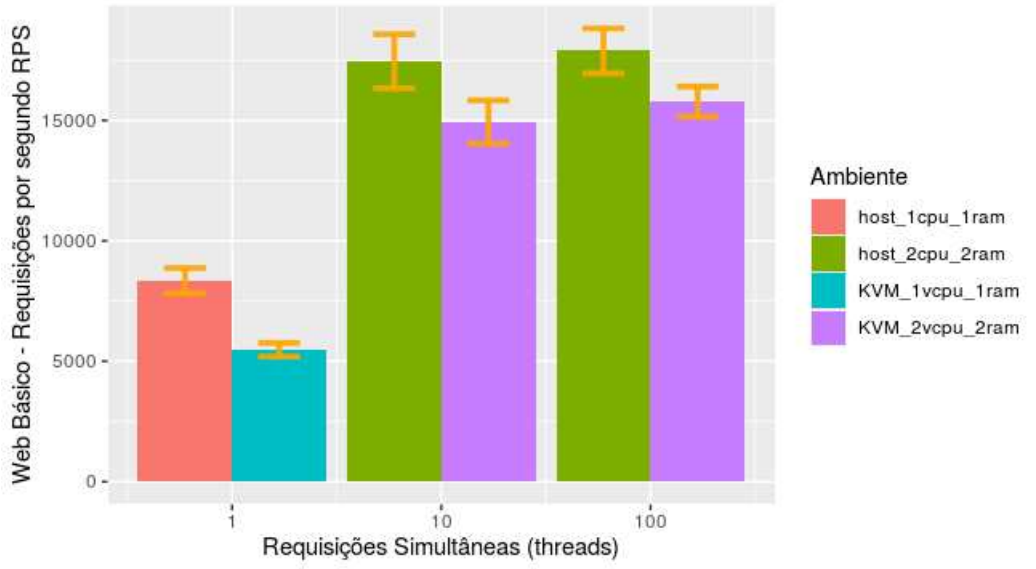

Figura 4.5: Máquina dedicada vs KVM no serviço Web básico.

Podemos observar na Figura 4.5 que, para a demanda de 1 requisição simultânea, houve uma diferença de 52% de desempenho a mais para a máquina dedicada em relação à virtual considerando que estão com a mesma quantidade de recursos. Isso significa que a recomendação para a demanda de 1 requisição no serviço web básico não foi uma boa recomendação, visto que uma diferença de 52% no desempenho é muito impactante na experiência do serviço. Já para a demanda de 10 requisições simultâneas, houve uma diferença de 16% a mais para a máquina dedicada em relação à virtual considerando a mesma quantidade de recurso em ambos testes. Podemos notar que a recomendação foi interessante devido a diferença aceitável de 16% comparada às vantagens que um ambiente virtualizado pode proporcionar. Por fim, temos os resultados com a demanda de 100 requisições simultâneas. Podemos observar que a diferença foi de 13% a mais para a máquina dedicada e esse resultado significa que a recomendação foi interessante, novamente, considerando as vantagens que um ambiente virtualizado pode trazer para a gerência de uma infraestrutura.

Posteriormente foram executados os *benchmarks* para o serviço de banco de dados como apresentado na Figura 4.6 comparando com os resultados obtidos na máquina dedicada com a máquina virtual sugerida, ambas com mesma quantidade de recursos. As recomendações sugeridas para as 3 demandas apresentadas foi o *flavor* com menos recursos contendo 1 vcpu e 1GB de RAM.

