INSTITUTO ENSINAR BRASIL

REDE DE ENSINO DOCTUM

Jhonatas Junnior Oliveira da Silva [1]

Romulo Henrique de Oliveira [2]

ESTUDO COMPARATIVO LABORATORIAL ENTRE MÁQUINA VIRTUAL E CONTAINER DOCKER PARA APLICAÇÃO WEB

COMPARATIVE LABORATORY STUDY BETWEEN VIRTUAL MACHINE AND CONTAINER DOCKER FOR WEB APPLICATION

Trabalho de Conclusão de Curso

Ipatinga MG

2022

[1](UNIDOCTUM) – aluno.jhonatas.silva@doctum.edu.br

[2](UNIDOCTUM) – aluno.romulo.oliveira@doctum.edu.br

Jhonatas Junnior Oliveira da Silva
Romulo Henrique de Oliveira

ESTUDO COMPARATIVO LABORATORIAL ENTRE MÁQUINA VIRTUAL E CONTAINER DOCKER PARA APLICAÇÃO WEB

COMPARATIVE LABORATORY STUDY BETWEEN VIRTUAL MACHINE AND CONTAINER DOCKER FOR WEB APPLICATION

Trabalho de Conclusão de Curso apresentado à Faculdade Doctum, como parte dos requisitos necessários à obtenção do título de Bacharel de Sistema de Informação.

Orientador: Luiz Fernando Alves Souza.

Ipatinga MG
2022

RESUMO

Este artigo, a fim de analisar o desempenho da virtualização, apresenta um estudo comparativo utilizando uma aplicação Web implantada em dois serviços de virtualização: a máquina virtual e o container Docker. Foram aplicados três níveis de cargas de requisições para o aumento do consumo de processamento e memória RAM. Os resultados mostraram um melhor aproveitamento do container Docker em relação à máquina virtual. Sendo assim, o Docker pode ser uma melhor escolha em implementações em nuvem.

Palavras-chave: Máquina Virtual, Container, Docker, Virtualização, Estudo comparativo.

ABSTRACT

This article, in order to analyze the performance of virtualization, presents a comparative study using a Web application deployed in two virtualization services: the virtual machine and the Docker container. Three levels of load requests were applied to increase the consumption of processing and RAM memory. The results showed a better use of the Docker container in relation to the virtual machine. As such, Docker might be a better choice in cloud deployments.

Keywords: Virtual Machine, Container, Docker, Virtualization, Comparative study.

SUMÁRIO

LISTA DE TERMOS E SIGLAS

API - Application Programming Interface - interface de programação de aplicação
CPU - Central Process Unit **-** Unidade central de processamento
Data centers - Centro de processamento de dados
Host - Máquina ou computador conectado a uma rede, podendo oferecer informações, recursos, serviços.
HTTP - Hypertext Transfer Protocol - Protocolo de Transferência de Hipertexto
Hypervisor - Processo que cria e executa máquinas virtuais
Kernel - Núcleo do sistema
LXC - Linux Containers – Container Linux
RAM - Random Access Memory - Memória de Acesso Aleatório
SQL - Structure Query Language – Linguagem de Consulta Estruturada
SO - Sistema operacional
Thread - Processo que divide tarefas a fim de executá-las de forma simultânea ou sequencialmente
VM - Virtual Machine - Máquina Virtual

1. INTRODUÇÃO

Atualmente, empresas dos mais diversos segmentos buscam inovações e destaque no mercado. Atentando para o fato de que a tecnologia da informação está envolvida na maioria dos projetos que visam quaisquer das pautas citadas anteriormente, a adesão da virtualização à infraestrutura de TI vem se tornando cada dia maior.

Hoje, existem diversas soluções no mercado para implementar a virtualização. Entretanto, algumas delas sofrem problemas quanto a sobrecarga de recursos computacionais. Tendo em vista essa situação, surge a ideia deste artigo de comparar duas das diversas soluções que promovem a virtualização. Assim, empregamos em um conceito de implementação em nuvem, a fim de buscar qual tem o melhor desempenho no quesito de consumo de recursos.

A proposta deste artigo é apresentar uma análise comparativa, aplicando dois conceitos de virtualização: containers Docker e máquina virtual. Além disso, expor os resultados estatísticos do comportamento de diferentes situações de cargas de trabalho realizadas sobre as ferramentas. Nesse sentido, exploramos o uso de CPU e memória RAM dos ambientes, um contendo uma aplicação web conteinerizada pelo Docker e o outro a mesma aplicação virtualizada pela máquina virtual. Por meio deste estudo comparativo laboratorial, acredita-se que os resultados podem auxiliar profissionais na tomada de decisão da melhor solução de virtualização para aplicações web.

2. REFERENCIAL TEÓRICO

Segundo Mattos (2008), nos anos 60, a IBM começou a desenvolver a primeira máquina virtual, que permitia que um único computador fosse dividido em vários. Com essa máquina virtual, surgiu o conceito da virtualização. Goldberg (1972) afirma que, nessa época, a tendência era que cada usuário tivesse um ambiente mono usuário completo, independente e desvinculado de ambientes de outros usuários.

Tanenbaum (2016) descreve as vantagens da utilização de máquinas virtuais. Além do isolamento, a redução de equipamentos proporciona a economia de dinheiro com eletricidade e com hardware, além de menos espaço utilizado.

Os containers auxiliam no tempo de configuração e preparação de um ambiente. Além disso, permitem o seu compartilhamento no formato de imagens, fornecem padronização e auxiliam na redução de conflitos entre as equipes que executam o mesmo software em uma infraestrutura diferente. Assim, facilitando também a transição entre os ambientes de implantação de um software. (GALDINO; RONDINA, 2020).

2.1 Virtualização

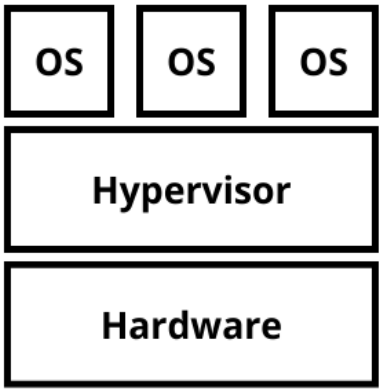
A virtualização é uma combinação de engenharia de hardware e software que cria máquinas virtuais (VM) e permite que vários sistemas operacionais sejam executados na mesma plataforma física.

