

UNIFESO - CENTRO UNIVERSITÁRIO SERRA DOS ÓRGÃOS

CCT - CENTRO DE CIÊNCIAS E TECNOLOGIA

GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

**VIRTUALIZAÇÃO OPEN-SOURCE PARA APLICAÇÕES FOCADAS EM  
INFRAESTRUTURA COMPUTACIONAL**

**WILLIAN PASSOS DE SOUZA**

TERESÓPOLIS

2020

UNIFESO - CENTRO UNIVERSITÁRIO SERRA DOS ÓRGÃOS  
GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

**VIRTUALIZAÇÃO OPEN-SOURCE PARA APLICAÇÕES FOCADAS EM  
INFRAESTRUTURA COMPUTACIONAL**

**WILLIAN PASSOS DE SOUZA**

TERESÓPOLIS  
CCT - CENTRO DE CIÊNCIAS E TECNOLOGIA

2020

S713 Souza, Willian Passos de.  
Virtualização Open-Source para aplicações focadas em infraestrutura computacional. /Willian Passos de Souza. – 2020.  
29f.

Trabalho de Conclusão de Curso (Bacharelado em Ciência da Computação) – Centro Universitário Serra dos Órgãos, UNIFESO, Teresópolis, 2020.  
Bibliografia: f. 24.  
Orientador: Laion Luiz Fachini Manfroi.

1-Ciências da Computação. 2. “Open-Source”. 3. “Virtualização” 4. “Hyper-V”. 5. “Cluster”. I. Título.

CDD 004

# **VIRTUALIZAÇÃO OPEN-SOURCE PARA APLICAÇÕES FOCADAS EM INFRAESTRUTURA COMPUTACIONAL**

**WILLIAN PASSOS DE SOUZA**

Trabalho Final de Curso submetido à banca examinadora constituída de acordo as Normas de Trabalho Final de Curso estabelecidas pelo Centro de Ciências e Tecnologia, como parte dos requisitos necessários para a obtenção do grau de Cientista da Computação.

Aprovado em: 17/11/2020

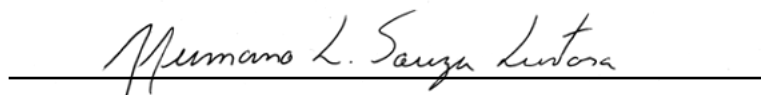
Por:



Orientador: Prof. M.Sc. Laion Luiz Fachini Manfro



D.Sc. Alberto Torres Angonese



D.Sc. Hermano Lourenço Souza Lustosa

## RESUMO

Atualmente, muitos computadores e programas utilizam a arquitetura de Von Neumann, o conjunto de memória de dados e instruções, onde os programas são executados de forma sequencial. Um cluster pode ser definido como um sistema que abrange dois ou mais computadores onde denominados nós que estão ligados em redes a fim de executar cargas de processamento em paralelo, aplicações ou outras tarefas. Para trabalhar com estes conjuntos, denominamos o master da rede, encarregado de compor a carga de processos simultâneo e homogêneo, que chamamos de nó ou nodos da rede. O conceito de cluster surgiu com o nome de beowulf e teve sua ascensão no começo da década de 90 pela NASA (Administração Nacional da Aeronáutica e Espaço) no intuito de obter maior poder computacional obtendo um supercomputador para suas pesquisas, acabou tornando-se nome genérico para todos os clusters que se assemelhavam à configuração do original. Os clusters projetados sob outras plataformas Open-Source aplica-se a distribuição OpenSSI com o foco desempenho em ambiente Linux, com denominações diferentes, o Kubernetes e aplicações como plataforma Docker entre outros hipervisores, pode proporcionar manipulação em contêineres com micro serviços ou com escalabilidade de recursos de armazenamento, ambiente de desenvolvimento, provisionamento de processamento (BEOWULF, 2003) nestes casos podemos atrelar as mesmas instruções de dados divididos em mais processadores e memória, podemos destinar o uso em ambiente em série aplicando-se em micro-serviço atrelado ao uso de ambiente de sistema virtual (BOOKMAN, 2002). Este trabalho busca inserir conceitos e aplicações com as tecnologias mais utilizadas atualmente no mercado de tecnologia, referenciando a essência dos clusters e realizando uma análise do que pode ser entregue em aplicações virtualizadas.

**Palavras – Chave:** Virtualização, Open-Source, Hyper-V, Cluster.

## ABSTRACT

Nowadays, many computers and softwares utilize Von Neumann's architecture, thus being only data memory and instructions, where softwares run in a sequential way. In this context, cluster may be defined as a system that comprehends two or more computers defined as nodes, which are connected in a network in order to process loads of parallel processing, applications or other tasks. In order to work with these ensembles, we denominate the network's master, which is in charge to set the load of simultaneous and homogeneous processes called nodes or network nodes. The concept of cluster sprout with Beowulf and gained more light in early 90's with NASA (National Aeronautics and Space Administration) in order to achieve more computational power for their researches and that ended up becoming a more general name for all clusters that were similar to the original. clusters designed under other Open-Source platforms applies to SSI distribution aiming performance in Linux environment with different denomination, like Kubernetes and applications in Docker among other hypervisors, platforms which may provide containers manipulation with micro services or with scalability in data storage, development environment, provisioning process (BEOWULF, 2003). In those cases, we may split the same data instructions into more processors and memory, and also utilize it into serial environment, applying it in micro-service destined for virtual system (BOOKMAN, 2002).

Keywords: Cluster, Hyper-V, Open-Source, Virtualization

## LISTA DE ABREVIATURAS E SIGLAS

CSS	Cascading Style Sheets (Folha de Estilo em Cascata)
CPU	<i>Central Process Unit (Unidade Central de Processamento)</i>
DNA	<i>Ácido Desoxirribonucleico</i>
DNS	<i>Domain Name System (Sistema de Nomes de Domínios)</i>
HD	<i>Hard Disc (Disco Rígido)</i>
HPC	<i>High-Performance Computing (Computação de Alto Desempenho)</i>
HTML	<i>HyperText Markup Language (Linguagem de Marcação de HiperTexto)</i>
HTTP	<i>Hypertext Transfer Protocol</i>
IP	<i>Internet Protocol (Protocolo de Internet)</i>
JS	<i>Java Script</i>
NASA	<i>National Aeronautics and Space Administration (Administração Nacional da Aeronáutica e Espaço)</i>
PING	<i>Packet Internet Network Groper</i>
SO	<i>Sistema Operacional</i>
SDDC	<i>Software Defined Data Center</i>
SDN	<i>Software Defined Network</i>
SSD	<i>Solid State Drive (Disco de Estado Sólido)</i>
SDS	<i>Software Defined Storage</i>
VHD	<i>Virtual Hard Disc (Disco Rígido Virtual)</i>
VM	<i>Virtual Machine (Máquina Virtual)</i>

## SUMÁRIO

<b>1</b>	<b>INTRODUÇÃO . . . . .</b>	<b>4</b>
1.1	OBJETIVO . . . . .	4
1.2	GERAL . . . . .	5
1.3	ESPECÍFICOS . . . . .	5
1.4	JUSTIFICATIVA . . . . .	5
<b>2</b>	<b>COMPUTADORES DE ALTO DESEMPENHO E VIRTUA- LIZAÇÃO . . . . .</b>	<b>6</b>
2.1	HYPER-V . . . . .	6
2.1.1	Cenário 1 . . . . .	7
2.1.2	Cenário 2 . . . . .	7
2.2	CENÁRIO DE APLICAÇÃO . . . . .	8
2.3	BENCHMARK . . . . .	9
2.3.1	DNS Primário . . . . .	10
2.3.2	DNS Secundário . . . . .	11
2.3.3	Serve-01 . . . . .	12
<b>3</b>	<b>TIPOS DE CLUSTER . . . . .</b>	<b>13</b>
3.1	CARGA DE PROCESSAMENTO . . . . .	13
3.2	DISPONIBILIDADE . . . . .	13
3.3	ARMAZENAMENTO . . . . .	13
<b>4</b>	<b>METODOLOGIA E PROPOSTA . . . . .</b>	<b>14</b>
4.1	AVALIAÇÃO DE AMBIENTE . . . . .	14
4.1.1	Console Hyper-V . . . . .	14
4.1.2	Rede . . . . .	14
4.1.3	Metodologia de Rede . . . . .	15
4.1.4	VMs . . . . .	15
4.1.5	Serviço Docker . . . . .	15
4.2	DIRETÓRIOS E AMBIENTES DE CONFIGURAÇÃO . . . . .	16
4.2.1	Servidor Web . . . . .	16
4.2.2	Servidor DNS . . . . .	17
4.2.3	Arquivos de Script . . . . .	18
4.2.4	resolv.conf . . . . .	18
4.2.5	Zonas de Domínio Web . . . . .	18



4.2.6	Zona Servidor Web . . . . .	19
4.2.7	Zona de Domínio DNS . . . . .	20
<b>5</b>	<b>CONCLUSÃO . . . . .</b>	<b>23</b>

# 1 INTRODUÇÃO

A exigência de um maior desempenho das aplicações faz com que a computação esteja sempre em constante evolução. No passado, a base da computação distribuída de alto desempenho eram processadores de uso genérico, ou seja, os processadores que executavam todos tipos de instruções e não eram dedicados. Atualmente, existem diversas estratégias para prover desempenho não exclusivamente dedicado ao processamento em processadores, podemos exemplificar estratégias com GPUs, computação em nuvem, virtualização, entre outras técnicas, cada uma em um cenário de aplicação diferenciado.

