Thales Alexsander Closs

Virtualização em ambiente corporativo com ferramentas *open-source*

Cuiabá - MT

2021

Thales Alexsander Closs

Virtualização em ambiente corporativo com ferramentas *open-source*

Trabalho de Conclusão de Curso do Curso de Bacharelado em Engenharia da Computação apresentado ao Departamento da Área de Informática do *Campus* Cuiabá – Cel. Octayde Jorge da Silva do Instituto Federal de Educação, Ciência e Tecnologia de Mato Grosso, como requisito para a obtenção do título de Bacharel em Engenharia da Computação.

Instituto Federal de Educação, Ciência e Tecnologia de Mato Grosso (IFMT)

Campus Cuiabá - Cel. Octayde Jorge da Silva

Curso de Bacharelado em Engenharia da Computação

Orientador: Prof. Esp. Giuliano Robledo Zucoloto Moreira

Cuiabá - MT

2021

Thales Alexsander Closs

Virtualização em ambiente corporativo com ferramentas *open-source*/ Thales Alexsander Closs. – Cuiabá - MT, 2021-

70 p. : il. (algumas color.) ; 30 cm.

Orientador: Prof. Esp. Giuliano Robledo Zucoloto Moreira

Trabalho de Conclusão de Curso (Graduação) – Instituto Federal de Educação, Ciência e Tecnologia de Mato Grosso (IFMT)
Campus Cuiabá - Cel. Octayde Jorge da Silva
Curso de Bacharelado em Engenharia da Computação, 2021.

1. virtualização. 2. alta disponibilidade. 3. *Proxmox*. 4. *Ceph*. 5. hiperconvergência. I. Giuliano Robledo Zucoloto Moreira. II. Instituto Federal de Educação, Ciência e Tecnologia de Mato Grosso (IFMT). Campus Cuiabá - Cel. Octayde Jorge da Silva. Departamento da Área de Informática. Curso de Bacharelado em Engenharia da Computação. IV. Virtualização em ambiente corporativo com ferramentas *open-source*.

CDU 02:141:005.7

Thales Alexsander Closs

Virtualização em ambiente corporativo com ferramentas *open-source*

Trabalho de Conclusão de Curso do Curso de Bacharelado em Engenharia da Computação apresentado ao Departamento da Área de Informática do *Campus* Cuiabá – Cel. Octayde Jorge da Silva do Instituto Federal de Educação, Ciência e Tecnologia de Mato Grosso, como requisito para a obtenção do título de Bacharel em Engenharia da Computação.

Trabalho aprovado. Cuiabá - MT, 09 de abril de 2021:

Prof. Esp. Giuliano Robledo Zucoloto Moreira
Orientador

Prof.ª Dr.ª Juliana Fonseca Antunes
Membro da banca

Prof. Dr. Aldo Antônio Vieira da Silva
Membro da banca

Cuiabá - MT
2021

Este trabalho é dedicado a toda minha família.

Agradecimentos

A Deus, pelo dom da vida, pela saúde e pelas oportunidades colocadas em meu caminho.

Aos meus pais, pelo amor incondicional, pela educação e princípios a mim passados e incentivo para sempre continuar.

A minha namorada Renatta, pelo apoio, companheirismo, dedicação e compreensão ao longo dessa jornada.

A vó Palmira, por todo suporte oferecido durante a realização do trabalho.

A todos os familiares que sempre apoiaram e incentivaram a minha jornada.

Ao Dr. Paulo, pela confiança depositada ao longo dos anos de trabalho, pelos conselhos e apoio fornecidos.

Ao professor, orientador e amigo, Giuliano Robledo Zucoloto Moreira, por todo apoio fornecido no caminho percorrido, pelo tempo e paciência a mim dedicados durante todo o período de orientação.

A professora Mônica Cristiane Moreira Crispim, pelo incentivo e contribuições na elaboração deste trabalho.

Aos professores, que contribuíram para uma formação acadêmica de qualidade.

Ao Departamento da Área de Informática e ao Instituto Federal de Educação, Ciência e Tecnologia de Mato Grosso - *Campus* Cel. Octayde Jorge da Silva, pelas condições fornecidas para a minha formação.

E por fim, agradeço a todos aqueles que contribuíram, de alguma forma, para a realização deste trabalho.

"Se fizéssemos todas aquelas coisas
de que somos capazes, nós nos
surpreenderíamos a nós mesmos."
(Thomas Edison)

Resumo

Este trabalho aborda o procedimento de implantação de um ambiente de virtualização utilizando as ferramentas *open-source Proxmox* e *Ceph* em um Cartório de Mato Grosso para atendimento a requisitos do provimento 74/2018 do Conselho Nacional de Justiça (CNJ). Foi realizada uma pesquisa acerca do tema virtualização e suas necessidades para implementação com alta disponibilidade e segurança de armazenamento (*backup*), seguida de um estudo de caso que produziu como resultado uma solução de virtualização em alta disponibilidade baseada em hiperconvergência, atendendo fielmente ao cumprimento de requisitos legais do provimento no tocante a um conjunto de requisitos de tecnologia da informação.

Palavras-chaves: Virtualização. Alta disponibilidade. *Proxmox*. *Ceph*. Hiperconvergência.

Abstract

This paper addresses the procedure for implementing a virtualization environment using the open-source tools Proxmox and Ceph in a Notary's Office of the State of Mato Grosso to meet the requirements of provision 74/2018 of the National Council of Justice (CNJ). Research was carried out on the topic of virtualization and its needs for implementation with high availability and storage security (backup), followed by a case study that resulted in a high availability virtualization solution based on hyperconvergence, faithfully meeting compliance with legal requirements for the provision of a set of information technology requirements

Key-words: Virtualization. High availability. Proxmox. Ceph. Hyperconvergence.