A ideia da virtualização surgiu em meados de 1960, quando os gigantes e caros computadores existentes da época já atingiam uma grande velocidade de processamento. Porém, eles se mostravam ineficientes quanto ao seu caro tempo de cálculo, devido ao seu gerenciamento de processos ser feito de forma manual. A fim de aproveitar o processamento computacional, a equipe responsável decidiu que era necessário executar vários processos paralelos. Com o surgimento desse conceito de tempo compartilhado (time sharing), originou-se a ideia de virtualização.

Nos anos 70, o conceito da virtualização se firmou por meio do cientista Robert P. Golderbeg. Ele lançou a base teórica da arquitetura para sistemas computacionais virtuais em sua dissertação (GOLDBERG, 1972), expondo o quão este conceito poderia ser importante para o desenvolvimento de novas arquiteturas e novas tecnologias. No mesmo ano a IBM lançou um mainframe capaz de executar, simultaneamente, diferentes sistemas operacionais sob supervisão de um programa de controle chamado hypervisor.

Alecrim (2012) afirma que a virtualização se tornou extremamente importante para o mundo cada vez “mais digital”. Pode-se conceituar virtualização como sendo uma soluções computacionais que permite a execução de vários sistemas operacionais e seus respectivos softwares, a partir de uma única máquina. Sendo um desktop convencional ou um potente servidor, a virtualização maximiza a utilização dos recursos.

Figura 1: Arquitetura da virtualização.



Fonte: Autores (2022).

De acordo com Machado (2007), a virtualização é a simulação de um hardware/software que roda sobre outro software. O conceito de ambiente simulado é chamado de máquina virtual (VM).

A virtualização veio para mudar o modo de pensar sobre recursos. Com a virtualização, você não se limita a executar apenas um sistema operacional em um servidor. É possível consolidar vários sistemas operacionais e aplicativos, o que propicia a simplificação de data centers, uso mais eficiente e redução de custos.

2.1.1 Categorias de Virtualização

Segundo Figueredo e De Lucca (2017), a virtualização possui três grandes categorias, sendo a primeira de Nível de Hardware. Trata-se de quando a virtualização é colocada diretamente na máquina física e provê às camadas superiores um hardware abstrato, semelhante ao original.

A segunda categorização é a Nível de Sistema Operacional. Nela, a camada de virtualização é adicionada entre o sistema operacional e as aplicações. As partições lógicas são vistas como máquinas isoladas, compartilhando o mesmo sistema operacional (FIGUEREDO; DE LUCCA, 2017).

A terceira categoria é conhecida como a de Nível de aplicação. Aqui, a camada de virtualização é inserida como uma aplicação dentro de um sistema operacional.

2.1.2 Tipos de Virtualização

De acordo com Figueredo e De Lucca (2017), existem três tipos de virtualização: servidores, desktop e aplicação, respetivamente.

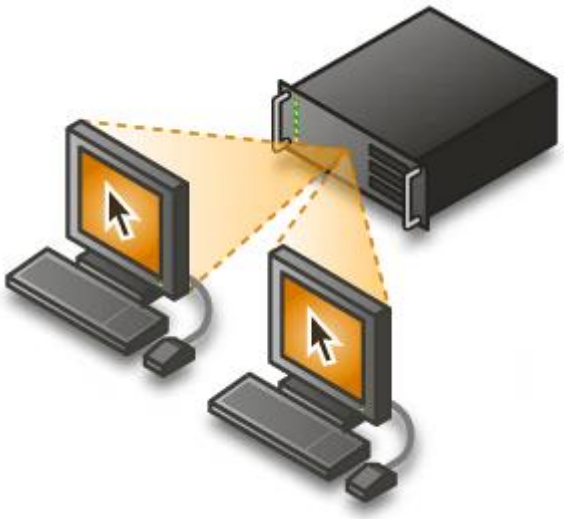
A virtualização de servidores (hypervisor) permite que vários servidores virtuais sejam executados em um servidor físico. Cada um deles é isolado e independente.

Figura 2: Representação da virtualização de servidores.

Fonte: Rocha (2013).

A virtualização de desktops segue o mesmo conceito da virtualização de servidore. Entretanto, os aplicativos e sistemas operacionais são executados de forma centralizada.

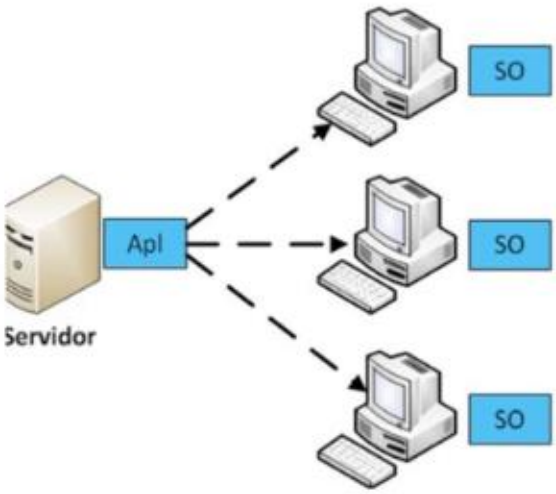
Figura 3: Representação de virtualização de desktops.



Fonte: Macedo (2012).

O último tipo é a virtualização de aplicação. Ele permite que as aplicações sejam executadas, sem a necessidade de instalação no dispositivo. Logo, elas são centralizadas e não têm a necessidade de atualização ou manutenção (FIGUEREDO; DE LUCCA, 2017).

Figura 4: Representação da virtualização de aplicações, em que a aplicação é executada em uma máquina e os outros usuários devem se conectar ao servidor onde ela se encontra para ser utilizada.



Fonte: Rocha (2013).