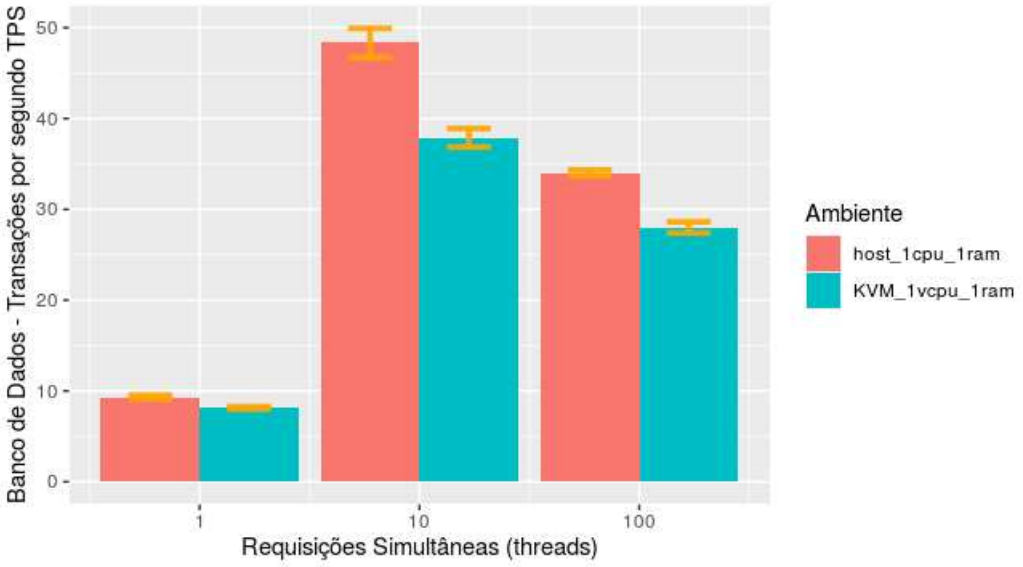

Figura 4.6: Máquina dedicada vs KVM em Banco de Dados.

Analisando a Figura 4.6, Para a demanda de 1 requisição simultânea, o desempenho apresentado nessa demanda foi 14% melhor na máquina dedicada em relação a *flavor* com menos recursos. Com essa diferença de 14% podemos considerar que a sugestão foi interessante para a demanda de 1 requisição simultânea, visto que esse *overhead* gerado pela virtualização não impacta fortemente no desempenho e consegue aproveitar as vantagens do ambiente virtualizado. Para a demanda de 10 requisições simultâneas, a sugestão de *flavor* foi a mesma, e o desempenho apresentado foi 27% melhor para a máquina dedicada. Essa diferença de 27% significa que a recomendação não foi ideal. Possivelmente essa recomendação pode não atender as necessidades do usuário. Para a demanda de 100 requisições simultâneas, o desempenho da máquina dedicada foi superior 21% em relação ao *flavor* sugerido de 1 vcpu e 1GB de RAM, essa recomendação também pode não ser ideal dependendo da preferência do usuário.Nos testes do serviço de banco de dados, a concorrência por recursos de escrita e leitura de disco limitaram bastante o desempenho alcançado com o aumento de recursos da máquina virtual e também da máquina dedicada, fazendo com que esses valores de desempenho não apresentassem grandes melhorias. Isso resultou em uma decisão que considerou mais a grande diferença no custo e optou por manter somente o *flavor* com custo menor, mais significativo para o processo de decisão.

O último serviço a ser considerado é o do web avançado. Na Figura 4.7, temos os resul-

tados da execução do *benchmark*, comparando os resultados obtidos na máquina dedicada com a máquina virtual sugerida na seção anterior, ambas com mesma quantidade de recursos. Considerando a sugestão de máquinas virtuais, a Figura 4.7 apresenta comparações para 1 requisição simultânea entre o *flavor* de menor quantidade de recursos e a máquina dedicada com os mesmos 1 Cpu e 1Gb RAM. Já para as demandas de 10 e 100 requisições simultâneas, o *flavor* sugerido foi o com mais recursos contendo 4 vcpu e 4GB de RAM,o qual foi comparado também com a máquina dedicada contendo a mesma quantidade de recursos.