Lista de Figuras

Lista de abreviaturas e siglas

AGPL	*Affero General Public License*
API	*Application Programming Interface*
CephFS	*Ceph File System*
CIFS	*Common Internet File System*
CLI	*Command Line Interface*
CNJ	Conselho Nacional de Justiça
CPU	*Central Processing Unit*
CRUSH	*Controlled Replication Under Scalable Hashing*
DAS	*Direct Attached Storage*
FC	*Fibre Channel*
GB	*Gigabyte*
GbE	*Gigabit Ethernet*
Gbps	*Gigabit per second*
GLP	*General Public License*
HA	*High Availability*
HBA	*Hot Bus Adapter*
HDD	*Hard Disk Drive*
HPC	*High Performance Computing*
IEEE	*Institute of Electrical and Electronic Engineers*
IP	*Internet Protocol*
ISA	*Instruction Set Architecture*
ISO	*International Organization for Standardization*
JVM	*Java Virtual Machine*

KVM	*Kernel-based Virtual Machine*
LACP	*Link Aggregation Control Protocol*
LAG	*Link Aggregation*
LAN	*Local Area Network*
LB	*Load Balancing*
LIBRADOS	*Library to access RADOS*
LXC	*Linux Containers*
LZO	*Lempel Ziv Oberhumer*
MAC	*Media Access Control*
Mbps	*Megabits per second*
MDS	*Metadata Servers*
MTU	*Maximum Transfer Unit*
NAS	*Network Attached Storage*
NFS	*Network File System*
OSD	*Object Storage Daemon*
OSI	*Open System Interconnection*
PC	*Personal Computer*
PG	*Placement Group*
POSIX	*Portable Operating System Interface*
PSU	*Power Supply Unit*
RADOS	*Reliable Autonomous Distributed Object Store*
RAID	*Redundant Array of Independent Disks*
RAM	*Random Access Memory*
RBD	*Rados Block Device*
RGW	*Rados Gateway*
RPM	Rotação por minuto

SAN	*Storage Area Network*
SDS	*Software Defined Storage*
SFP	*Small Form-factor Pluggable*
SMB	*Server Message Block*
SO	Sistema Operacional
STP	*Spanning Tree Protocol*
TB	*Terabyte*
TCP	*Transmission Control Protocol*
TI	Tecnologia da Informação
UPS	*Uninterruptible Power Supply*
VLan	*Virtual Local Area Network*
VM	*Virtual Machine*
VMM	*Virtual Machine Monitor*

Sumário

1 Introdução

A dinâmica do mundo moderno necessita, cada vez mais, de serviços digitais e estes devem estar disponíveis sempre que solicitados. Não é incomum falhas de hardware causarem indisponibilidade de aplicações (TOLFO, 2019). Esta dinâmica faz com que as empresas necessitem de velocidade no provisionamento de serviços e processos rápidos de recuperação de falhas, além da otimização máxima do custo da infraestrutura (CARISSIMI, 2008).

Tendo em vista tais necessidades, a virtualização de servidores surge como uma alternativa a infraestruturas tradicionais, baseadas em servidores físicos. Os avanços tecnológicos, maior poder computacional dos processadores atuais, disponibilidade de redes de dados mais velozes e unidades de armazenamento escaláveis com baixo custo, além de softwares específicos para tal, impulsionaram o uso da tecnologia de virtualização (CARISSIMI, 2008), (TAKEUTI, 2014), (VERAS, 2011).

A essência da virtualização de servidores é transformar "um servidor físico em vários servidores virtuais" (VERAS, 2011, p. 85), onde cada servidor virtual possui seu próprio hardware virtual e sistema operacional. Tal característica promove melhor aproveitamento de hardware, economia de espaço físico e consumo eficiente de energia elétrica.

Esta é a tecnologia base da computação em nuvem (*cloud computing*), um dos agentes de transformação do mundo digital. A virtualização suporta a disponibilidade dos serviços em nuvem, acelera o processo de lançamento de produtos e promove a redução de custos, com o consumo de computação sob demanda, eliminando a aquisição de hardware (TOLFO, 2019).

Não limitada ao *cloud computing*, a virtualização gera transformação em empresas que contam com infraestrutura própria, sendo elas de qualquer porte (VERAS, 2011), podendo usufruir de seus recursos mais notáveis, entre eles "[...] segurança, confiabilidade e disponibilidade, custo, adaptabilidade, balanceamento de carga e suporte a aplicações legadas" (MATTOS, 2008, p. 1).

Este trabalho apresenta a virtualização e sua aplicação, por meio de Estudo de Caso, em um Cartório de Mato Grosso utilizando as tecnologias *open-source Proxmox* e *Ceph*. O Estudo de Caso foi realizado com a finalidade de modernizar a infraestrutura e atender exigências do Provimento 74/2018 do Conselho Nacional de Justiça (CNJ), que define os padrões mínimos de tecnologia da informação nos cartórios do País a fim de garantir a segurança, integridade e disponibilidade de dados permitindo a continuidade da atividade notarial e registral no Brasil.

A transformação causada pelos meios digitais impacta diretamente o setor de cartórios, onde a tecnologia vem sendo aplicada para garantir, cada vez mais, a segurança jurídica do cidadão e gerar inovação.

Atualmente os cartórios dependem de sistemas informatizados para realizar suas atribuições, e estes devem estar disponíveis e íntegros a todo tempo, principalmente no Estado de Mato Grosso, onde todos os atos são realizados de forma eletrônica e integrados aos sistemas do Tribunal de Justiça do Estado, além de outros órgãos e entidades.

Considerando as necessidades expostas e as características da virtualização foi verificada a viabilidade de implantar a tecnologia de virtualização com ferramentas *open-source* no ambiente do Cartório, sendo este o objetivo geral do trabalho. Para concretizá-lo foram definidos 5 objetivos específicos, voltados a atender um conjunto pertinente de requisitos legais do provimento 74/2018 do CNJ:

- Oferecer a *High Availability* (HA) dos serviços;
- Disponibilizar um volume de armazenamento para hospedagem das *Virtual Machine*s (VMs) de no mínimo 2 *Terabyte*s (TBs);
- Prover a segurança e disponibilidade de dados;
- Fornecer suporte a processos de *backup* das VMs;
- Disponibilizar o ambiente para produção.

Dado o contexto, foi realizada uma pesquisa bibliográfica, apresentada no Capítulo 2, abordando os conceitos e técnicas de virtualização disponíveis, os recursos necessários para a sua implantação aliada a HA, as opções de armazenamento de dados, a utilização de redes de computadores e o *backup*. No Capítulo 3 são apresentados os materiais utilizados e é descrita implantação da solução de virtualização, desde a preparação do ambiente físico até os testes realizados em VMs. No Capítulo 4 são apresentados os resultados e discussões da implantação realizada no estudo de caso. Por fim é apresentada a Conclusão deste trabalho e propostas de trabalhos futuros.

Nota sobre as marcas registradas usadas neste trabalho

Neste trabalho são citados diversos nomes de produtos que são marcas tradicionais ou possuem marcas registradas, porém apenas com a finalidade de relatar fidedignamente o procedimento acadêmico deste trabalho sem qualquer exploração dos direitos de uso da marca.

2 Referencial teórico

2.1 Virtualização

Para Veras (2011, p. 87), a virtualização pode ser conceituada como o "[...] particionamento de um servidor físico em vários servidores lógicos". E como "[...] uma camada de abstração entre hardware e o software [...]" que protege o acesso direto dos servidores lógicos ao hardware. Estes servidores lógicos são denominados máquinas virtuais.