Este trabalho está focado no estudo prático e avaliativo em aplicações OpenSource utilizando o virtualizador Hyper-V, podendo orquestrar clusters para setores que lidam com grandes volumes de dados (matemática, física, astronomia, genética, financeiro e jogos) (BEOULF.ORG, 2003) dentre outros. Tratamos aqui recursos computacionais de aplicação em análise de dados complexos, provendo benchmarks de análise de desempenho da orquestração em série dos computadores em laboratórios, escalabilidade (se houver a necessidade de um crescimento de recursos virtuais), dependendo de como aplicar a técnica de acordo com o domínio de cada problema (BOOKMAN, 2003). O uso da computação paralela em meio às técnicas de virtualização possuem suas contribuições na otimização do desempenho, podendo estar contidos em sistemas complexos e diversos, utilizando threads em aglomeração de computadores para oferecer mais recursos computacionais para processamento e escalabilidade de serviços, não apenas em computadores físicos, mas também em serviços aplicados a sistemas em "containers", onde os grandes *datacenters* modernos estão baseados (SOLTESZ, et al. 2007).

## 1.1 OBJETIVO

A proposta de objetivo deste trabalho é obter um ambiente de avaliação demonstrando a configuração e a construção de um cenário utilizando ambiente de softwares livres com Linux Ubuntu para servidor Web e CentOS para servidor DNS. Nestes servidores serão hospedados uma página web no ambiente de rede, simulando uma aplicação utilizando a ferramenta Docker provendo desses recursos de serviços. Buscaremos avaliar o desempenho em laboratório utilizando a virtualização no Hyper-V com o intuito de orquestrar uma rede para aplicação de demandas de processos de alto desempenho, avaliação de benchmarks, usabilidade de processamentos, balanço de carga e disponibilidade de serviços em computadores virtuais. É analisada a performance das máquinas virtuais durante a *live migration* e testes de conectividade.

## 1.2 GERAL

Mostrar a utilização de um sistema de processamento paralelo entre as máquinas virtuais, fazendo migração no Hyper-V, teste de conectividade englobando as etapas e procedimentos necessários, partindo desde sua montagem virtual até o desenvolvimento de uma aplicação de ambiente de desenvolvimento de infraestrutura computacional provendo um sistema de servidor Web e servidor DNS com análise de desempenho, usabilidade e gerenciamento.

## 1.3 ESPECÍFICOS

Esta pesquisa tem como fundamentação a montagem e utilização de um laboratório para a implementação de uma aplicação *failover* com foco em disponibilidade de serviços em container voltado para aplicação de código aberto. Esta montagem possui o intuito de avaliar e mostrar todas as ferramentas e os orquestradores virtuais com tecnologias na plataforma Docker, aplicado ao Hyper-V. O principal objetivo específico é demonstrar o comportamento em ambiente de provisionamento de redundancias e tolerante a falhas desses serviços, analisando o desempenho e sua escalabilidade, tirando grande proveito disto através da paralelização.

## 1.4 JUSTIFICATIVA

Temos uma busca constante por mudanças, principalmente quando foco do mercado de computação está sempre procurando adaptar as tendências comerciais, cada vez mais direcionando para o crescimento da população e o aumento natural do consumo de dados através de aplicações para resoluções de modelos do mundo real.

A ciência e a tecnologia, impulsionadas pela grande capacidade dos sistemas de computação atuais, permitem a verificação de teorias e a construção de modelos computacionais cada vez mais complexos, onde a indústria utiliza e aplica estas ferramentas com o intuito de aumentar sua produção e resolver problemas complexos, cada vez mais automatizados.

Com base nos critérios de avaliação de cenários onde, em grande parte das universidades atualmente possuem laboratórios que ficam boa parte do tempo ociosos, podemos disponibilizar os recursos computacionais sob a demanda de projetos, testes, entre outras aplicações escalonáveis. Isto pode proporcionar aos alunos e aos professores pesquisadores em ambiente cluster com ferramentas de virtualização (PITANGA, 2002).

## 2 COMPUTADORES DE ALTO DESEMPENHO E VIRTUALIZAÇÃO

A frequente necessidade de maior capacidade computacional em sistemas de computação intensiva faz com que os processadores e outros componentes tenham que trabalhar cada vez com mais rapidez. Porém, limites físicos ainda existem na computação, tais como: termodinâmica, limites de armazenamento, limites de comunicação e de frequência. Uma forma de contornar essa restrição é a utilização de técnicas que possibilitam o processamento distribuído. A grande meta da agregação de recursos computacionais é prover respostas para as limitações encontradas nas arquiteturas centralizadas utilizando técnicas que possibilitem o processamento distribuído.

O aumento de capacidade de processamento tem possibilitado pesquisas científicas, dentre outros, porém os sistemas computacionais com apenas um único núcleo no processador passaram a não atender essas demandas em usabilidade em cenários de grandes escala. Para isto temos mais uma justificativa para a computação de alto desempenho, que abrange os supercomputadores, compostos de máquinas em clusters (BROWN, 2006).

Para processar grandes volumes de dados em um curto período de tempo, podemos exemplificar com sistemas que fazem análises meteorológicas, que integram cadeias de nucleotídeos presentes no DNA humano, etc (PACHECO, 2012).

A tecnologia de Virtualização já é utilizada há algum tempo e permitiu usar "versões virtuais" do hardware, com a capacidade de "virtualizar" as aplicações ao dia-a-dia, tornando-se flexível e melhor gerenciável. O armazenamento na virtualização trata-se de uma camada de software aplicada nos discos físicos ou dispositivos com o nome de SDS (Software Defined Storage), em redes por sua vez, podem criar uma estrutura lógica baseado na rede física permitindo interação entre os hosts físicos e as necessidades virtualizadas chamada de SDN (Software Defined Network). Para que o sistema operacional nativo opera em compartilhamento com sistemas virtuais usando-se da mesma máquina física, logo cada sistema virtualizado passa a conter seus próprios serviços de rede, armazenamento e processamento agregando valor a infra-estrutura de TI trazendo eficiência de investimento e flexibilidade recursos em servidores com baixo valor agregado, menor consumo energético e tornando-se o SDDC (Software Defined Data Center) (SILBERSCHATZ, 2001).

### 2.1 HYPER-V

Entre variadas plataformas de virtualização existentes, o Hyper-V da Microsoft se mostrou eficiente, garantindo um ambiente confiável para aplicação de virtualização com escalabilidade, que foi empregado no desenvolvimento deste trabalho. Nestes cenários abordados, podemos identificar o nível do comportamento do uso de virtualização aplicado

à máquina física, quais as possibilidades de gerenciamento temos utilizando o virtualizador diretamente ao hardware, na busca por um maior desempenho. Na visão de uma outra instância, o virtualizador foi aplicado em uma camada em que o sistema operacional faz o gerenciamento da virtualização e os micros serviços.

#### 2.1.1 Cenário 1

Esta camada de virtualização fica mais próxima ao hardware, com o intuito de obter mais performance, motivado pelo fato de que as máquinas virtuais realizam a comunicação de imediato com o hypervisor.