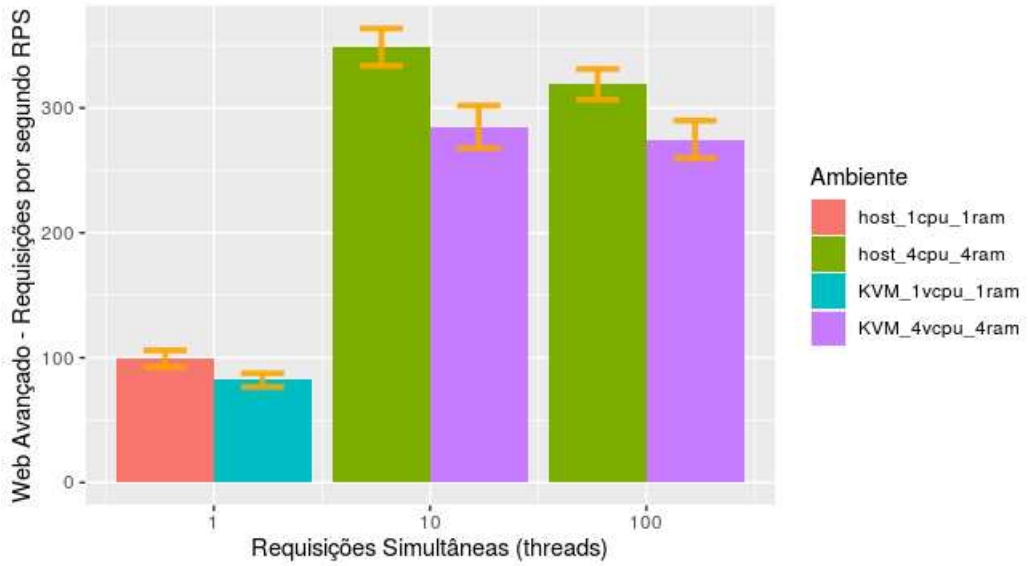

Figura 4.7: Máquina dedicada vs KVM em Web Avançado.

Observando a Figura 4.7 para 1 requisição simultânea, o servidor dedicado apresentou 20% mais requisições por segundo do que a máquina virtual sugerida . Essa diferença de desempenho na faixa dos 20% pode ser aceitável para o decisor se o foco dele não for somente o critério de desempenho. Já para 10 requisições simultâneas, temos 22% de desempenho a mais na máquina dedicada em relação à virtual com flavor sugerido de 4 vcpu e 4GB de RAM. Essa diferença é similar a apresentada em uma requisição simultânea e também pode ser aceitável considerando o mesmo caso. Se o decisor prioriza outras variáveis e não somente o desempenho pode ser uma alternativa viável. Por fim, na demanda de 100 requisições simultâneas, observamos 16% mais requisições por segundo para o servidor dedicado em relação à máquina virtual sugerida. Essa diferença apresenta uma alternativa viável,

visto que um comprometimento de 16% no desempenho em relação aos benefícios que um ambiente virtualizado pode entregar é um valor aceitável, considerando às preferências do decisor.

Logo, podemos concluir a partir da validação do trabalho, que o mecanismo de sugestão pode ser considerado um mecanismo viável para auxílio do administrador de sistemas na migração de um ambiente de servidores dedicados para um ambiente virtualizado, considerando que o mecanismo entregou uma comparação satisfatória na maior parte das recomendações sugeridas.

Um ponto importante a ser ressaltado é que o resultado da recomendação pode ser influenciado de diferentes maneiras, positivamente ou negativamente de acordo com as ponderações escolhidas pelo decisor, tanto na parte da escolha do hipervisor quanto na parte da decisão de dimensão da máquina virtual. Portanto, os resultados obtidos na comparação do ambiente virtualizado com o ambiente real podem mudar de acordo com esses aspectos.

Capítulo 5

Conclusão

Este trabalho se ateve ao objetivo de avaliar diferentes ambientes de virtualização comumente utilizados seguindo critérios da norma ISO/IEC 25010 e, com os resultados dessa avaliação, utilizar dois métodos de tomada de decisão para auxiliar a migração de uma infraestrutura de servidores dedicados para um ambiente virtualizado.

O trabalho contribuiu para obtermos uma melhor análise sobre os principais hipervisores do mercado, considerando tanto métricas objetivas quanto subjetivas. Adicionalmente o administrador de sistemas pode expressar suas prioridades em relação aos ambientes de virtualização, não somente comparando resultados de desempenho como em grande parte dos trabalhos que avaliam hipevisores, mas também considerando as métricas subjetivas da norma ISO/IEC 25010, possibilitando uma recomendação que melhor se adéqua às necessidades do ambiente virtualizado.