De acordo com Carissimi (2008), a origem das máquinas virtuais remonta ao início dos anos 70, onde cada *mainframe* possuía seu próprio sistema operacional e era necessário executar um código legado ou software desenvolvido para uma plataforma específica, tal estratégia foi aplicada com sucesso pela IBM na linha de *mainframes* 370 e seus sucessores.

Já na década de 80, com a popularização do *Personal Computer* (*PC*) e os modestos hardwares, a virtualização foi deixada de lado. Até que a linguagem Java e sua máquina virtual trouxeram o tema a tona novamente, aliadas a soluções engenhosas da *VMWare* para suportar a virtualização na arquitetura x86 (LAUREANO; MAZIERO, 2008).

Em sua obra Veras (2011) apresenta pesquisa realizada pela *Forrester Consulting* que constatou o aumento de eficiência da área de Tecnologia da Informação (TI), a melhora do *Time-to-Market* e maior previsibilidade dos serviços de TI das empresas entrevistadas. A pesquisa ainda observou que as companhias deixam de ter maiores ganhos por falta de iniciativas em direção a virtualização, em quase 50% das companhias as taxas de utilização da virtualização são elevadas e o retorno sobre o investimento ocorre em menos de doze meses para a maior parte delas. Os motivos apontados pelas companhias para utilizar a virtualização de servidores foram: a redução de custos com hardware e melhores condições para recuperação de desastres e continuidade do negócio.

Carissimi (2008) apresenta pesquisa realizada pela *Network World* em que as principais razões para a virtualização de servidores são:

> "aumentar a taxa de utilização de servidores, reduzir os custos operacionais de datacenters, melhorar os procedimentos de recuperação de desastres e de backup, criar ambientes mais flexíveis para desenvolvimento e teste de software e reduzir custos de administração de TI" (CARISSIMI, 2008, p. 195).

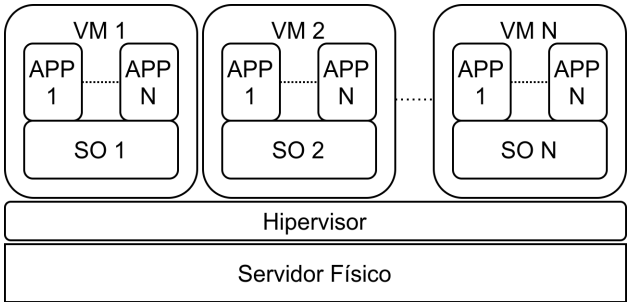
Veras (2011) apresenta pesquisa realizada pelo *IDC Corporate USA*, que reforça a tese da subutilização da capacidade dos servidores. Os números levantados apontam que nas empresas que não adotam a virtualização a capacidade utilizada dos servidores é de 15%, com a adoção a capacidade utilizada pode subir para no mínimo 60%, otimizando os

custos de operação.

2.1.1 Ambiente de virtualização

Um ambiente virtualização pode ser dividido em três partes básicas, sendo elas: o sistema real ou hospedeiro (*host system*), onde estão contidos os recursos reais de software e hardware; a camada de virtualização, também conhecida como *Virtual Machine Monitor* (VMM) ou hipervisor; e o sistema virtual ou convidado (*guest system*) que é executado na VM (LAUREANO; MAZIERO, 2008). A Figura 1 ilustra o ambiente virtual.

Figura 1 – Ambiente de máquina virtual



Fonte: Adaptado de Veras (2011, p. 88, *tradução nossa*).

A camada de virtualização faz interface com o sistema operacional convidado, fornecendo um conjunto de instruções de máquina equivalente ao processador físico. Um servidor físico pode executar várias máquinas virtuais (VERAS, 2011).

Cada máquina virtual possui um ambiente totalmente isolado das demais, com seu próprio sistema operacional e aplicações de usuário. A máquina virtual tem o mesmo comportamento de um computador físico, possui *Central Processing Unit* (CPU), memória *Random Access Memory* (RAM), disco rígido e interfaces de rede. O sistema operacional, as aplicações de usuário e demais computadores de uma rede não tem a capacidade de diferenciar um computador físico de um virtual. Por ser construída por software não existem componentes de hardware, o que dá maior flexibilidade comparada a uma máquina física (VERAS, 2011).

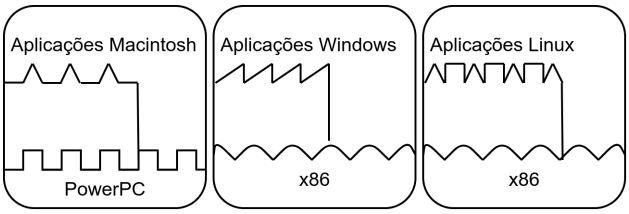
2.1.2 Fundamentos

Um sistema computacional é relativamente complexo devido a interação de uma variedade de componentes de hardware e software. No entanto uma hierarquia com diferentes níveis de abstrações e interfaces, tanto no hardware como no software, permite que cada componente seja visto como um subsistema independente que fornece serviços aos demais (CARISSIMI, 2008), (VERAS, 2011).

Um sistema computacional é formado pelo hardware, software e aplicações. O hardware executa as operações solicitadas pelas aplicações e estas são gerenciadas pelo sistema operacional que também controla o acesso ao hardware (LAUREANO, 2008).

Projetos de hardware e sistemas operacionais, via de regra, são mantidos de forma independente. Logo surgem plataformas operacionais diversas e incompatíveis entre si , como é possível observar na Figura 2 que representa duas arquiteturas e três sistemas operacionais distintos. A arquitetura fornece uma interface bem definida para o controle do hardware pelo sistema operacional e para a execução das aplicações do usuário (LAUREANO, 2008).

Figura 2 – Algumas das plataformas operacionais existentes



Fonte: Adaptado de Laureano (2008, p. 161).

Este estudo aborda a arquitetura x86, por ser predominante na área de servidores. Para compreender o princípio de funcionamento da virtualização é preciso conhecer aspectos fundamentais do hardware e do sistema operacional (VERAS, 2011)

2.1.2.1 Hardware

A arquitetura de hardware pode ser dividida em *Instruction Set Architecture* (ISA), projeto do sistema e microarquitetura. A ISA abstrai o processador através do conjunto de instruções e inclui os modos de endereçamento e os registradores existentes. O projeto de sistema abrange os componentes externos ao processador. A microarquitetura refere-se organização interna do processador. Cada parte oferece uma interface e um conjunto de serviços para as demais (VERAS, 2011).