2.1.3 Máquinas virtuais

Uma máquina virtual é a representação virtual de um servidor físico, composta por um SO, drives, bibliotecas e aplicativos. Cada máquina virtual se constitui em um ambiente de funcionamento independente, que se comporta como se fosse um computador separado.

Segundo Ghimiray (2022), uma máquina virtual é um computador que é executado inteiramente em software, em vez de um hardware físico. Ela utiliza uma emulação através de um hypervisor, que imita um conjunto completo de hardware, como CPU, memória, placa de rede etc. Quando uma máquina virtual é iniciada, certa quantidade de capacidade do processador e demais periféricos de seu hardware é distribuído automaticamente pela camada de virtualização ou hypervisor.

Os hypervisors são diretamente responsáveis pela hospedagem e administração das máquinas virtuais no host ou no servidor físico. Além disso, oferecem uma visão uniforme do hardware adjacente, que significa que eles podem operar em hardware de diferentes fornecedores. Os hypervisors são o componente de software que executa a máquina virtual e gerencia a camada entre as máquinas virtuais e os recursos físicos disponíveis para a virtualização. Descrevendo seu conceito de forma abreviada, a VM é um computador simulado dentro de um real.

2.2 Container

É um método que consiste em isolar processos e aplicações, permitindo elas trabalharem individualmente sem conflitos com outras aplicações além de facilitar a portabilidade em diferentes ambientes. Ele contém um conjunto de processos que são executados a partir de uma imagem, essa que fornece todas as configurações necessárias para o processo.

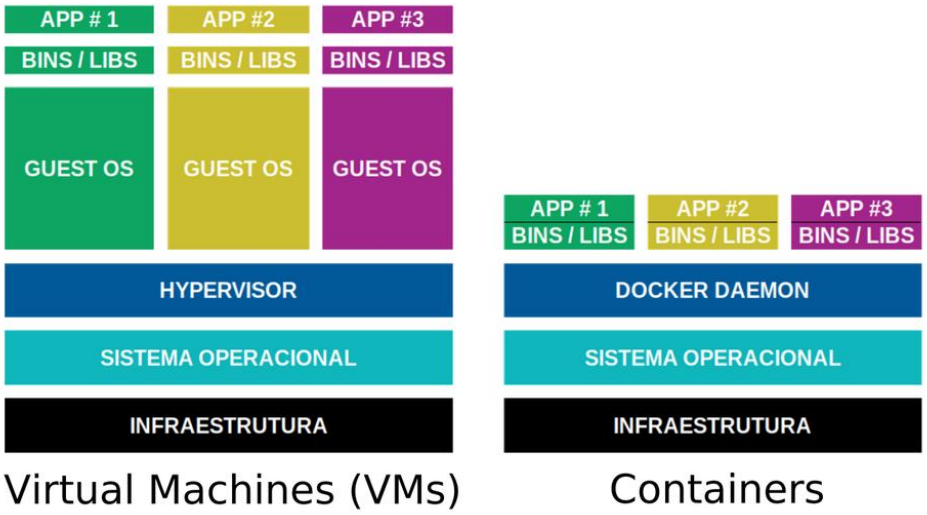
Conteinerização trata-se de um empacotamento de requisitos de uma aplicação em desenvolvimento, transformando-a em forma de uma imagem base. Essa imagem mesma pode ser executada em um espaço isolado (container) em diferentes sistemas. Ela combina a aplicação, suas dependências e as bibliotecas do sistema, organizadas em um “pacote”. Diferente da tecnologia de virtualização em VM, os containers compartilham o mesmo sistema operacional em meio ao seu funcionamento.

Esse tipo de virtualização dispensa o uso de hypervisor, em que a camada que gerencia o container se conecta diretamente ao sistema operacional para, assim, chegar á camada de recursos físicos do host.

De acordo com Fell (2018), a vantagem de utilizar containers é que não é necessário existir um sistema operacional para virtualizar uma aplicação. Sendo assim, gera-se um ganho na economia de recursos computacionais se comparado aos meios tradicionais de virtualização.

Tanto máquinas virtuais quanto containers têm certas semelhanças no quesito de isolamento e recursos computacionais utilizados. Porém, quando se trata de suas arquiteturas, é visível suas diferenças.

Figura 5: A figura a seguir apresenta as duas infraestruturas.



Fonte: Delgado (2020).

A figura compara as arquiteturas da VM (Máquina virtual) e do Container usando Docker, é possível ver que em um container existem três aplicações sendo executadas sobre um mesmo SO (Sistema Operacional). Já na VM, cada aplicação depende de um SO para que possa ser executada. É possível notar que, com a conteinerização, o kernel é compartilhado entre as aplicações tornando-a mais leve em relação à VM.

Segundo Rubens (2015), containers e máquinas virtuais podem ser vistas como tecnologias complementares. Afinal, é possível utilizar containers em VM (Máquinas Virtuais), otimizando o consumo de recurso de hardware, além de permitir aumentar seu isolamento e segurança.

2.2.1 Docker

O Docker é uma plataforma que automatiza a implantação de aplicações dentro de ambientes isolados, denominados containers (Docker Inc., 2022). Seu principal objetivo é proporcionar múltiplos ambientes isolados dentro do mesmo servidor, mas acessíveis externamente via exposição das portas.

O container Docker foi criado pela empresa dotCloud. Seguindo os mesmos conceitos de computação existentes em containers Linux, a ferramenta fornece recursos que permitem criar e manter containers virtuais de forma isolada uns dos outros. O conceito de container na virtualização já existe há muitos anos. Porém, o Docker criou uma ferramenta que é maior que a soma de suas partes. Assim,

consegue fornecer um conjunto de ferramentas e APIs de forma conjunta, que permite gerenciar tecnologias em nível de kernel, como LXCs, cgroups e um sistema de arquivos copy-on-write. O resultado é um potencial divisor de águas para DevOps, administradores de sistema e desenvolvedores. (MERKEL, 2014).