A validação do trabalho proporcionou uma visão de como seria a execução do procedimento apresentado, primeiramente considerando a tabela da escala de Saaty com as métricas da ISO 25010 para decisão do hipervisor utilizando o método AHP. No segundo momento, foram realizados testes de *benchmark* de serviços de rede mais utilizados em ambientes virtualizados, considerando configurações diferentes de demanda para que a matriz de decisão do método TOPSIS pudesse ser montada com valores reais de um ambiente de testes, para auxiliar na decisão da dimensão da máquina virtual.

Apesar dos *benchmarks* de desempenho apresentarem valores satisfatórios em relação ao servidor dedicado para o hipervisor KVM, os *benchmarks* de serviços de rede apresentaram variações significativas de desempenho, mostrando que existe diferença entre o *overhead*

que o hipervisor apresenta ao usar recursos do servidor dedicado e o *overhead* na execução de serviços. Também foi possível concluir que as recomendações apresentadas de hipervisor e dimensão de máquina virtual foram viáveis em sua maior parte, considerando a ponderação utilizada nos métodos de decisão pelo decisor.

5.1 Trabalhos Futuros

Após os resultados obtidos na validação deste trabalho com a análise de ambientes de virtualização e auxílio na tomada de decisão, nos limitamos a considerar apenas 3 métricas de desempenho, 2 métodos de tomada de decisão, 3 tipos de serviços de rede e 3 configurações de demanda dos serviços. Algumas dessas limitações ocorreram por falta de recursos de hardware disponíveis no momento do trabalho. Outra limitação do trabalho é o fato de necessitar que o administrador de sistemas informe valores de demanda de sua infraestrutura atual, o que torna o trabalho menos interessante. Logo, elencamos alguns possíveis trabalhos futuros:

- Adicionar mais métricas de desempenho, como Input/Output (I/O) e Vazão de Rede.
- Considerar o uso de outros métodos de decisão, ou uma composição de dois ou mais métodos.
- Adicionar outros tipos de serviço de rede nos testes, como servidor de e-mail e servidor de arquivos.
- Coletar valores de demanda, e tipos de serviços executados em um ambiente real de rede.

Referências Bibliográficas

[1] ISO/IEC 25010, howpublished = https://iso25000.com/index.php/en/iso-25000-standards/iso-25010, note = Acessado em: 2021-03-15.

[2] ISO/IEC 25010:2011, howpublished = https://www.iso.org/standard/35733.html, note = Acessado em: 2021-03-15.

[3] PostMark, howpublished = https://openbenchmarking.org/test/pts/postmark, note = Acessado em: 2021-03-15.

[4] RAMspeed SMP, howpublished = https://openbenchmarking.org/test/pts/ramspeed, note = Acessado em: 2021-03-15.

[5] Timed linux kernel compilation. https://openbenchmarking.org/test/pts/build-linux-kernel. Acessado em: 2021-03-15.

[6] Omnia AbdElRahem, Ayman M Bahaa-Eldin, and Ayman Taha. Virtualization security: A survey. In *2016 11th International Conference on Computer Engineering & Systems (ICCES)*, pages 32–40. IEEE, 2016.

[7] Keith Adams and Ole Agesen. A comparison of software and hardware techniques for x86 virtualization. *ACM Sigplan Notices*, 41(11):2–13, 2006.

[8] Lysandra Golhath Knopp Alves, Thiago Pimentel Nykiel, and Mischel Carmen Neyra Belderrain. Comparação analítica entre métodos de apoio multicritério à decisão (amd). *Anais do 13O Encontro de Iniciação Científica e Pós-Graduação do ITA–XIII ENCITA/2007. Instituto Tecnológico de Aeronáutica, São José dos Campos, SP, Brasil, Outubro*, 1, 2007.

[9] Emilly Shelly Gonçalves Alvim, Isabel Elisa dos Santos, Larissa Gomes Sena, Rodrigo Randow de Freitas, and Wellington Gonçalves. Modelo de apoio à tomada de decisão para seleção de fornecedores por meio do analytic hierarchy process (ahp). In *V Congresso Brasileiro de Engenharia de Produção*, 2015.

[10] Claude Banville, Maurice Landry, Jean-Marc Martel, and Christèle Boulaire. A stakeholder approach to mcda. *Systems Research and Behavioral Science: The Official Journal of the International Federation for Systems Research*, 15(1):15–32, 1998.