O conjunto de instruções é uma interface entre o hardware o e sistema operacional, todos os códigos de máquina aceitos pelo processador estão nesta interface (VERAS, 2011). As instruções são dividias em privilegiadas e não privilegiadas, fundamentais para a virtualização. As instruções privilegiadas podem alterar o estado e alocação de recursos do sistema, enquanto as instruções não privilegiadas não podem alterar a alocação e estado de recursos compartilhados por vários processos, como processadores, memória principal e registradores especiais (MATTOS, 2008).

Carissimi (2008) cita a obra de Popek e Goldberg (1974) em que três propriedades são necessárias para que um sistema computacional ofereça suporte a virtualização de forma satisfatória, são elas:

- Eficiência: as instruções de máquina que não comprometem o funcionamento do sistema hospedeiro devem ser executadas diretamente no hardware.

- Controle de recursos: nenhum programa arbitrário pode afetar os recursos do sistema, a máquina virtual deve ter controle total sobre os recursos virtualizados.

- Equivalência: o comportamento de um programa executado em uma máquina virtual deve ser idêntico ao executado em máquina física. Duas exceções podem ser consideradas, o aumento no tempo de execução para algumas instruções e conflitos de acesso a recursos, que devem ser tratados de forma adequada.

Dadas estas propriedades, as instruções ISA são classificadas em: sensíveis, que geram exceções quando executadas em modo de usuário; sensíveis de controle, que permitem alterar recursos do sistema; e sensíveis comportamentais, que depende de configurações de recursos como registradores internos ou modo de execução do processador (CARISSIMI, 2008).

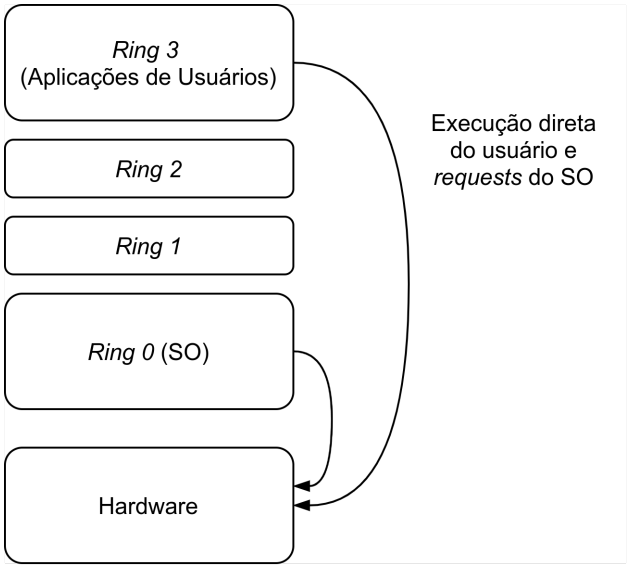
Dois teoremas também foram definidos a partir delas. O primeiro teorema define que qualquer instrução que possa afetar o funcionamento do hipervisor deve passar pelo seu controle. O segundo teorema trata da possibilidade do hipervisor executar uma instância de si mesmo (CARISSIMI, 2008).

Em termos práticos o primeiro teorema é suficiente para um sistema de virtualização, porém não necessário, sendo possível contorná-lo utilizando duas técnicas. A primeira, identifica instruções sensíveis em tempo de execução e as desvia para o hipervisor. Na segunda, o sistema operacional da máquina virtual é modificado e as instruções sensíveis são alteradas para chamadas ao hipervisor (CARISSIMI, 2008).

Tais técnicas são necessárias para contornar problemas da arquitetura x86. Esta arquitetura provê quatro modos de operação, denominados anéis de proteção ou *rings*. Estes vão do 0 ao 3, sendo o 0 o de maior privilégio, onde o Sistema Operacional (SO) é executado, e o 3 o de menor privilégio, onde as aplicações do usuário são executadas. O problema reside em instruções não privilegiadas que também são sensíveis e podem acessar o processador diretamente, sem supervisão do hipervisor o que viola o primeiro teorema (CARISSIMI, 2008),(VERAS, 2011). A Figura 3 apresenta a hierarquia da arquitetura x86.

No entanto, estas não são as únicas alternativas. Com o desenvolvimento de extensões de suporte à virtualização para a arquitetura x86, por parte dos fabricantes de processadores AMD e Intel, é possível contornar de forma mais eficiente a execução de instruções sensíveis (CARISSIMI, 2008).

Figura 3 – Hierarquia da arquitetura x86



Fonte: Adaptado de Veras (2011, p. 104).

2.1.2.2 Sistema Operacional

De acordo com Veras (2011), o sistema operacional é um alocador de recursos, um sistema computacional é composto por muitos recursos e estes são inteiramente gerenciados pelo sistema operacional para a execução das aplicações de usuário.

A operação de um sistema computacional dá-se em dois modos distintos, o modo usuário e o modo supervisor. No modo usuário são executadas as aplicações, elas não podem executar instruções privilegiadas. No modo supervisor podem ser executadas os dois tipos de instrução, este modo tem controle total sobre o processador. O sistema operacional é executado neste modo e quando uma aplicação em modo usuário necessita utilizar recursos do processador ele é responsável por alocar os recursos solicitados e configurar o *bit* de controle para o modo usuário, antes de liberar a aplicação (MATTOS, 2008).

O sistema operacional prove duas interfaces, são elas: chamadas de sistema (*syscalls*), que são oferecidas pelo núcleo do sistema operacional aos processos de usuário, permitindo o acesso controlado a recursos que necessitam de privilégios elevados; e as chamadas de bibliotecas (*libcalls*), estas fornecem funções que simplificam a construção de programas, além de encapsular chamadas do sistema operacional. Cada biblioteca possui interface própria, denominada *Application Programming Interface* (*API*) (LAUREANO; MAZIERO, 2008).

Em um ambiente virtualizado o hipervisor é executado em modo supervisor e as máquinas virtuais em modo usuário. Assim sendo, o hipervisor é responsável por tratar a execução das instruções privilegiadas originadas nas máquinas virtuais (MATTOS, 2008).

2.1.3 Tipos e Classificações

Para Veras (2011), a virtualização pode ser dividida em três classificações e dois tipos.

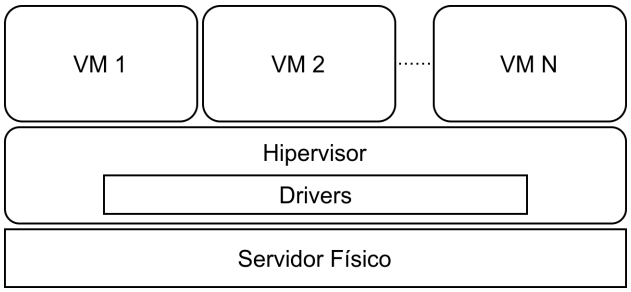
2.1.3.1 Tipo I

Para Laureano (2008, p. 8), este tipo se caracteriza por um "sistema em que o monitor é implementado entre o hardware e os sistemas convidados (guest system)".