2.2.1.1 Estrutura do Docker

O container Docker é dividido em duas camadas: a imagem e o container. Uma Imagem é um pacote independente e executável que inclui todos os recursos necessários para executar algum software. Isso inclui o código, bibliotecas, arquivos de configuração e variáveis de ambiente. Já o container é uma instância em execução de uma imagem, executada completamente isolada do ambiente hospedeiro por padrão. Ou seja, somente acessando os arquivos e portas que são configuradas para isso (GALDINO; RONDINA, 2020).

2.2.1.2 Docker Compose

É uma ferramenta complementar que define e executa aplicativos Docker de vários containers. Através do Compose, um arquivo YAML é utilizado para configurar os serviços dos containers. Depois do arquivo criado, com um comando (sudo docker compose up -d), todos os serviços são criados e iniciados. Com ele, é possível ter o controle de criar ou derrubar containers que estão sob seu controle tudo de uma vez.

2.3 MySQL

O MySQL é um sistema de código aberto que faz o gerenciamento de banco de dados baseado em modelos relacional. Seu funcionamento é do tipo cliente-servidor, com uma interface de fácil entendimento por parte do cliente. O serviço utiliza a linguagem SQL (Structure Query Language – Linguagem de Consulta Estruturada).

2.3.1 phpMyAdmin

De acordo com a documentação do phpMyAdmin, ele “é uma ferramenta de software livre escrita em PHP, destinada a administrar o MySQL pela Web”. (phpMyAdmin, 2022).

Ele funciona pelo navegador provendo uma interface de fácil compreensão. O

phpMyAdmin trabalha como uma ferramenta de gestão de banco de dados, permitindo aos usuários a capacidade de administrar e executar qualquer instrução sob o banco.

2.4 VirtualBox

VirtualBox é um virtualizador completo de uso geral para hardware x86, direcionado ao uso de servidor, desktop e incorporado (Virtualbox, 2022). Ele é um monitor de máquina virtual ou hypervisor que permite criar, gerenciar a camada entre as máquinas virtuais e os recursos físicos disponíveis para a virtualização e executar as VM'S. Disponível de forma gratuita pela empresa Oracle, ele é multiplataforma podendo ser executado em Windows, Mac OS X, Linux e Solaris.

2.5 JMeter

JMeter é um software criado pela Apache Softwares em 2007. Ele tem como objetivo testar performance de aplicações WEB através de requisições, simulando um grupo de usuários enviando solicitações para o servidor de destino configurado. Por ser de código aberto é possível implementar plugins que melhoram os testes realizados.

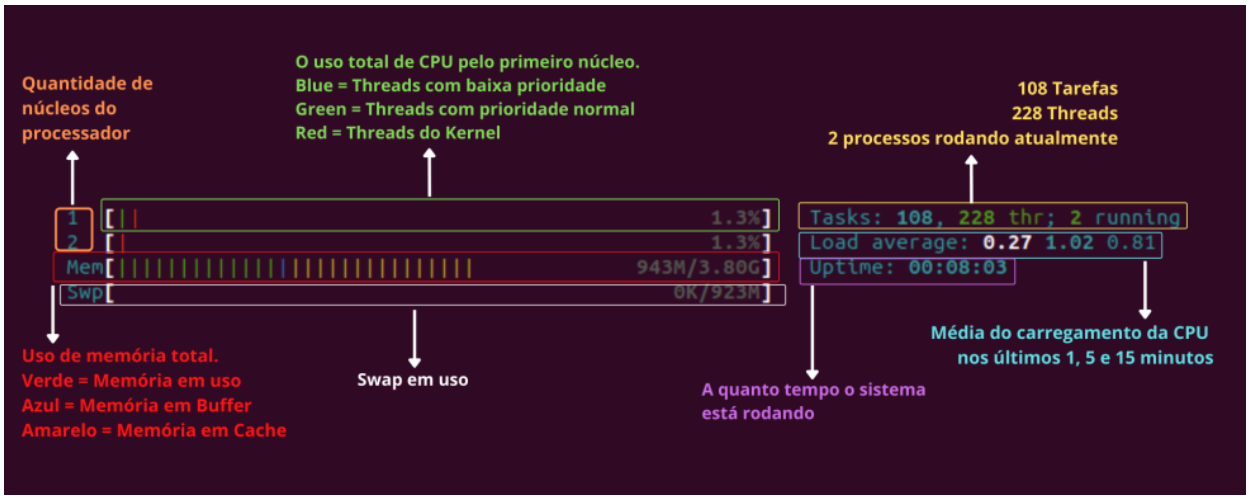
O Jmeter tem componentes que auxiliam na administração de scripts de testes e no controle dos cenários criados, além de permitir configurar ambientes de testes para executar em servidores.

2.6 HTOP

HTOP é um visualizador e gerenciador de processos interativo que serve para monitorar o desempenho do sistema em tempo real. Era uma ferramenta inicialmente exclusiva para Ubuntu Linux, porém a partir da versão 2.0 se tornou multiplataforma. Assim, além do Linux, FreeBSD, OpenBSD e MacOS.

Ele apresenta uma lista dos processos em execução de um computador, normalmente ordenada pela quantidade de uso da CPU.

Figura 6: Imagem do painel principal do HTOP, contendo um explicativo dos blocos de informação.



Fonte: Souza (2020).

3. METODOLOGIA

Esta seção apresentará a metodologia proposta que tem como objetivo avaliar o desempenho da máquina virtual e container. Os recursos analisados para a avaliação de desempenho serão: processador e memória RAM.

3.1 Especificações

Os sistemas escolhidos para a comparação de desempenho foram as máquinas virtuais e container, respectivamente VirtualBox e Docker. As especificações do ambiente de cada um estão apresentados na tabela a seguir:

Tabela 1

Identificação	Sistema	CPU	Memória	Disco	SO
Servidor 1	Docker	Intel Core i5-10300H	16 GB	50 GB	Ubuntu 22.04.1 LTS
Servidor 2	Virtual Box	Intel Core i5-10300H	16 GB	50 GB	Ubuntu 22.04.1 LTS

Fonte: Autores (2022).