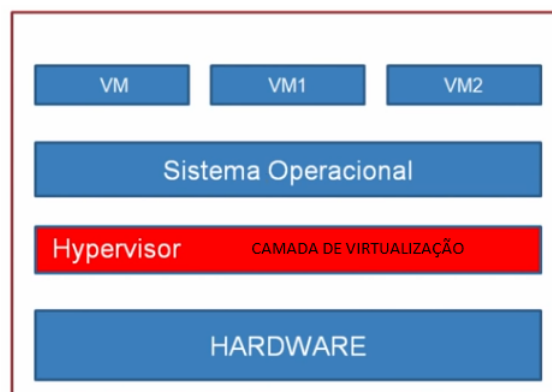


Figura 1 – *Cenário 1 da Camada de virtualização*

#### 2.1.2 Cenário 2

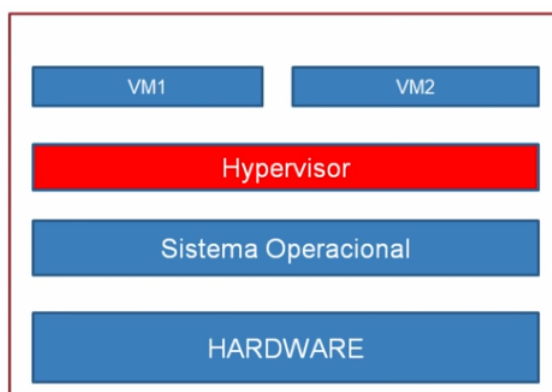


Figura 2 – *Cenário 2 da Camada de virtualização*

Neste outro cenário temos a camada do Hyper-V acima do sistema operacional, não há troca de dados direto com a camada de hardware, por isso pode não conter um desempenho comparado ao primeiro cenário, pois o sistema operacional irá gerenciar as máquinas virtuais.

## 2.2 CENÁRIO DE APLICAÇÃO

A proposta do cenário adotado para esta avaliação do ambiente é demonstrada através da metodologia empregada para compor a estrutura, exemplificando como estão dispostos os servidores, suas denominações de funções e os pacotes de serviços utilizados. Aos demais computadores físicos, foram adotados os mesmos recursos de sistemas que as máquinas virtuais para análise da aplicação fora do ambiente virtual, com o intuito de mostrar a aplicação dos servidores e o comportamento de acesso à estas aplicações.

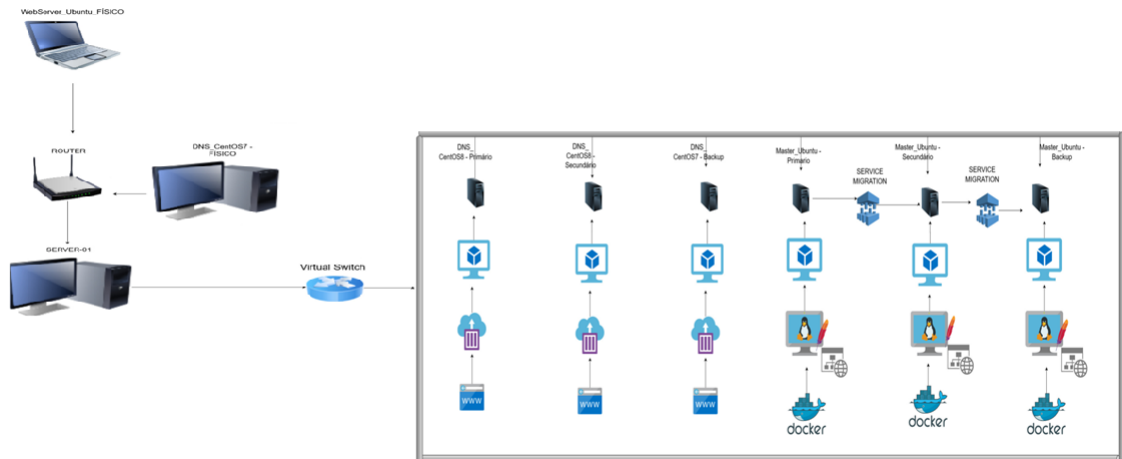


Figura 3 – Metodologia de Aplicação

A troca de sistema enquanto a VM está em execução é uma parte essencial para estrutura de virtualização, disponibilidade e manter serviços ininterruptos por permitir trocar configurações sejam para portabilidades, armazenamento ou necessidade de manutenção e mantendo um ambiente contra falhas. Na implementação como avaliação de ambiente com o Hyper-V, a exportação e importação de sistemas virtuais torna o desenvolvimento uma parte de algo de infraestrutura que funcionam desde o datacenter até pequenas organizações. Cenários baseados em computadores básicos que podem escalonar ou aumentar os recursos computacionais de forma fácil e segura, essa forma de migração de sistemas permite troca ou inserção de SO a cada demanda estabelecida.

O armazenamento adotado pode variar de acordo com o domínio de cada ambiente. Na visão de grande massa de dados, pode ser mostrado que tanto aplicações como ambiente de virtualização podem estar contidos em variados tipos de dispositivo de armazenamento.

Neste trabalho a carga de armazenamento para virtualização no Hyper-V foi o desenvolvimento do ambiente. Uma parte do HD foi criado com um diretório onde estão todos os VHDs gerados das VMs do console Hyper-V, o sistema hospedeiro do computador onde foi feita a virtualização foi utilizado SSD, com um maior desempenho e flexibilidade do gerenciamento do sistema e, conseqüentemente, seus serviços.

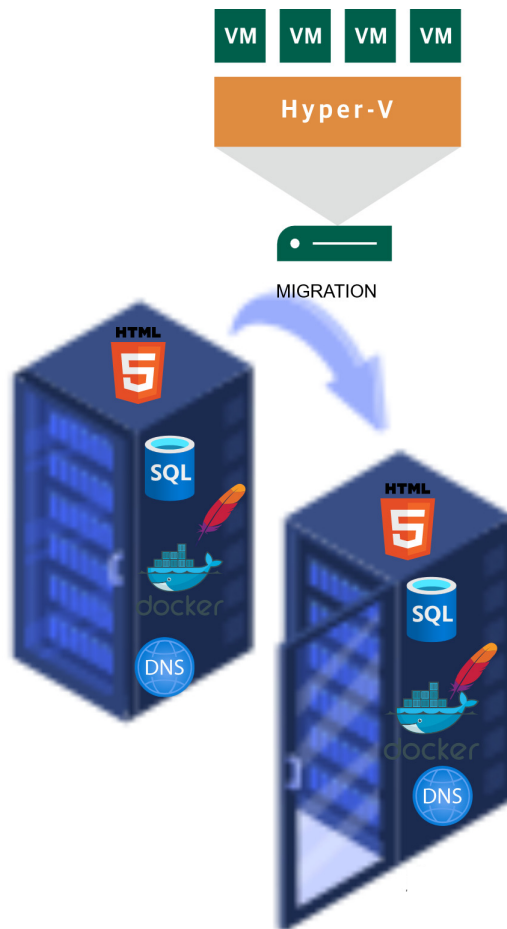


Figura 4 – *Ambiente de Migração*

O gerenciador do Hyper-V concebe todas as VMs mostrando o estado de execução, uso da CPU e das memórias em seu painel, o tempo de execução o qual estão ativas em consumo dos recursos do computador adotado como Servidor Físico (SERVER-01) que compõe todo o cenário de virtualização, além de um nó do cluster.

Os recursos que a máquina física apresenta são os servidores virtuais primários, secundário e os de backup, de forma padronizada, para que fique de forma homogênea e organizada.

## 2.3 BENCHMARK

Como uma das formas para demonstrar que um serviço está disponível, bem como os computadores em rede, foi utilizado o comando *ping* para análise de pacotes e identificação da latência entre os serviços e servidores.