Neste tipo, o hipervisor é executado diretamente no hardware, também conhecido como baremetal, sua função é gerenciar e compartilhar os recursos de hardware entre as máquinas virtuais. São exemplos desse tipo VMware ESX Server, Microsoft Hyper-V e Citrix XenServer (VERAS, 2011).

Veras (2011) ainda divide este grupo em monolítico e microkernelizado. No monolítico, o hipervisor detém todos os *drivers* necessários para o funcionamento do hardware emulado das máquinas virtuais, apresentado na Figura 4.

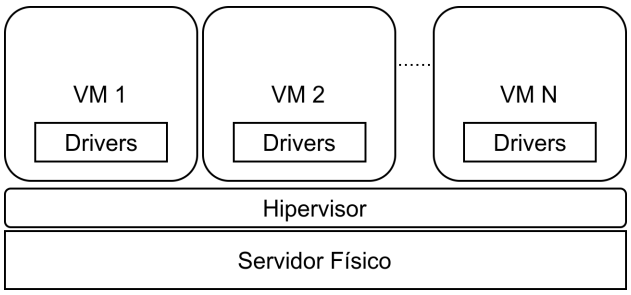
Figura 4 – Hipervisor monolítico



Fonte: Adaptado de Veras (2011, p. 102, *tradução nossa*).

No modelo microkernelizado, os *drivers* residem na própria máquina virtual, sendo mais seguro por oferecer uma superfície menor de ataque, a Figura 5 representa este modelo.

Figura 5 – Hipervisor microkernelizado



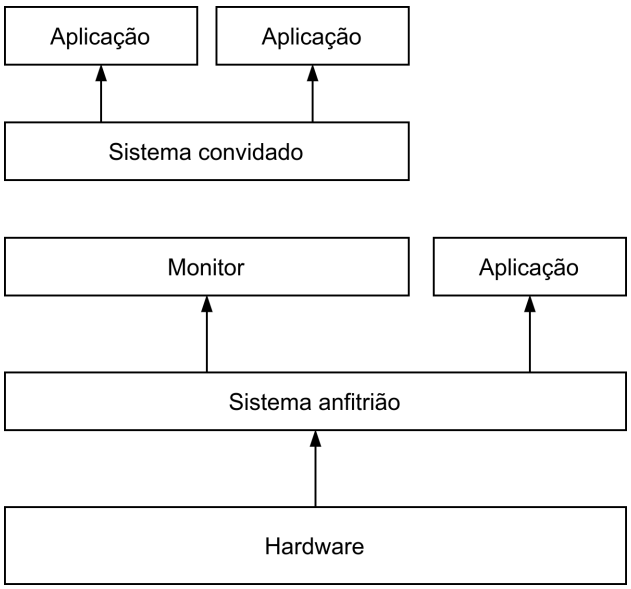
Fonte: Adaptado de Veras (2011, p. 102, *tradução nossa*).

2.1.3.2 Tipo II

Neste tipo, o monitor de virtualização executa sobre um sistema operacional hospedeiro na forma de um processo do sistema real (LAUREANO, 2008).

Na camada de virtualização temos o sistema operacional convidado, o hardware virtual criado pelo sistema operacional anfitrião, com recursos do hardware nativo, como demonstrado na Figura 6 . A máquina virtual Java, *Java Virtual Machine* (JVM), é um exemplo deste modelo (VERAS, 2011).

Figura 6 – Hipervisor tipo II

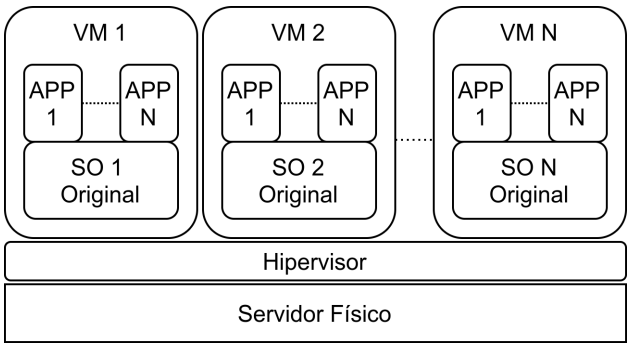


Fonte: Adaptado de Laureano (2008, p. 22).

2.1.3.3 Virtualização total

Na virtualização total, uma réplica virtual do hardware subjacente é criada e disponibilizada para o sistema operacional convidado, não sendo necessária nenhuma alteração no sistema operacional (CARISSIMI, 2008). A Figura 7 representa o modelo.

Figura 7 – Modelo da virtualização total



Fonte: Adaptado de Veras (2011, p. 106, *tradução nossa*).

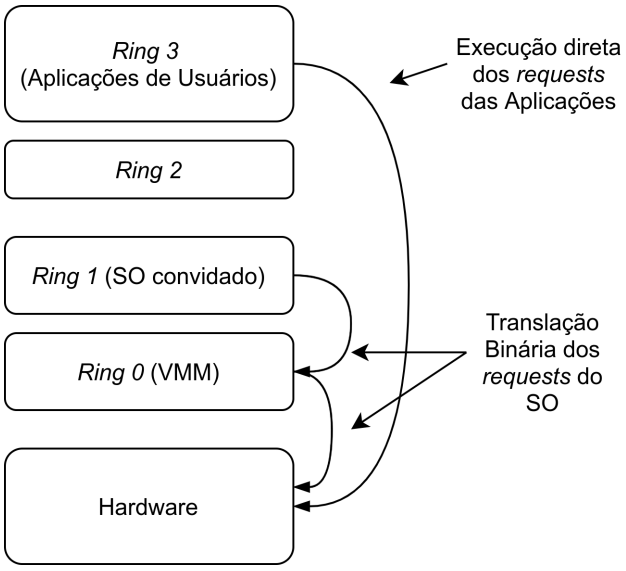
> "Este tipo de VIRTUALIZAÇÃO facilita a migração de máquinas virtuais entre servidores físicos, pois existe total independência das aplicações e dos recursos físicos do servidor" (VERAS, 2011, p. 104).

O hipervisor é responsável por controlar todo o processo de chamada ao hardware, o que pode prejudicar o desempenho (VERAS, 2011).

A utilização de dispositivos e *drivers* genéricos é ponto negativo desta abordagem. A existência de uma diversidade de dispositivos torna difícil a implementação de todos em um modelo virtual, com isso pode ocorrer uma subutilização de recursos do hardware real (CARISSIMI, 2008).

A virtualização total utiliza técnicas de translação binária e execução direta, o que dispensa qualquer alteração no sistema convidado. Desta forma, o sistema convidado passa a operar no *ring* 1 (VERAS, 2011). A Figura 8 mostra este processo.