[11] Paul Barham, Boris Dragovic, Keir Fraser, Steven Hand, Tim Harris, Alex Ho, Rolf Neugebauer, Ian Pratt, and Andrew Warfield. Xen and the art of virtualization. *ACM SIGOPS operating systems review*, 37(5):164–177, 2003.

[12] Raja Benali, Hana Teyeb, Ali Balma, Samir Tata, and Nejib Ben Hadj-Alouane. Evaluation of traffic-aware vm placement policies in distributed cloud using cloudsim. In *2016 IEEE 25th International Conference on Enabling Technologies: Infrastructure for Collaborative Enterprises (WETICE)*, pages 95–100. IEEE, 2016.

[13] David Bernstein. Containers and cloud: From lxc to docker to kubernetes. *IEEE Cloud Computing*, 1(3):81–84, 2014.

[14] Petrônio Bezerra, Gustavo Martins, Reinaldo Gomes, Fellype Cavalcante, and Anderson Costa. Evaluating live virtual machine migration overhead on client's application perspective. In *2017 International Conference on Information Networking (ICOIN)*, pages 503–508. IEEE, 2017.

[15] Maria Betania Aparecida Campos. Métodos multicritérios que envolvem a tomada de decisão. 2011.

[16] Alexandre Carissimi. Virtualização: da teoria a soluções. *Minicursos do Simpósio Brasileiro de Redes de Computadores–SBRC*, 2008:173–207, 2008.

[17] Andrea De Montis, Pasquale De Toro, Bert Droste-Franke, Ines Omann, and Sigrid Stagl. Criteria for quality assessment of mcda methods. In *3rd Biennial Conference of the European Society for Ecological Economics, Vienna*, pages 3–6. Citeseer, 2000.

[18] Leandro Peçanha De Souza, Carlos Francisco Simões Gomes, and Alexandre Pinheiro De Barros. Implementation of new hybrid ahp–topsis-2n method in sorting and prioritizing of an it capex project portfolio. *International Journal of Information Technology & Decision Making*, 17(04):977–1005, 2018.

[19] Borislav Djordjevic, Valentina Timcenko, Nenad Kraljevic, and Nemanja Macek. File system performance comparison in full hardware virtualization with esxi, kvm, hyper-v and xen hypervisors. *Advances in Electrical and Computer Engineering*, 21(1):11–20, 2021.

[20] Michael Eder. Hypervisor-vs. container-based virtualization. *Future Internet (FI) and Innovative Internet Technologies and Mobile Communications (IITM)*, 1, 2016.

[21] Adrian Farrel. *A INTERNET e seus protocolos: uma análise comparativa*, volume 1. Elsevier Brasil, 2013.

[22] Natalia Castro Fernandes and OCMB Duarte. Provendo isolamento e qualidade de serviço em redes virtuais. *XXIX Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuıdos-SBRC*, pages 1–14, 2011.

[23] Antonios Gkortzis, Stamatia Rizou, and Diomidis Spinellis. An empirical analysis of vulnerabilities in virtualization technologies. In *2016 IEEE International Conference on Cloud Computing Technology and Science (CloudCom)*, pages 533–538. IEEE, 2016.

[24] Robert P Goldberg. Survey of virtual machine research. *Computer*, 7(6):34–45, 1974.

[25] LFAM Gomes, Carlos Francisco Simões GOMES, and AT de ALMEIDA. Tomada de decisão gerencial: enfoque multicritério. 2ª edição. *São Paulo: Atlas*, 2006.

[26] Luiz Flavio Autran Monteiro Gomes, Marcela Cecilia Araya González, and Claudia Carignano. *Tomada de decisões em cenários complexos: introdução aos métodos discretos do apoio multicritério à decisão*. Thomson, 2004.

[27] Fernando Ribeiro Guglielmetti, Fernando Augusto Silva Marins, and Valério Antonio Pamplona Salomon. Comparação teórica entre métodos de auxílio à tomada de

decisão por múltiplos critérios. *Encontro Nacional de Engenharia de Produção*, 23, 2003.