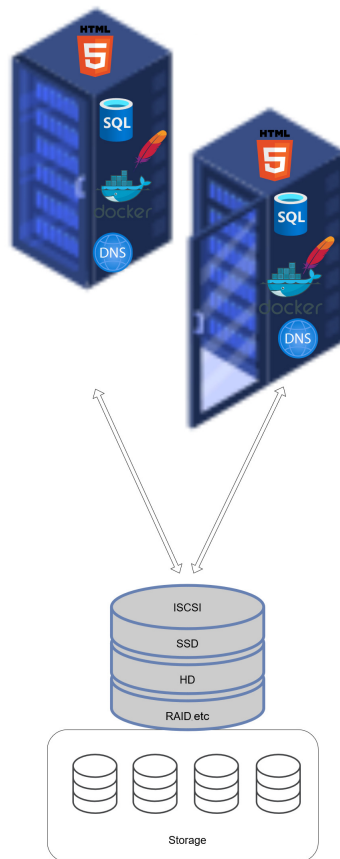


Figura 5 – *Ambiente de Armazenamento*

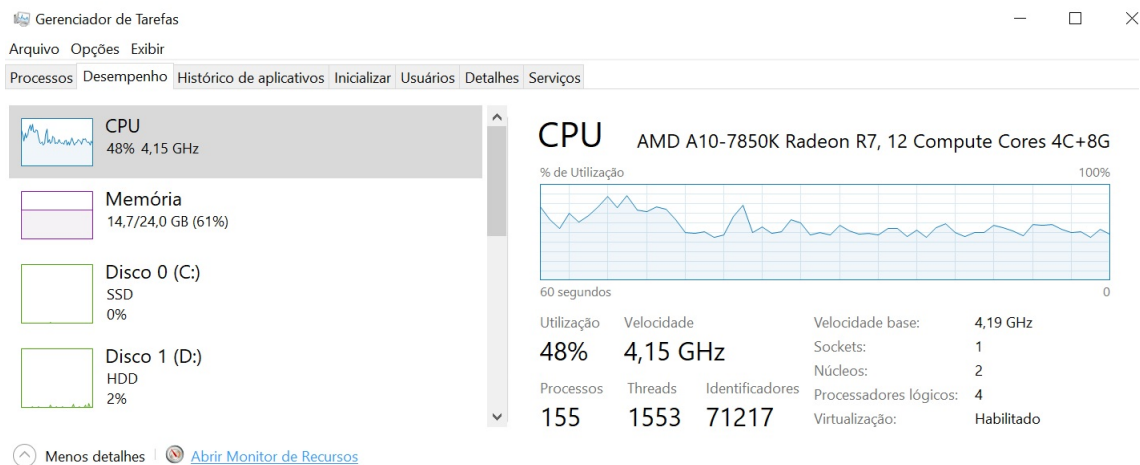


Figura 6 – *Gerenciador de Tarefa*

### 2.3.1 DNS Primário

Desta forma, o tráfego do servidor DNS primário resolvendo os serviços do servidor Web primário, ao pingar para o [www.no-paladar.com.br](http://www.no-paladar.com.br) o servidor DNS retorna entre 1 à 2 milissegundos, mostrando o resultado com o IP resultante do servidor Web 192.168.0.108



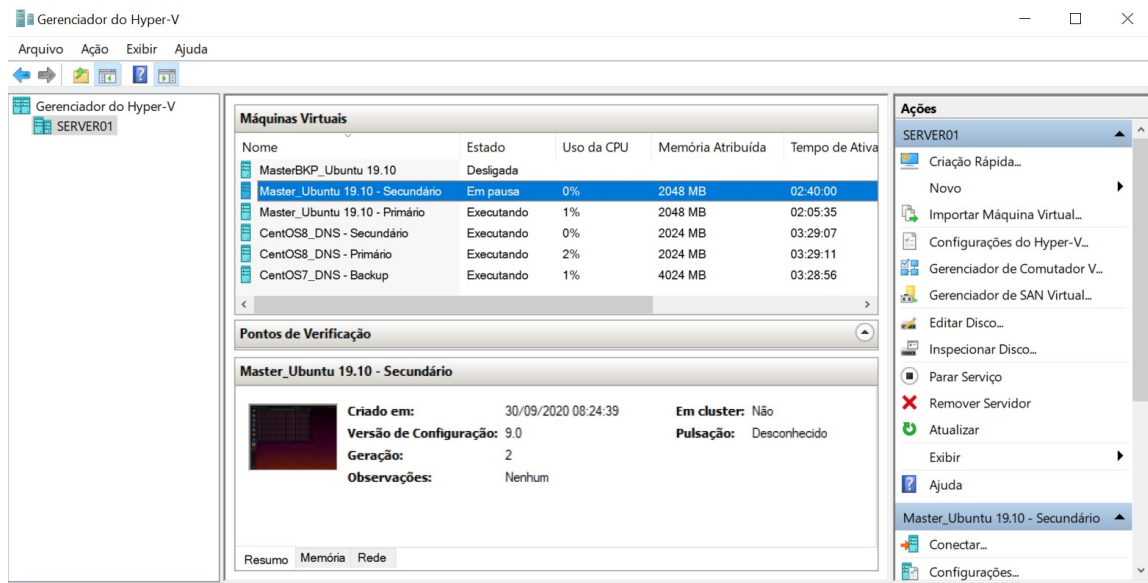


Figura 7 – Gerenciador Hyper-V

```
[root@dns named]# ping www.no-paladar.com.br
PING master.no-paladar.com.br (192.168.0.108) 56(84) bytes of data:
64 bytes from 192.168.0.108 (192.168.0.108): icmp_seq=1 ttl=64 time=1.17 ms
64 bytes from 192.168.0.108 (192.168.0.108): icmp_seq=2 ttl=64 time=0.333 ms
64 bytes from 192.168.0.108 (192.168.0.108): icmp_seq=3 ttl=64 time=0.374 ms
64 bytes from 192.168.0.108 (192.168.0.108): icmp_seq=4 ttl=64 time=1.48 ms
64 bytes from 192.168.0.108 (192.168.0.108): icmp_seq=5 ttl=64 time=0.488 ms
64 bytes from 192.168.0.108 (192.168.0.108): icmp_seq=6 ttl=64 time=1.76 ms
64 bytes from 192.168.0.108 (192.168.0.108): icmp_seq=7 ttl=64 time=0.455 ms
64 bytes from 192.168.0.108 (192.168.0.108): icmp_seq=8 ttl=64 time=1.80 ms
64 bytes from 192.168.0.108 (192.168.0.108): icmp_seq=9 ttl=64 time=2.11 ms
64 bytes from 192.168.0.108 (192.168.0.108): icmp_seq=10 ttl=64 time=1.05 ms
64 bytes from 192.168.0.108 (192.168.0.108): icmp_seq=11 ttl=64 time=1.32 ms
64 bytes from 192.168.0.108 (192.168.0.108): icmp_seq=12 ttl=64 time=2.47 ms
64 bytes from 192.168.0.108 (192.168.0.108): icmp_seq=13 ttl=64 time=1.43 ms
64 bytes from 192.168.0.108 (192.168.0.108): icmp_seq=14 ttl=64 time=0.344 ms
64 bytes from 192.168.0.108 (192.168.0.108): icmp_seq=15 ttl=64 time=0.549 ms
64 bytes from 192.168.0.108 (192.168.0.108): icmp_seq=16 ttl=64 time=2.06 ms
64 bytes from 192.168.0.108 (192.168.0.108): icmp_seq=17 ttl=64 time=0.377 ms
64 bytes from 192.168.0.108 (192.168.0.108): icmp_seq=18 ttl=64 time=0.456 ms
64 bytes from 192.168.0.108 (192.168.0.108): icmp_seq=19 ttl=64 time=1.76 ms
64 bytes from 192.168.0.108 (192.168.0.108): icmp_seq=20 ttl=64 time=0.457 ms
64 bytes from 192.168.0.108 (192.168.0.108): icmp_seq=21 ttl=64 time=3.37 ms
```

Figura 8 – Ping Servidor DNS Primário

ao qual o serviço pertence.