Figura 8 – Translação binária da virtualização total



Fonte: Adaptado de Veras (2011, p. 105).

Todas as instruções solicitadas pelo sistema convidado devem ser testadas pelo hipervisor, as instruções privilegiadas são interceptadas e emuladas sobre o sistema nativo, evitando alteração do comportamento de todo o sistema (VERAS, 2011).

O hipervisor deve fazer a "conversão" do espaço de endereçamento de memória do sistema hóspede para o sistema hospedeiro, o que acarreta em uma queda de desempenho. Tal medida é necessária para contornar problemas técnicos que surgem devido a forma como os sistemas operacionais são implementados (CARISSIMI, 2008).

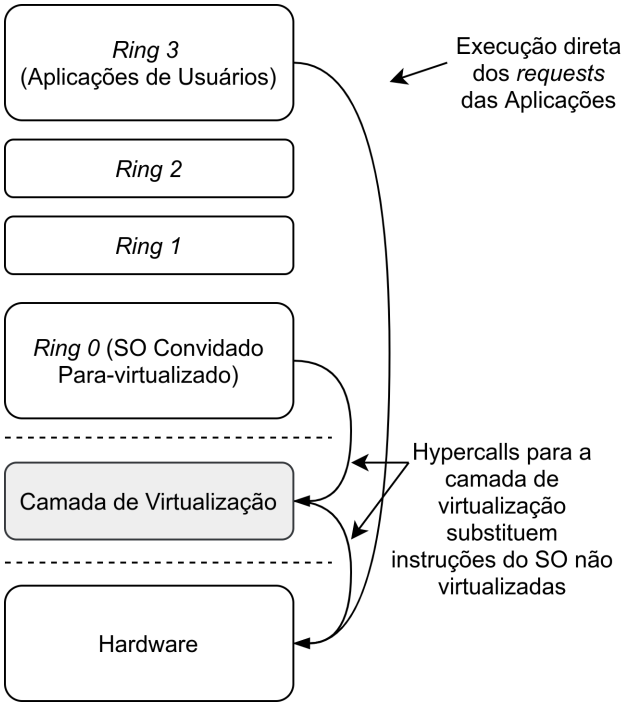
2.1.3.4 Para-virtualização

A para-virtualização surgiu para contornar as desvantagens, em termos de processamento, da utilização da virtualização total. A máquina virtual é criada a partir de

uma abstração do hardware, que não é idêntico ao físico, e os dispositivos de hardware são acessados por *drivers* do próprio hipervisor. Em compensação, o sistema operacional convidado requer modificações (VERAS, 2011).

O sistema convidado é alterado para fazer chamadas ao hipervisor, sempre que precisar executar uma instrução considerada sensível. O hipervisor não precisa testar todas as instruções (CARISSIMI, 2008). A Figura 9 mostra o fluxo de chamadas.

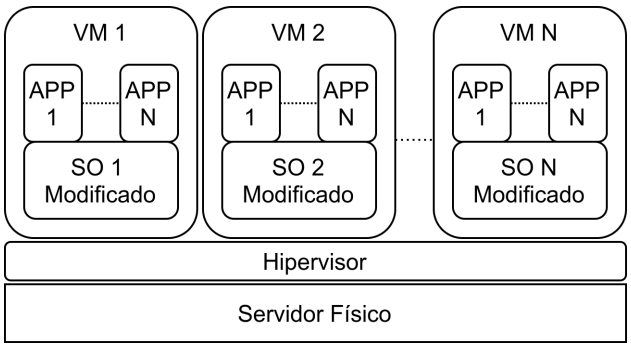
Figura 9 – Chamadas na Para-virtualização



Fonte: Adaptado de Veras (2011, p. 107).

Para Veras (2011), o aspecto que mais dificulta a utilização da para-virtualização é a necessidade de modificação no sistema operacional convidado, para isso, os fornecedores de sistema operacional devem fornecer uma versão adequada à para-virtualização. A Figura 10 representa uma estrutura para-virtualizada.

Figura 10 – Modelo da para-virtualização



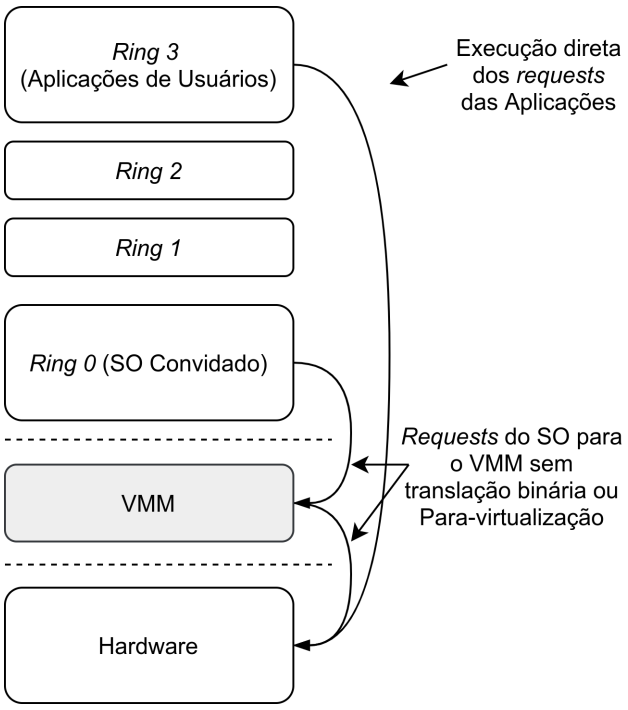
Fonte: Adaptado de Veras (2011, p. 107, *tradução nossa*).

2.1.3.5 Virtualização assistida por hardware

Extensões para a arquitetura x86 foram criadas, pela Intel e AMD, para oferecer suporte à virtualização e melhorar o desempenho global da solução. Com isso, a vantagem de utilizar uma solução para-virtualizada, em termos de processamento, foi superada (VERAS, 2011).

As primeiras versões desenvolvidas não superavam o desempenho da virtualização total baseada em translação binária, ainda assim os fabricantes continuaram a investir no aperfeiçoamento da tecnologia. Somente os novos sistemas baseados na arquitetura X86_64 conseguem utilizar a virtualização assistida por hardware como suporte à virtualização total, melhorando assim o desempenho (VERAS, 2011).

Figura 11 – *Rings* da virtualização assistida por hardware



Fonte: Adaptado de Veras (2011, p. 108).

As tecnologias desenvolvidas pela Intel (Intel VT) e AMD (AMD-V), alteram o modo de operação dos processadores, posicionando o hipervisor um *ring* abaixo do *ring* 0, como pode ser observado na Figura 11. O desempenho é melhorado e o hipervisor tem prioridade total sobre o sistema operacional (VERAS, 2011).