Ambos ambientes não continham nenhuma aplicação diferente das padrões pré-instaladas pelo ubuntu em sua instalação nativa. O Docker foi executado

diretamente no sistema operacional do host físico, seu ambiente identificado como servidor 1 foi instalado a versão 20.10.18 (versão Docker). Já o docker compose, em sua versão 2.10.2. Estando com ambos instalados, foram criados 3 containers: o primeiro contendo arquivos da parte visual da aplicação com o php apache na versão 7.2; o segundo com MySQL em sua versão 8.0; e o terceiro container contendo o phpMyAdmin. Na configuração do docker compose expomos a aplicação para rodar de forma externa pela porta 8001.

Já no ambiente da VM, identificado como servidor 2, foi instalado o php apache em sua versão 7.2, o MySQL na versão 8.0 e o phpMyAdmin. Neste ambiente a aplicação também foi exposta para acesso externo através do IP da VM.

A aplicação WEB disponibilizada nos dois ambientes para o teste foi uma agenda telefônica desenvolvida em PHP, utilizando como base de dados o banco MySQL. A aplicação possibilita ao usuário cadastrar, editar, visualizar e remover contatos. Em sua tela principal, são listados todos os contatos salvos. Nos dois ambientes, foram cadastrados exatamente 100 contatos. A tela em específico utilizada nos testes de requisição foi a tela principal, listando os 100 contatos.

Para a realização dos testes de carga utilizando a ferramenta JMeter em sua versão 5.5, executamos três grupos de requisição na aplicação. Eles foram rodados em ambos ambientes: VM e containers em Docker:

Tabela 2

Grupos de teste	Usuários	Tipo de requisição	Método
1	1000	HTTP	GET
2	5000	HTTP	GET
3	10000	HTTP	GET

Fonte: Autores (2022).

Cada usuário especificado nos grupos de teste é uma thread que realiza uma requisição HTTP, configurada como método GET, método que realiza a busca na raiz da URL da aplicação. Sendo assim cada grupo é um simulado de diversos usuários acessando simultaneamente uma página.

Os testes de requisição do JMeter foram feitos em um computador conectado à mesma rede que a máquina host.

Com o plano de teste montado, cada grupo foi executado três vezes, os primeiros testes foram executados no ambiente dos containers Docker, em seguida no ambiente da VM. O desempenho dos ambientes, se comportando meio as requisições

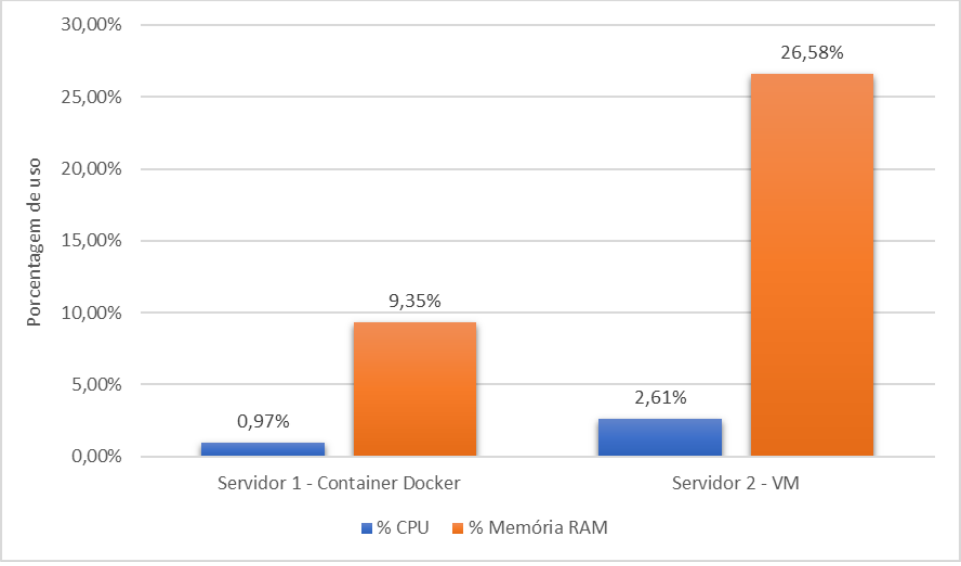
feitas pelo Jmeter, foi analisado e medido em tempo real pelo HTOP. Esse último monitorou de forma iterativa os recursos da máquina host.

Nos resultados fornecidos pelo HTOP, são expostos o consumo de cada núcleo separadamente e o valor total do consumo da memória RAM sob o valor disponível. Para obtenção dos resultados totais em porcentagem, no caso do processador, foi somado o consumo dos 8 núcleos do CPU e foi dividido pela quantidade total de núcleo: 8 (oito). Na memória RAM, seu consumo total foi multiplicado por 100 (cem) e o resultado dessa operação foi dividido pela quantidade total da memória disponível para uso: 15.50GB.

4 RESULTADOS E DICUSSÕES

Antes de iniciar os testes foram registradas as informações contidas no monitor de informações do HTOP, tanto da máquina rodando o cenário com o Docker quanto da mesma rodando o cenário com a VM. Todos sob as mesmas condições, ambos em modo ocioso. Ou seja, sem nenhum tipo de requisição sob seus determinados domínios.

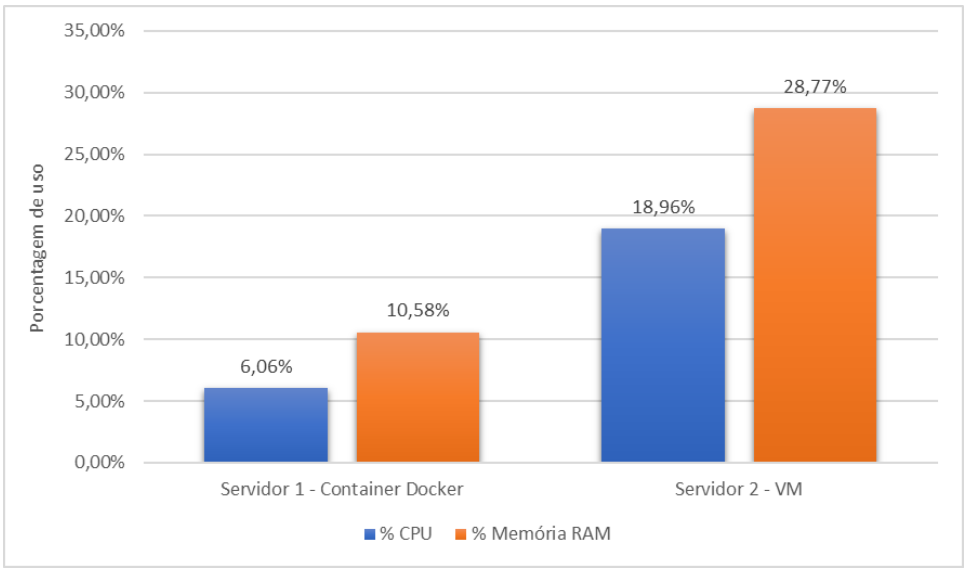
Figura 7 – Gráfico de relação dos ambientes container Docker e VM sem nenhuma requisição sob a aplicação (servidores ociosos).