### 2.3.2 DNS Secundário

Para exemplificar, foi realizada a mesma avaliação ao servidor DNS secundário, que estará sempre resolvendo o servidor Web secundário, como os servidores são redundantes, ele passará a prover o serviço `www.no-paladar.com.br` pelo IP `192.168.0.110` com o retorno da latência entre 1 à 2 milissegundos.

```

[root@dns dns]# ping www.no-paladar.com.br
PING webserver.no-paladar.com.br (192.168.0.110) 56(84) bytes of data:
64 bytes from 192.168.0.110 (192.168.0.110): icmp_seq=1 ttl=64 time=1.37 ms
64 bytes from 192.168.0.110 (192.168.0.110): icmp_seq=2 ttl=64 time=0.425 ms
64 bytes from 192.168.0.110 (192.168.0.110): icmp_seq=3 ttl=64 time=0.425 ms
64 bytes from 192.168.0.110 (192.168.0.110): icmp_seq=4 ttl=64 time=0.552 ms
64 bytes from 192.168.0.110 (192.168.0.110): icmp_seq=5 ttl=64 time=0.430 ms
64 bytes from 192.168.0.110 (192.168.0.110): icmp_seq=6 ttl=64 time=0.489 ms
64 bytes from 192.168.0.110 (192.168.0.110): icmp_seq=7 ttl=64 time=3.61 ms
64 bytes from 192.168.0.110 (192.168.0.110): icmp_seq=8 ttl=64 time=0.529 ms
64 bytes from 192.168.0.110 (192.168.0.110): icmp_seq=9 ttl=64 time=0.492 ms
64 bytes from 192.168.0.110 (192.168.0.110): icmp_seq=10 ttl=64 time=0.686 ms
64 bytes from 192.168.0.110 (192.168.0.110): icmp_seq=11 ttl=64 time=0.370 ms
64 bytes from 192.168.0.110 (192.168.0.110): icmp_seq=12 ttl=64 time=1.36 ms
64 bytes from 192.168.0.110 (192.168.0.110): icmp_seq=13 ttl=64 time=0.328 ms
64 bytes from 192.168.0.110 (192.168.0.110): icmp_seq=14 ttl=64 time=0.407 ms
64 bytes from 192.168.0.110 (192.168.0.110): icmp_seq=15 ttl=64 time=0.390 ms
64 bytes from 192.168.0.110 (192.168.0.110): icmp_seq=16 ttl=64 time=0.382 ms
64 bytes from 192.168.0.110 (192.168.0.110): icmp_seq=17 ttl=64 time=0.363 ms
64 bytes from 192.168.0.110 (192.168.0.110): icmp_seq=18 ttl=64 time=0.345 ms
64 bytes from 192.168.0.110 (192.168.0.110): icmp_seq=19 ttl=64 time=2.53 ms
64 bytes from 192.168.0.110 (192.168.0.110): icmp_seq=20 ttl=64 time=0.362 ms
64 bytes from 192.168.0.110 (192.168.0.110): icmp_seq=21 ttl=64 time=0.608 ms

```

Figura 9 – Ping Servidor DNS Secundário

```

PS C:\Users\Server-01 > ping www.no-paladar.com.br -t

Disparando master.no-paladar.com.br [192.168.0.108] com 32 bytes de dados
Resposta de 192.168.0.108: bytes=32 tempo<1ms TTL=64
Resposta de 192.168.0.108: bytes=32 tempo=3ms TTL=64
Resposta de 192.168.0.108: bytes=32 tempo=3ms TTL=64
Resposta de 192.168.0.108: bytes=32 tempo<1ms TTL=64
Resposta de 192.168.0.108: bytes=32 tempo=1ms TTL=64
Resposta de 192.168.0.108: bytes=32 tempo<1ms TTL=64
Resposta de 192.168.0.108: bytes=32 tempo<1ms TTL=64
Resposta de 192.168.0.108: bytes=32 tempo=3ms TTL=64
Resposta de 192.168.0.108: bytes=32 tempo<1ms TTL=64
Resposta de 192.168.0.108: bytes=32 tempo<1ms TTL=64
Resposta de 192.168.0.108: bytes=32 tempo<1ms TTL=64
Resposta de 192.168.0.108: bytes=32 tempo=1ms TTL=64
Resposta de 192.168.0.108: bytes=32 tempo<1ms TTL=64
Resposta de 192.168.0.108: bytes=32 tempo=2ms TTL=64
Resposta de 192.168.0.108: bytes=32 tempo=2ms TTL=64
Resposta de 192.168.0.108: bytes=32 tempo<1ms TTL=64
Resposta de 192.168.0.108: bytes=32 tempo=1ms TTL=64
Resposta de 192.168.0.108: bytes=32 tempo=1ms TTL=64
Resposta de 192.168.0.108: bytes=32 tempo=1ms TTL=64

```

Figura 10 – Ping SERVE-01

### 2.3.3 Serve-01

Toda avaliação feita anteriormente foi pelas máquinas virtuais, o ping no Server-01 foi implementado mostrando que o serviço pode ser acessado pela rede externa.

### 3 TIPOS DE CLUSTER

#### 3.1 CARGA DE PROCESSAMENTO

Neste trabalho o balanceamento de carga é realizado para que não haja problemas de desempenho ou congestionamento relacionado às respostas das requisições, com uma infraestrutura escalável. Foi buscado aqui o melhoramento para a distribuição de tarefas agregadas aos serviços em cada domínio das máquinas virtuais. Neste trabalho a carga de processo é feito pelo uso dos sistemas virtuais utilizando os processos da plataformas Docker, Apache e serviço de DNS, alocado para gerenciar o uso de uma aplicação Failover que consiste em um servidor de aplicação Web. A contribuição desta técnica é a parte redundante de prover os serviços pré-determinados ininterruptamente sendo a principal tarefa do SO juntamente com o Hypervisor. A proposta de utilização destas ferramentas em conjunto é a busca por uma outra instância do servidor, que será projetada para manter as VMs e os serviços Web e DNS. Há a possibilidade da migração em tempo-real dessas máquinas, tanto a carga de trabalho para o próprio host hospedeiro do Hypervisor, quanto ao host virtual.

#### 3.2 DISPONIBILIDADE

Um serviço denominado "ativo", que possa estar em funcionamento ininterruptamente, consiste em estar apto para suprir as necessidades, as demandas, a partir de um sistema confiável, com emprego de técnicas para tolerância a falhas.

Neste trabalho a disponibilidade é a redundância do servidor de aplicação Web. Ou seja, se o serviço de algum servidor parar ou até mesmo apresentar problemas de natureza de infraestrutura computacional, deve manter a continuidade de execução.

#### 3.3 ARMAZENAMENTO

Neste ambiente adotado para avaliação não foi inserido um servidor exclusivo para armazenamento, partindo do princípio que não haverá alocação de banco de dados nem armazenamento de dados. Contudo, ressaltamos que algumas configurações foram feitas para que toda a parte computacional virtual consistisse de uma configuração do diretório do host hospedeiro, para fins didáticos e como uma boa estratégia de implementação e organização, espelhando com uma maior fidelidade um ambiente real. Os VHDs das máquinas virtuais ficaram dispostos para que caso haja a necessidade de migração de servidor, uma correção possa configurar uma outra instância, já previamente preparada na rede.

## 4 METODOLOGIA E PROPOSTA

### 4.1 AVALIAÇÃO DE AMBIENTE

Neste capítulo são descritas as abordagens dos testes avaliativos do ambiente para análise de desempenho em benchmarks. Este mesmo método pode ser implantado em uma infraestrutura computacional de distribuição de carga de trabalhos com necessidade de disponibilidade de serviços de servidores Web. Geralmente estes cenários são provisionados pela composição de servidores DNS dedicados, resolvendo os nomes de domínios dos serviços disponíveis. Estes servidores podem ser disponibilizados em container pelas plataformas livres mencionadas neste trabalho.

Ressaltamos que a avaliação proposta neste trabalho não tem o foco na segurança dos servidores, tampouco nas implementações de segurança de rede. Levamos em consideração que este tipo de estudo necessita realização de um foco exclusivo, contudo este projeto faz parte da composição de um cenário real onde passa a prover de serviços entre cliente-servidor.

#### 4.1.1 Console Hyper-V

A utilização do console Hyper-V permite criar sistemas virtuais gerenciáveis, o SO escolhido para montar o cenário foi em Linux Ubuntu e o CentOS com os recursos adotados para interligar os sistemas virtuais a fim de visualizar a computação paralela. Foram disponibilizados na mesma rede os computadores virtuais para que os hosts consigam estabelecer acesso uma aplicação Web provisionado com a função de Failover. Há também a presença do servidor DNS para resolver nomes de domínios aplicados aos servidores Web e avaliar a composição da computação paralela atrelado às técnicas de disponibilidade adotadas na iteração das máquinas virtuais, aplicadas à obtenção de redundâncias entre servidores.

#### 4.1.2 Rede

No quesito de conectividade, o host hospedeiro (HOST - SERVER01) possui apenas uma única placa de rede física que estabelece interação para os sistemas virtuais. Em uma implementação desta maneira pode ocorrer falhas de conexões nos sistemas virtuais por não conter redundância de portas Ethernet, o que seria mais coerente a para disposição de serviços voltada para computação paralela. Para que o estudo tenha fins de avaliação em um ambiente Hyper-V e redundância em servidores e seus serviços de aplicação, os nós da rede são configurados em uma rede virtual para as VMs onde elas passam a fazer uso da mesma porta de entrada de conexão virtual. Para a troca de dados entre os hosts

virtuais bem como toda rede, foi estabelecido um comutador virtual o qual foi aplicado os servidores DNS primários, secundários, comutar entre os servidores Web primário e secundário bem como aos hosts físicos de backup, criado para a avaliação da aplicação gerada dos servidores Web na rede WI-FI fora do ambiente virtual.