2.1.4 Soluções de mercado

As soluções apresentadas a seguir são apenas as de Tipo I, foco deste trabalho.

2.1.4.1 *VMware*

VMWare é um dos nomes mais conhecidos quando o assunto é a virtualização. Fornece soluções completas, com softwares de virtualização que partem de ambientes *desktop* a ambientes *datacenters*, divididos nas categorias gerenciamento e automação, infraestrutura virtual e plataformas de virtualização (MATTOS, 2008).

O *VMware ESX Server* é a versão comercial do produto *VMware* voltado para o uso em servidores de grande porte. É um hipervisor do Tipo I e possui um sistema operacional próprio e otimizado para gerenciar máquinas virtuais (SILVA, 2007).

O *ESXi* é uma versão mais "leve" do *ESX Server*. O console de gerenciamento presente na versão *ESX* foi removido. Tal console é utilizado para operações locais via linha de comando e na versão *ESXi* dá lugar à ferramentas de gerenciamento remoto, uma tendência dos novos produtos *VMware* (VMWARE, 2009).

Os sistemas *ESX* e *ESXi* são baseados em um *Kernel* Linux customizado pela *VMware*, chamado vmkernel, e fornecem suporte a virtualização total, para-virtualização e a virtualização assistida por hardware (CHENG, 2014), (VMWARE, 2009).

2.1.4.2 *Hyper-V*

O *Hyper-V* é a solução de virtualização da Microsoft, foi introduzida a partir da versão Windows server 2008. É a solução recomendada em uma rede que utiliza somente *hosts* com sistema operacional Windows e serviço de controle de domínio *Active Directory* ativado (CHENG, 2014).

O *Hyper-V* pode ser utilizado nos sistemas Windows e Windows Server. Quando instalada em modo servidor o *Hyper-V* se comporta como um hipervisor do Tipo I e quando instalada em modo aplicação se comporta como um hipervisor do Tipo II. Também é possível instalar o *Hyper-V* como um servidor autônomo, sem a necessidade da instalação do Windows Server (MICROSOFT, 2016).

Quando instalado, no Windows Server o *Hyper-V*, passa a virtualizar o sistema operacional e controlá-lo pela camada de virtualização. Ou seja, o Windows Server tornou-se uma máquina virtual gerenciada pela camada de virtualização do *Hyper-V* (TAKEUTI, 2014).

A versão atual do *Hyper-V* é a 2019. É gratuita para utilização como servidor autônomo, para utilização com Windows Server ou Windows o mesmo deve ser licenciado separadamente (MICROSOFT, 2021).

2.1.4.3 *XenServer*

O *XenServer* é uma solução de virtualização do Tipo I que surgiu do projeto de pesquisa da Universidade de Cambridge chamado XenoServer, seu código é aberto e baseado na *General Public License* (*GLP*) (SILVA, 2007).

O *XenServer* é um projeto muito popular. Inicialmente suportava somente a para-virtualização, em sua versão 3.0 passou a suportar a virtualização total em processadores com suporte a virtualização assistida por hardware (MATTOS, 2008).

Em 2007 a Citrix adquiriu a *XenoSource*, mantenedora do projeto, e passou a dar suporte ao projeto, além de fornecer uma versão empresarial da solução (CONNOR, 2007).

2.1.4.4 *Proxmox*

O *Proxmox* é uma solução de virtualização *open-source baseada* no GNU/Linux, distribuído sob licença GNU *Affero General Public License* (*AGPL*) v3 (CHENG, 2014).

A solução implementa duas tecnologias de virtualização, o *Kernel-based Virtual Machine* (*KVM*) e a virtualização baseada em containers *Linux Containers* (*LXC*) (Proxmox Server Solutions Gmbh, 2020).

O hipervisor suporta a virtualização total e a virtualização assistida por hardware fornecidos pelo *KVM* (CHENG, 2014).

A Proxmox Server Solutions Gmbh, mantenedora do projeto, oferece licenças empresariais de suporte que contam, também, com acesso ao repositório empresarial (Proxmox Server Solutions Gmbh, 2020).

2.2 Alta disponibilidade

A alta disponibilidade tem por objetivo manter os serviços disponíveis pelo maior tempo possível, para tal, são aplicadas um conjunto de técnicas, conceitos e ferramentas (PAVAN et al., 2014).

Para Caciato (2015), a alta disponibilidade fornece a capacidade de manter os serviços funcionando, mesmo com ocorrência de falhas. Tais falhas podem ser de software, hardware, energia e rede.

De acordo com Veras (2011), a alta disponibilidade pode acontecer a nível de hardware, sistema operacional e aplicação. No nível de hardware são utilizados componentes redundantes e tolerantes a falhas. No nível de sistema operacional pode ser aplicado um *cluster* de *failover*, que é um grupo de servidores independentes e com armazenamento compartilhado trabalhando juntos para aumento de disponibilidade. No nível de aplicação

o sistema operacional é utilizado como apoio para se ter alta disponibilidade em uma carga de trabalho específica.

Os sistemas computacionais, em nível de software ou hardware, possuem capacidade de mascarar falhas eventuais, tal capacidade lhes garante uma disponibilidade básica, oferecida sem nenhum mecanismo específico. Esta disponibilidade básica aliada a mecanismos específicos de detecção, mascaramento e recuperação de falhas proporcionam a alta disponibilidade (NASCIMENTO et al., 2010).

A implantação de um ambiente de alta disponibilidade pode variar de acordo com as necessidades das aplicações, recursos financeiros e modelo de negócios. Basicamente algumas configurações podem ser adotadas e combinadas para aumentar a disponibilidade, são elas: redundância, a fim de eliminar pontos únicos de falha; *failover*, que é a capacidade de o sistema identificar falhas e executar procedimentos determinados; *cluster*, conjunto de computadores interligados em rede que se comunicam por meio de seus sistemas, cada computador é chamado de nó; e balanceamento de carga, para distribuição uniforme das cargas de trabalho entre os nós de um *cluster* (CACIATO, 2015).

2.2.1 Virtualização e Alta Disponibilidade

Os softwares de virtualização trouxeram novas opções para a criação de *clusters*, tornando possível a alta disponibilidade em máquinas virtuais, utilizando o próprio software de virtualização (VERAS, 2011).