Fonte: Autores (2022).

No servidor 1, o consumo de processamento foi de 0,97% e da memória RAM, do total de 15.50GB, foi consumido 1.45GB — equivalente a 9,35%. No servidor 2, o consumo de processamento foi 2,61% e da memória RAM foram consumidos 4.12GB, do total de 15.50GB — equivalente a 26,58%.

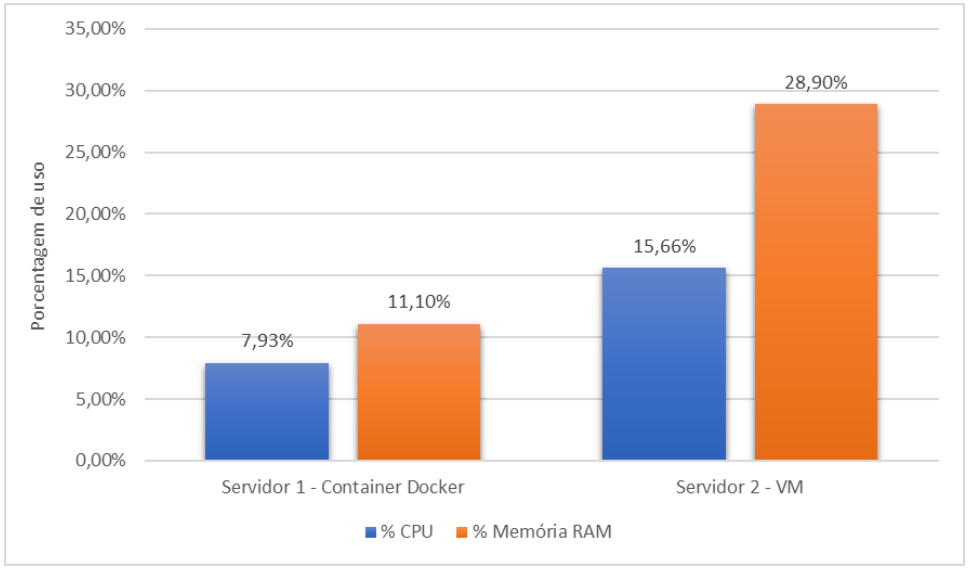
Figura 8 – Gráfico de relação dos ambientes sob o grupo de teste 1. com mil requisições.



Fonte: Autores (2022).

No servidor 1, o consumo de processamento foi de 6,06% e da memória RAM foi 1.64GB — equivalente a 10,58%. No servidor 2, o consumo de CPU foi de 18,96% e 4.46GB de memória RAM — equivalente a 28,77%.

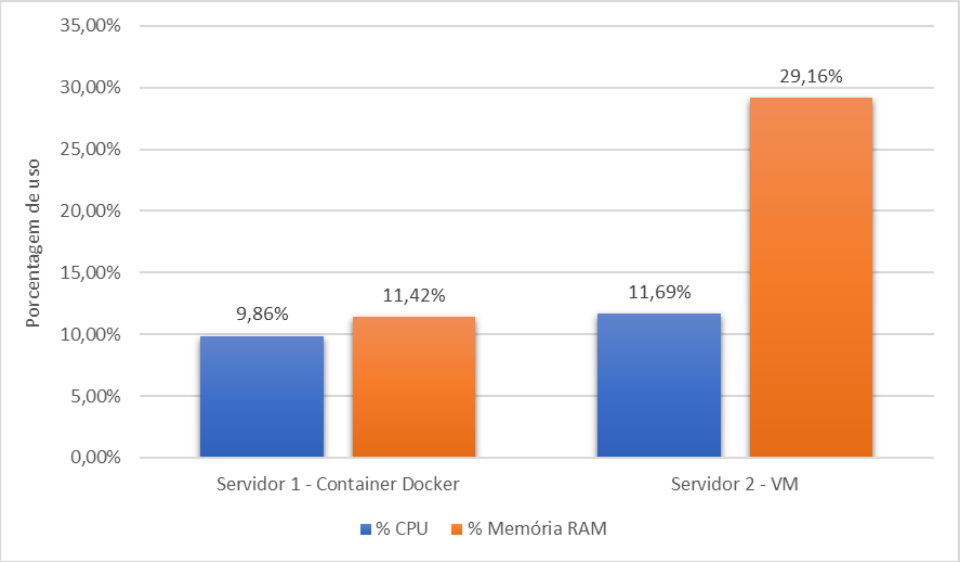
Figura 9 - Gráfico de relação dos ambientes sob o grupo de teste 2. com 5 mil requisições.



Fonte: Autores (2022).

O servidor 1 teve um consumo de CPU de 9,31% e 1.72GB de memória RAM, do total de 15.50GB — ou seja, 16,12% do total. No servidor 2, o consumo de processamento foi 10,2% e 4.48GB de memória RAM — 29,16%.

Figura 10 - Gráfico de relação dos ambientes sob o grupo de teste 3. com 10 mil requisições.



Fonte: Autores (2022).

No servidor 1 o consumo de CPU foi de 9,86% e 1.77GB de memória RAM, do total de 15.50GB — 11,42%. O servidor 2 teve um consumo de 11,69% de CPU e 4.52GB de memória RAM — 29,16% do total.