#### 4.1.3 Metodologia de Rede

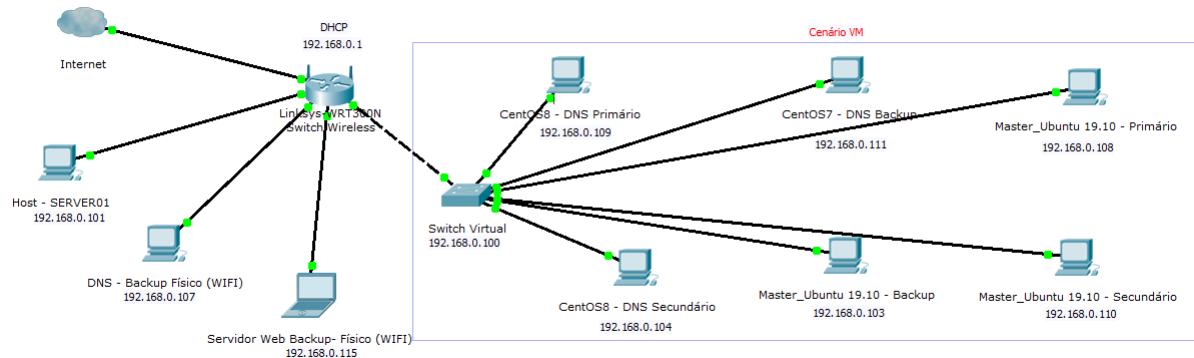


Figura 11 – *Mapa de Rede*

#### 4.1.4 VMs

Para os sistemas abordados, Linux na versão(Ubuntu 19.10) para servidores Web, em todos os sistemas isolados das máquinas virtuais foi utilizado o pacote de serviço Bind9, que hospeda e cria zonas de domínios, Apache cria o cenário com os arquivos Web (HTML, CSS, JS) Docker para dispor uma imagem do Apache subindo um serviço em container dentro da máquina virtual alocando uma imagem do sistema com os arquivos da página adotada para domínio desta estrutura abordada em questão, FirewallD para configuração de portas e serviços na rede. Para resolução de nomes de domínios foi utilizado o CentOS nas versões 7 e 8 com pacote de serviço nameD para publicar na rede os domínios o qual foi implementado o ambiente de resolução de nomes fazendo a proposta de redundância e disponibilidade dos serviços dessas máquinas virtuais neste trabalho.

#### 4.1.5 Serviço Docker

Nas máquinas virtuais representada por WebServer, apresenta a configuração e a utilização do Docker container após a instalação dos pacotes e configurações, com o comando (`docker run -d -rm -name no-paladar -p 8080:80 -v /home/Documento/No-Paladar:/var/www/html php:7.4-apache`). `run` - para rodar o serviço do Docker. `-rm` - Remove qualquer duplicidade de container que esteja com mesmo IP ou mesmo nome. `-name` - Atribui um nome ao container.

```
● docker.service - Docker Application Container Engine
   Loaded: loaded (/lib/systemd/system/docker.service; enabled; vendor preset: enabled)
   Active: active (running) since Thu 2020-10-15 09:42:32 -03; 55min ago
     Docs: https://docs.docker.com
  Main PID: 827 (dockerd)
    Tasks: 18
   Memory: 73.3M
    CGroup: /system.slice/docker.service
            └─ 827 /usr/bin/dockerd -H fd:// --containerd=/run/containerd/containerd.sock
               3597 /usr/bin/docker-proxy -proto tcp -host-ip 0.0.0.0 -host-port 8080 -container-ip 172.17.0.2 -containe

out 15 09:41:41 master dockerd[827]: time="2020-10-15T09:41:41.416559870-03:00" level=error msg="Failed to load containe
out 15 09:41:41 master dockerd[827]: time="2020-10-15T09:41:41.522703599-03:00" level=error msg="Failed to load containe
out 15 09:42:01 master dockerd[827]: time="2020-10-15T09:42:01.445999348-03:00" level=info msg="Removing stale sandbox
out 15 09:42:03 master dockerd[827]: time="2020-10-15T09:42:03.413655968-03:00" level=warning msg="Error (Unable to co
out 15 09:42:07 master dockerd[827]: time="2020-10-15T09:42:07.443362711-03:00" level=info msg="Default bridge (docker
out 15 09:42:12 master dockerd[827]: time="2020-10-15T09:42:12.235818416-03:00" level=info msg="Loading containers: do
out 15 09:42:29 master dockerd[827]: time="2020-10-15T09:42:29.569074966-03:00" level=info msg="Docker daemon" commit=
out 15 09:42:29 master dockerd[827]: time="2020-10-15T09:42:29.799766171-03:00" level=info msg="Daemon has completed i
out 15 09:42:32 master dockerd[827]: time="2020-10-15T09:42:32.784536579-03:00" level=info msg="API listen on /run/doc
out 15 09:42:32 master systemd[1]: Started Docker Application Container Engine.
```

Figura 12 – Serviço Docker

A VM passa a responder em container Docker usando uma imagem do Apache e o serviço do Bind9 para resolver o domínio da própria VM Web para entregar o provisionamento da aplicação ao servidor DNS, o container passa a responder pelo IP 172.17.0.2 cada serviço deste é atribuído um IP de comunicação entre a imagem do container criada e a máquina virtual, a porta destinada para a aplicação foi a 8080:80 onde passa a responder por um serviço HTTP como é mostrada.

## 4.2 DIRETÓRIOS E AMBIENTES DE CONFIGURAÇÃO

### 4.2.1 Servidor Web

No ambiente da cada VM adotada, no diretório /etc/bind consiste as configurações de domínios aplicados para esses servidores, os arquivos de configuração como o arquivo (db.108.0.168.192) para fazer o reverso da aplicação em container submetido ao serviço Docker, o arquivo (db.no-paladar.com.br) que constitui a configuração da zona de domínio da aplicação Web em questão e o arquivo (named.conf.default.zones) fica alocado as configurações dessas zonas criadas e configuradas.

A estrutura definida para esta implementação de servidor Web fica no diretório em /var/www/html, onde cada VM passa a responder aos arquivos configurados neste ambiente, o pacote de serviço utilizado, passa a criar as instâncias com Docker que por sua vez cria uma imagem desses arquivo ao iniciar com serviço do Apache para que passam a responder ao serviço Web.

As configurações dos diretórios de aplicação são importantes para um bom funcionamento das diretrizes das funções do ambiente do Hyper-V ou qualquer outro cenário. Muitos serviços dependem das configurações dos diretórios, dos arquivos contidos e das variáveis de ambiente. Sistemas Open-Source contam com uma variedade ampla de confi-

```

64 bytes from 192.168.0.108 (192.168.0.108): drwxr-xr-x 2 root root 4096 set 28 09:12 css
64 bytes from 192.168.0.108 (192.168.0.108): -rw-r--r-- 1 root root 3600 set 28 08:11 give.html
64 bytes from 192.168.0.108 (192.168.0.108): drwxr-xr-x 2 root root 4096 set 28 08:56 img
64 bytes from 192.168.0.108 (192.168.0.108): -rwxr-xr-x 1 root root 4586 ago 26 15:26 index.html
64 bytes from 192.168.0.108 (192.168.0.108): root@master:/var/www/html# cd /etc/bind
64 bytes from 192.168.0.108 (192.168.0.108): root@master:/etc/bind# ls -all
64 bytes from 192.168.0.108 (192.168.0.108): total 72
64 bytes from 192.168.0.108 (192.168.0.108): drwxr-sr-x 2 root bind 4096 out 2 09:13 .
64 bytes from 192.168.0.108 (192.168.0.108): drwxr-xr-x 138 root root 12288 out 15 10:26 ..
64 bytes from 192.168.0.108 (192.168.0.108): -rw-r--r-- 1 root root 2761 mai 15 09:09 bind.keys
64 bytes from 192.168.0.108 (192.168.0.108): -rw-r--r-- 1 root root 237 jun 27 2019 db.a
64 bytes from 192.168.0.108 (192.168.0.108): -rw-r--r-- 1 root bind 306 set 10 08:34 db.108.0.168.192
64 bytes from 192.168.0.108 (192.168.0.108): -rw-r--r-- 1 root root 271 jun 27 2019 db.127
64 bytes from 192.168.0.108 (192.168.0.108): -rw-r--r-- 1 root root 237 jun 27 2019 db.255
64 bytes from 192.168.0.108 (192.168.0.108): -rw-r--r-- 1 root root 353 jun 27 2019 db.empty
64 bytes from 192.168.0.108 (192.168.0.108): -rw-r--r-- 1 root root 270 jun 27 2019 db.local
64 bytes from 192.168.0.108 (192.168.0.108): -rw-r--r-- 1 root bind 306 out 2 09:13 db.no-paladar.com.br
64 bytes from 192.168.0.108 (192.168.0.108): -rw-r--r-- 1 root bind 463 jun 27 2019 named.conf
64 bytes from 192.168.0.108 (192.168.0.108): -rw-r--r-- 1 root bind 672 set 8 09:57 named.conf.default-zones
64 bytes from 192.168.0.108 (192.168.0.108): -rw-r--r-- 1 root bind 165 jun 27 2019 named.conf.local
64 bytes from 192.168.0.108 (192.168.0.108): -rw-r--r-- 1 root bind 846 jun 27 2019 named.conf.options
64 bytes from 192.168.0.108 (192.168.0.108): -rw-r----- 1 bind bind 77 jul 24 12:52 rndc.key
64 bytes from 192.168.0.108 (192.168.0.108): -rw-r--r-- 1 root root 1317 jun 27 2019 zones.rfc1918
root@master:/etc/bind# paint