[28] Nelson HEIN, Larissa DEGENHART, Mara VOGT, Adriana KROENKE, and Ivan Marcos CAMPESTRINI. Método topsis na avaliação das empresas listadas no ibrx-100: uma avaliação multicritério dos impactos ambientais. In *V Congresso Brasileiro de Engenharia de Produção*, 2015.

[29] Khoa Huynh and Stefan Hajnoczi. Kvm/qemu storage stack performance discussion. In *Linux Plumbers Conference*, 2010.

[30] I IEC. Iso/iec 25010: 2011-systems and software engineering—systems and software quality requirements and evaluation (square)—system and software quality models, 2011.

[31] Stamatis Karnouskos, Roopak Sinha, Paulo Leitão, Luis Ribeiro, and Thomas I Strasser. Assessing the integration of software agents and industrial automation systems with iso/iec 25010. In *2018 IEEE 16th International Conference on Industrial Informatics (INDIN)*, pages 61–66. IEEE, 2018.

[32] Deepak Kumar and Amaizo Folly Felix Magloire. Hypervisor based performance characterization: Xen/kvm. In *2017 2nd International Conference on Telecommunication and Networks (TEL-NET)*, pages 1–4. IEEE, 2017.

[33] David Marshall. Understanding full virtualization, paravirtualization, and hardware assist. *VMWare White Paper*, 17:725, 2007.

[34] Anum Masood, Muhammad Sharif, Mussarat Yasmin, and Mudassar Raza. Virtualization tools and techniques: Survey. *Nepal Journal of Science and Technology*, 15(2):141–150, 2014.

[35] Microsoft. Azure cloud. https://azure.microsoft.com/pt-br/. Acessado em: 2021-07-15.

[36] Microsoft. Azure cloud pricing. https://azure.microsoft.com/pt-br/pricing/calculator. Acessado em: 2021-07-15.

[37] Yoshio Moritoh and Yoshiro Imai. Development of an integrated information server system for it education through server virtualization technology. *New Computer Architectures and their Applications*, 5(1):39–47, 2015.

[38] Robayet Nasim and Andreas J Kassler. Deploying openstack: Virtual infrastructure or dedicated hardware. In *2014 IEEE 38th International Computer Software and Applications Conference Workshops*, pages 84–89. IEEE, 2014.

[39] Ramon Rodriguez. Models, methods, concepts and applications of the analytic hierarchy process. *Interfaces*, 32(6):93, 2002.

[40] Mendel Rosenblum and Tal Garfinkel. Virtual machine monitors: Current technology and future trends. *Computer*, 38(5):39–47, 2005.

[41] Thomas L Saaty. The analytic hierarchy process (ahp). *The Journal of the Operational Research Society*, 41(11):1073–1076, 1980.

[42] Gustavo AA Santana. *Data center virtualization fundamentals: understanding techniques and designs for highly efficient data centers with Cisco Nexus, UCS, MDS, and beyond*. Cisco Press, 2013.

[43] Kardi Teknomo. Analytic hierarchy process (ahp) tutorial, 2006.

[44] Gwo-Hshiung Tzeng and Jih-Jeng Huang. *Multiple attribute decision making: methods and applications*. CRC press, 2011.

[45] Rudolf Vetschera and Heinz Walterscheid. A process-oriented framework for the evaluation of managerial support systems. *Information & management*, 28(3):197–211, 1995.

[46] Dejana T Vojnak, Borislav S orđević, Valentina V Timčenko, and Svetlana M Štrbac. Performance comparison of the type-2 hypervisor virtualbox and vmware workstation. In *2019 27th Telecommunications Forum (TELFOR)*, pages 1–4. IEEE, 2019.

[47] William Von Hagen. *Professional xen virtualization*. John Wiley & Sons, 2008.

[48] Yoji Yamato. Openstack hypervisor, container and baremetal servers performance comparison. *IEICE Communications Express*, 4(7):228–232, 2015.

[49] Jung Ji Young, Shin Hee Kyoung, Minwoo Park, and Yongtae Shin. Cloud computing transformation*: Considering adoption of mixed security. In *2021 21st ACIS International Winter Conference on Software Engineering, Artificial Intelligence, Networking and Parallel/Distributed Computing (SNPD-Winter)*, pages 90–95. IEEE, 2021.