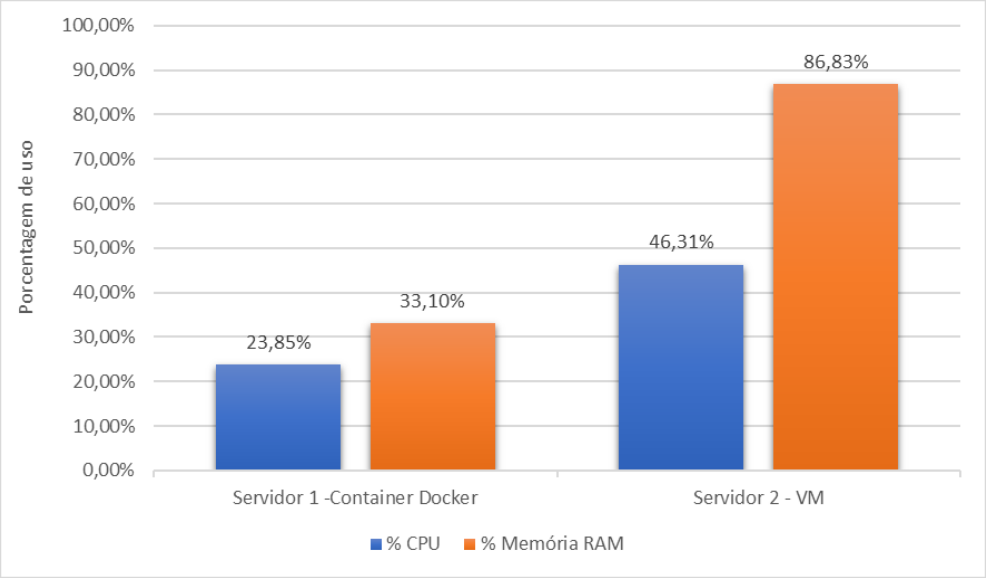
Após a execução dos grupos de teste, obtivemos os seguintes resultados:

O consumo de CPU do servidor 1 no primeiro grupo de teste, com mil requisições, se mostrou consideravelmente menor que o servidor 2. Ou seja, 6,06% e 18,96%, respectivamente. O consumo de memória RAM também foi menor, sendo 10,58% e 28,77%.

No segundo grupo de teste, com 5 mil requisições, o servidor 1 teve o consumo de CPU de 7,93% e o servidor 2 15,66%. Já o consumo da memória RAM do servidor 1 foi de 11,10% e 28,90% do servidor 2.

No terceiro teste, com 10 mil requisições, o consumo de CPU do servidor 1 foi 9,86% e o do servidor 2 foi de 11,69%. Em relação à memória RAM, o consumo do servidor 1 foi de 11,42% e o do servidor 2 foi de 29,16%.

Figura 10 - Gráfico de relação da soma total dos resultados obtidos sob os testes realizados nos dois ambientes, container Docker e máquina virtual.



Fonte: Autores (2022).

Com os resultados obtidos, foi feita uma soma de todas as apurações de consumo de CPU do servidor 1 (container Docker) e do servidor 2 (VM), bem como também do consumo de memória RAM.

Através dos resultados foi notado que o consumo tanto de CPU quanto de memória RAM do container Docker se saiu mais eficiente que da VM. Sendo assim, sob as condições propostas neste trabalho a utilização em container Docker acabou se saindo melhor que a VM.

5. CONCLUSÃO

Neste artigo, realizamos uma avaliação experimental, a fim de medir e comparar o desempenho de uma aplicação web quando implantada usando containers Docker e máquina virtual (VM). Assim, medimos o uso de CPU e memória RAM sob diferentes níveis de cargas feitas através do Jmeter.

Os resultados mostraram em geral que; nos 3 níveis de cargas aplicados nos cenários montados, os serviços da Web baseados em container são superiores aos serviços da Web baseados na VM, com relação as métricas medidas neste estudo quanto a desempenho.

O consumo de memória RAM da VM é maior que a configurada inicialmente. Portanto, a VM necessita de uma memória adicional para a virtualização (VMWare, 2021), enquanto os containers são a emulação no nível de abstração do sistema

operacional (SO). Segundo Minsu, Hwamin e Kiyeol (2019), existe um desperdício de recursos adicionais para a implantação da VM, mesmo que ela não esteja sendo utilizada.

Por meio do estudo deste artigo, concluímos que a implementação em nuvem utilizando containers Docker, é superior à VM, devido a seu baixo consumo de recursos e alto desempenho.

6. REFERÊNCIAS

ALECRIM, Emerson. O que é virtualização e para que serve?. **Infowester**. 25 ago. 2012. Disponível em: <http://www.infowester.com/virtualizacao.php>. Acesso em: 23 out. 2022.

CORREA, José Norberto. Disponibilização de aplicações ia containers de software em cluster de alto deempenho. **LIFTER** , Florianópolis, p. 1-93, 2016. Disponível em: <https://repositorio.ufsc.br/xmlui/bitstream/handle/123456789/171410/TCC_Lifter_JoseNorberto.pdf?sequence=1&isAllowed=y>. Acesso em: 1 out. 2022.

DELGADO, Caio. Containers ou Máquinas Virtuais - Quando usar cada um?. **Caio Delgado**. 20 abr de 2020. Disponível em: < https://caiodelgado.dev/container-vm/>. Acesso em: 24 set. 2022.

DOCKER INC. **Docker**, 2022. Disponível em: < https://www.docker.com/>. Acesso em: 9 out. 2022.

FELL, Gerson. **Análise de desempeno entre máquinas virtuais e containers para aplicações web**. Trabalho de Conclusão de Curso. Engenharia da Computação, da Universidade do Vale do Taquari, Lajeado, 2018. Disponível em: <https://www.univates.br/bduserver/api/core/bitstreams/eda4bece-04c0-408b-aa0e-68390ca3d70f/content>. Acesso em: 8 out. 2022.