```

Figura 13 – Ambiente de Diretório Servidor Web

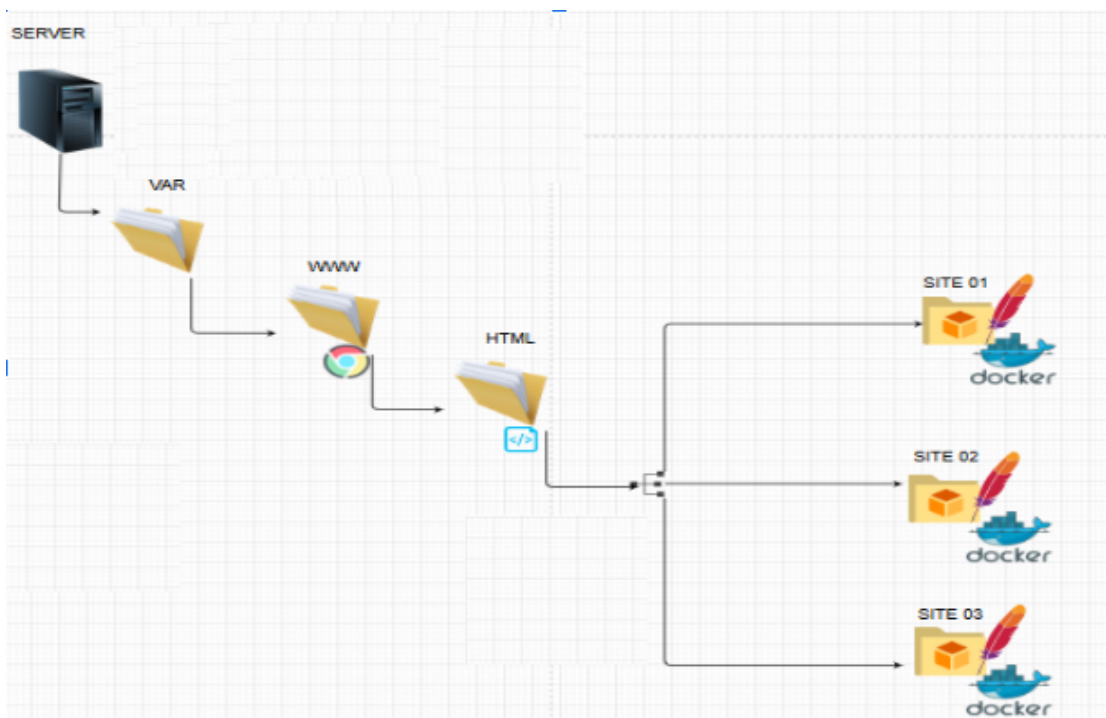


Figura 14 – Metodologia de Diretório

gurações, partindo de cada escopo do problema, as aplicações de arquivos em um sistema de código aberto provê estruturas computacionais que utilizam o núcleo desses sistemas para atrelar uma grande variedades de serviços (STALLMAN, 2020).

#### 4.2.2 Servidor DNS

No ambiente do servidor DNS, os arquivos de configuração ficam dispostos em /var/named, com os arquivos previamente estabelecidos e configurados no arquivo (db.109.0.168.192)



onde realiza o reverso do domínio deste servidor, resolvendo a zona criada no (db.no-paladar.com.br) com provisionamento para publicar na rede o servidor Web.

```
[root@dns dns]# ^C
[root@dns dns]# nano /var/named/
[root@dns dns]# cd /var/named/
[root@dns named]# ls -all
total 32
drwxrwx--T. 5 root named 197 Oct 15 09:10 .
drwxr-xr-x. 22 root root 4096 Sep 3 07:29 ..
drwxrwx---. 2 named named 101 Oct 11 07:16 data
-rwxrwxrwx. 1 root root 278 Sep 4 07:51 db.109.0.168.192
-rwxrwxrwx. 1 root root 373 Oct 7 07:28 db.no-paladar.com.br
drwxrwx---. 2 named named 60 Oct 15 09:10 dynamic
-rw-r-----. 1 root named 2253 Jul 7 10:14 named.ca
-rw-r-----. 1 root named 152 Jul 7 10:14 named.empty
-rw-r-----. 1 root named 152 Jul 7 10:14 named.localhost
-rw-r-----. 1 root named 168 Jul 7 10:14 named.loopback
-rw-r-----. 1 root root 3314 Aug 12 11:48 named.root
drwxrwx---. 2 named named 6 Jul 7 10:14 slaves
[root@dns named]#
```

Figura 15 – Ambiente de Diretório Servidor DNS

#### 4.2.3 Arquivos de Script

Os arquivos de implementação dos servidores e serviços são organizados de forma estruturada, trazendo a possibilidade de automação e robustez. O Docker traz uma composição de script chamada de “docker-compose” onde toda parte de servidores, portas, serviços são configurados para que de uma forma ágil consiga prover os serviços e as funcionalidades que a composição dos orquestradores dos containeres venham desempenhar.

Para essa aplicação são utilizados os arquivos do tipo .yaml, uma linguagem de marcação que provê um dos recursos diretamente com HTML. Uma opção para automação dos servidores podem ser implementadas pelo desenvolvedor do ambiente. O arquivo do tipo .sh (Shell Script) faz com que todo o ambiente seja configurado de forma automatizada e escalável.

#### 4.2.4 resolv.conf

Nas configurações onde cada computador nó da rede passa a responder como servidor, no diretório /etc/resolv.conf, configurar os nameservers e search de resolução de nomes de domínio dessas estações virtuais.

#### 4.2.5 Zonas de Domínio Web

As máquinas virtuais passam a responder as suas zonas com servidor Web com as configurações entre os serviços do Docker e o Apache, acontece no arquivo (db.no-paladar.com.br) criando um domínio entre o container Docker estabelecido.





nele é atribuído diretamente o IP ao nome determinado para o domínio como exemplo (master.no-paladar.com.br).

```
GNU nano 4.3 db.108.0.168.192
$TTL 604800
@      IN SOA  master.no-paladar.com.br. webmaster.no-paladar.com.br. (
1      ; Serial
86400  ; Refresh
86400  ; Retry
86400  ; Expire
86400 ) ; Negative Cache TTL
;
@      IN NS   master.no-paladar.com.br.
192.168.0.108 IN PTR master.no-paladar.com.br.

11 linhas lidas
^G Obter Ajuda Gravar Onde está? Recort txt Justificar Pos at
^X Salir ^R Ler o arq Substituir Colar txt ^T Verif ortog ^ Ir p/
```