FIGUEREDO, Luan Rubens Rodrigues; DE LUCCA, Gustavo dos Santos. Virtualização e docker. **Revista Vincci**, v. 2, n. 2, p. 152-179, ago./dez., 2017. Disponível em: <https://revistavincci.satc.edu.br/index.php/Revista-Vincci/article/view/89/51>. Acesso em: 25 set. 2022.

GALDINO, Higor Eduardo Borges; RONDINA, João Marcelo. **Virtualização de Containers utilizando Docker** .2020. Disponível em: <https://www.researchgate.net/publication/345081364_Virtualizacao_de_Containers_utilizando_Docker>. Acesso em: 20 ago. 2022.

GHIMIRAY, Deepan. O que é uma máquina virtual e como ela funciona?. **Avast**. 24 jun. 2022. Disponível em: <https://www.avast.com/pt-br/c-virtual-machine>. Acesso em: 21 ago. 2022.

GOLDBERG, Robert P. **Architectural Principles for Virtual Computer Systems**. Tese. Universidade de Harvard: Cambridge, 1972. Disponível em: <https://apps.dtic.mil/sti/pdfs/AD0772809.pdf>. Acesso em: 16 out. 2022.

MACEDO, Diego. Conheça os quatro modelos de Virtualização de Desktop. **Blog Diego Macêdo Um pouco de tudo sobre T.I**. 12 mar. 2012. Disponível em: <https://www.diegomacedo.com.br/conheca-os-quatro-modelos-de-virtualizacao-de-desktop/>. Acesso em: Acesso em: 16 out. 2022.

MACHADO, Francis Berenger; MAIA, Luiz Paulo. **Arquitetura de sistemas operacionais**. 4ª ed. Rio de Janeiro: LTC, 2007. Disponível em: < https://www.academia.edu/12107184/Arquitetura_de_Sistemas_Operacionais_Francis_Berenger_Machado>. Acesso em: 17 out. 2022.

MATTOS, Diogo Menezes Ferrazani. **Virtualização**: VMWare e Xen. 2008. Disponível em: <https://www.researchgate.net/publication/255657320_Virtualizacao_VMWare_e_Xen >. Acesso em: 18 out. 2022.

Memória da Maquina Virtual. **Vmware**. Disponível em: < https://docs.vmware.com/br/VMware-vSphere/7.0/com.vmware.vsphere.resmgmt.doc/GUID-C25A8823-F595-4322-BD0D-4FD5B081F877.html >. Acesso em: 18 out. 2022.

MINSU, Chae; HWAMIN, Lee; KIYEOL, Lee. **A performance comparison of linux containers and virtual machines using Docker and KVM**. 2019. Disponível em: <https://link.springer.com/article/10.1007/s10586-017-1511-2#auth-HwaMin-Lee>. Acesso em: 19 out. 2022.

PHPMYADMIN. **phpMyAdmin**. 2022; Disponível em: <https://www.phpmyadmin.net/>. Acesso em: 22 out. 2022.

O surgimento da virtualização e as grandes mudanças que ela trouxe. **Hialinx T.I**. 26 nov. 2019. Disponível em: < https://www.hialinx.com.br/post/o-surgimento-da-virtualiza%C3%A7%C3%A3o-e-as-grandes-mudan%C3%A7as-que-ela-trouxe#:~:text=A%20ideia%20da%20virtualiza%C3%A7%C3%A3o%20surgiu,ser%20feito%20manualmente%20pelo%20operador.>. Acesso em: 22 out. 2022

RESENDE, Pedro Augusto. **Adoção da arquitetura de containers docker:** um estudo de caso. Trabalho de Conclusão de Curso. Universidade Católica de Brasília, Brasília, 2019. Disponível em: <https://repositorio.ucb.br:9443/jspui/bitstream/123456789/12857/1/PedroAugustoResendeTCCGradua%C3%A7ao2019.pdf.> Acesso em: 22 out. 2022.

RUBENS, Paul**.** What are containers and why do you need them?**. CIO**. 27 jun. 2015. Disponível em: <http://www.cio.com/article/2924995/enterprisesoftware/what-are-containers-and-why-do-you-need-them.html> Acesso em: 10 set. 2022

SOUZA, Vinícius. O comando HTOP no Linux. **Iron Linux**. S.d. Disponível em: <https://ironlinux.com.br/o-comando-htop-no-linux/>. Acesso em: 29 out. 2022.

TANEMBAUM, Andrew S. **Sistemas Opercionais Modernos**. Tradução: Daniel Vieira e Jorge Ritter. 4ª ed. São Paulo: Pearson Education do Brasil, 2016. Disponível em: <http://ldemetrio.com.br/Livros/Livros_TI/segunda_unid/Sistemas%20Operacionais%20Modernos%20-%20Tanenbaum%20-%204%20Edi%C3%A7%C3%A3o.pdf>. Acesso em: 8 out. 2022.

ROCHA, Victor. Tipos de virtualização. [S. l.], 15 mar. 2013. Disponível em: https://www.tiespecialistas.com.br/tipos-de-virtualizacao/. Acesso em: 16 out. 2022.

VIRTUALBOX. [S. l.], 9 out. 2022. Disponível em: https://www.virtualbox.org/. Acesso em: 9 out. 2022.

MERKEL, Dirk. Docker: Lightweight Linux Containers for Consistent Development and Deployment. Https://www.seltzer.com, [s. l.], 1 mar. 2014. Disponível em: https://www.seltzer.com/margo/teaching/CS508.19/papers/merkel14.pdf. Acesso em: 9 out. 2022.

WHAT are containers and why do you need them?. In: RUBENS, Paul. What are containers and why do you need them?. [S. l.], 27 jun. 2017. Disponível em: https://www.cio.com/article/247005/what-are-containers-and-why-do-you-need-them.html. Acesso em: 7 ago. 2022.

[i]

[i] Junnior J O, Henrique R O. Fernando L A S. **Estudo comparativo laboratorial entre máquina virtual e container Docker para aplicação Web.** Sistemas de Informação – artigo de TCC. Rede de Ensino Doctum. Ipatinga-MG, 2022.