Figura 18 – *Reverso Web*

#### 4.2.7 Zona de Domínio DNS

As configurações entre os servidores pouco diferem quando tratamos do ambiente e do arquivo no sistema, porém algumas configurações fazem parte de cada processo em questão, como o tipo de pacote de serviço usado para publicar um DNS na rede usado foi o NameD, recurso este que já está embarcado no CentOS. Os arquivos de domínio ficam no diretório /var/named onde foram criado o (db.no-paladar.com) em que o DNS vai passar a responder pelo nome resolvendo o IP do servidor Web na rede e o arquivo (db.109.0.168.192) faz o reverso autenticando o nome de domínio escolhido com o servidor Web provendo os serviços estabelecidos.

```
$TTL 86400
@      IN SOA  dns.no-paladar.com.br. webmaster.no-paladar.com.br. (
2      ; Serial
86400  ; Refresh
86400  ; Retry
86400  ; Expire
86400 ) ; Negative Cache TTL
;
@      IN NS   dns.no-paladar.com.br.
dns    IN A    192.168.0.109

master IN A    192.168.0.108
webserver IN A  192.168.0.110
no-paladar.com.br. IN A  192.168.0.108
no-paladar.com.br. IN A  192.168.0.110
www    IN CNAME master
```

Figura 19 – *Ambiente de Diretório Servidor DNS*

No arquivo de domínio do DNS aplica ao nome de resolução de domínio (dns.no-paladar.com.br) e com IP 192.168.0.109, já que o mesmo vai resolver o domínio do servidor

Web na rede publicando-o a hospedagem (no-palada.com.br) . O servidor DNS passa a provê os recursos do servidor Web através do arquivo db.no-paladar.com.br.

```

dns@dns:/etc/named
File Edit View Search Terminal Help
GNU nano 2.9.8 /etc/named.conf

session-keyfile "/run/named/session.key";

/* https://fedoraproject.org/wiki/Changes/CryptoPolicy */
include "/etc/crypto-policies/back-ends/bind.config";
};

logging {
    channel default_debug {
        file "data/named.run";
        severity dynamic;
    };
};

zone "." IN {
    type hint;
    file "named.ca";
};

include "/etc/named.rfc1912.zones";
include "/etc/named.root.key";

zone "no-paladar.com.br" IN {
    type master;
    file "/var/named/db.no-paladar.com.br";
};

zone "109.0.168.192.in-addr.arpa" IN {
    type master;
    file "/var/named/db.109.0.168.192";
};

```

Figura 20 – Zona DNS

Nas configurações criadas para a zona do servidor DNS, se mantém em (no-paladar.com.br) com configurações em diretório distinto do servidor Web. Este por sua vez resolve aos servidores Web o DNS para rede, a sua zona reversa criada 109.0.168.192 faz a autenticação validando o IP do servidor Web ao nome de domínio publicando a aplicação na rede.

```
* named.service - Berkeley Internet Name Domain (DNS)
Loaded: loaded (/usr/lib/systemd/system/named.service; enabled; vendor preset: disabled)
Active: active (running) since Thu 2020-10-15 09:10:07 EDT; 39min ago
Process: 3720 ExecStop=/bin/sh -c /usr/sbin/rndc stop > /dev/null 2>&1 || /bin/kill -TERM $MAINPID (code=exited, status=0/SUCCESS)
Process: 3733 ExecStart=/usr/sbin/named -u named -c ${NAMED_CONF} :OPTIONS (code=exited, status=0/SUCCESS)
Process: 3732 ExecStartPre=/bin/bash -c if [ ! "$DISABLE_ZONE_CHECKING" == "yes" ]; then /usr/sbin/named-checkconf -z "${NAMED_CONF}" ; fi (code=exited, status=0/SUCCESS)
Main PID: 3738 (named)
Tasks: 4 (limit: 10689)
Memory: 88.2M
CGroup: /system.slice/named.service
└─3738 /usr/sbin/named -u named -c /etc/named.conf -4
```

```
Oct 15 09:10:07 dns named[3738]: zone 1.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.ip6.arpa/IN: loaded serial 0
Oct 15 09:10:07 dns named[3738]: zone 1.0.0.127.in-addr.arpa/IN: loaded serial 0
Oct 15 09:10:07 dns named[3738]: zone 109.0.168.192.in-addr.arpa/IN: loaded serial 1
Oct 15 09:10:07 dns named[3738]: zone no-paladar.com.br/IN: loaded serial 2
Oct 15 09:10:07 dns named[3738]: zone localhost/IN: loaded serial 0
Oct 15 09:10:07 dns named[3738]: all zones loaded
Oct 15 09:10:07 dns named[3738]: running
Oct 15 09:10:07 dns systemd[1]: started Berkeley Internet Name Domain (DNS).
Oct 15 09:10:07 dns named[3738]: managed-keys-zone: Key 20326 for zone . acceptance timer complete: key now trusted
Oct 15 09:10:08 dns named[3738]: resolver priming query complete
```

Figura 21 – *Servico NameD do DNS*

Após os serviços previamente estabelecidos, nesta imagem notamos o `named.service`

em execução no servidor DNS, suas zonas de configuração carregadas e provendo para rede a aplicação do servidor Web em questão.

## 5 CONCLUSÃO

Nesta proposta de TCC buscamos analisar os recursos que podem estar presentes em cenários de comparação nos ambientes de computação de alto desempenho. Este trabalho não possui o foco de apontar os recursos presentes em grandes estruturas empresariais, contudo, salientamos que muitos destes ambientes trabalham em computação em nuvem e, de alguma forma, operam espelhados conforme os cenários aqui descritos. Esta proposta busca demonstrar que mesmo computadores não focados em alto desempenho podem prover serviços de redundância das soluções providas pelos servidores de grandes empresas.

Ambientes de alto desempenho buscam sempre uma otimização de recursos computacionais para a disponibilização do tempo ocioso de uma máquina. Com os resultados encontrados neste trabalho, é possível identificar que a execução de alguns testes específicos pode não ser impactado se a migração em tempo-real necessitar ocorrer, visto que toda a capacidade de processamento está dedicada ao uso da CPU virtual. Logo, o hipervisor Hyper-v também foi considerado como estratégia para ambientes virtuais que demandem alto processamento e alta disponibilidade. Como trabalhos futuros, podemos estender esta pesquisa a diversos outros hipervisores, possibilitando uma grande gama de análises entre ferramentas de virtualização gratuitas e pagas em diversos cenários de aplicação.

### *TRABALHOS FUTUROS*

Para as próximas pesquisas envolvendo esta proposta, identificamos a oportunidade da implementação do docker-compose como parte de estrutura de um script automatizando e provendo os serviços de forma mais escalável. Ainda com o objetivo do provisionamento de recursos, podemos estabelecer segurança digital para que tanto os servidores, computadores virtuais bem como a rede não fique vulnerável para ataques e invasões aplicando serviços de firewall com base em políticas de segurança. Outros hipervisores podem compor como proposta de gerenciamento de máquinas virtuais bem como aplicações com outras plataforma de distribuição de sistemas livres, outros tipos de servidores como de banco de dados pode está fazendo parte de avaliação na manipulação de armazenamento para entrada e saída de dados, aplicações para clusterização poderá compor para carga de processamento dividindo processos em mais componentes para rede e está avaliando este tipo de ambiente.

## REFERÊNCIA BIBLIOGRÁFICA

BEOWULF.ORG. **The Beowulf Cluster Site: Scyld Computing Corporation..**

Disponível em: <<https://www.beowulf.org/>>. Acessado em 18 de fevereiro de 2020

BOOKMAN, C. **Linux Clustering: Building and Maintaining.** ed. New Riders, 2002.

BROWN, R. G. **What makes a Cluster Beowulf? Duke University Physics Department,** 2006.

DUNBAR, B. **NASA** Acessado em 18 de fevereiro de 2020. Disponível online: <http://www.nasa.gov/>.

GEIST, A. et al. **PVM: parallel virtual machine, Oak Ridge National Labs** 2001.

Acessado em 18 de fevereiro de 2020. Disponível on-line: <http://www.csm.ornl.gov/pvm/>.

GROP, W; LUSK, E. **MPI: The Message Passing Interface (MPI) Standard, Argonne National Laboratories,** Acessado em 28 de julho de 2020. Disponível on-line: <http://www-unix.mcs.anl.gov/mpi/>.

PITANGA, M. **Construindo Supercomputadores com Linux,** ed. Brasport, 2002.

SOLTESZ, S. et al. **Container-based operating system virtualization: A scalable, high-performance alternative to hypervisors. In Proceedings of the 2nd ACM SIGOPS European Conference on Computer Systems,** 2007.

SILBERCHATZ, A; GALVIN, P. **Sistemas Operacionais,** 1ª edição. Campus, Rio de Janeiro, 2001.

STALLMAN, R. **Linux e o sistema GNU,** 2020. Acessado 14/10/2020. Disponível em: <https://www.gnu.org/gnu/linux-and-gnu.pt-br.html>