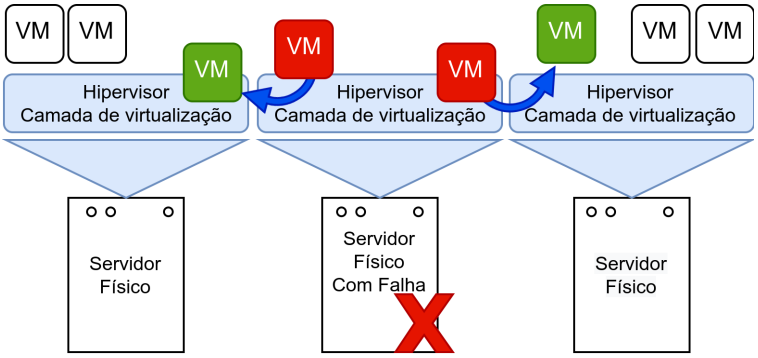
Veras (2011) classifica tais *clusters* em: *Clusters* HA, que promovem redundância com *failover* automático; *Clusters Load Balancing* (LB), que proporcionam melhor capacidade para execução da carga de trabalho; *Clusters High Performance Computing* (HPC), que visam o aumento da performance da aplicação; e o Grid, este tipo visa as qualidades de HA e HPC.

Qualquer *cluster* deve possuir múltiplos nós; interconexão, geralmente baseada em rede Ethernet; acesso ao volume de armazenamento; e o software de gerenciamento do *cluster* (VERAS, 2011).

Em um *cluster* de HA todos os servidores físicos são monitorados continuamente por um serviço chamado *heartbeat*, que possui um agente em cada nó do *cluster*. Na ausência de resposta ao *heartbeat*, de algum nó, as máquinas virtuais são imediatamente reiniciadas em outros nós com recursos computacionais disponíveis, como demonstrado na Figura 12 (CACIATO, 2015).

O armazenamento dos dados das máquinas virtuais é um ponto fundamental para a alta disponibilidade, para isso podem ser empregadas algumas técnicas como a replicação de discos entre os nós do *cluster* e o uso de um *storage* compartilhado.

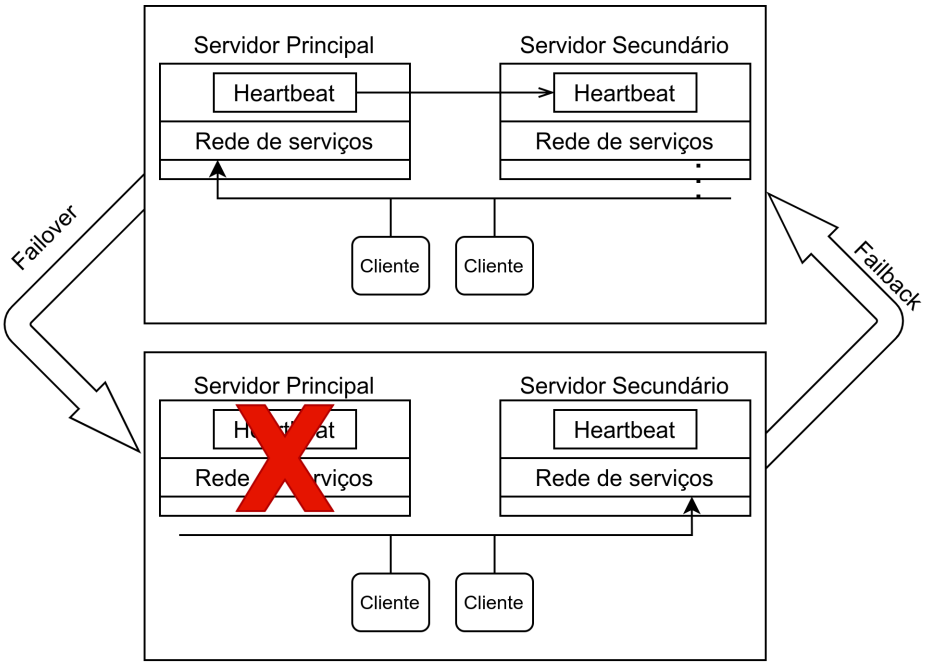
Figura 12 – Migração de *VM*s



Fonte: Adaptado de Caciato (2015, p. 20).

Na replicação de discos os dados armazenados nos nós são constantemente sincronizados, via rede, neste cenário geralmente são aplicados dois nós, o principal e o secundário. Em caso de falha do nó principal o secundário assume seu lugar e dá continuidade ao serviço, processo de *failover*. Quando o nó com falha é retornado os discos são novamente sincronizados, havendo sucesso, ele passa a ser novamente o principal, processo de *failback* (NASCIMENTO et al., 2010). Este ciclo pode ser visto na Figura 13.

Figura 13 – Processo de *failover* e *failback* de servidores



Fonte: Adaptado de Nascimento et al. (2010, p. 10).

Dessa forma é possível promover a alta disponibilidade em ambientes mais simples. Na Seção 2.3 será apresentada a técnica de armazenamento compartilhado, onde, basicamente, os nós de um *cluster* tem acesso a um único volume de armazenamento, onde são gravadas as informações das máquinas virtuais.

2.3 *Storage*

O *storage* é o responsável pelo armazenamento e confiabilidade dos dados de um sistema computacional, sendo um componente crítico da infraestrutura, dado o crescimento massivo de dados, a exigência de confiabilidade e, em ambientes virtualizados, a máxima disponibilidade possível (VERAS, 2011).

De acordo com Veras (2011) o *storage* é um dos componentes de um sistema de armazenamento, dito o principal, podendo utilizar meio magnético ou de estado sólido para persistência das informações. São, ainda, partes do sistema de armazenamento: os servidores, onde as aplicações e sistema operacional gerenciam o armazenamento de dados; e a conectividade, responsável pela interconexão de servidores e dispositivos de armazenamento, sendo composta por camadas físicas e lógicas.

Em um ambiente virtualizado o *storage* é fundamental para a técnica de migração ao vivo, onde, em caso de falha de um hipervisor, o processamento de suas máquinas virtuais é transferido para outro hipervisor sem a interrupção dos serviços (PAVAN et al., 2014).

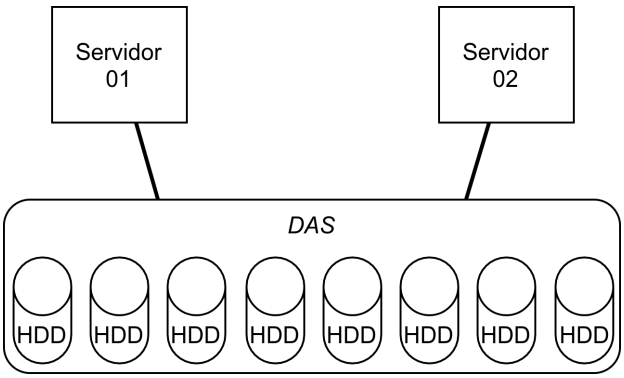
A seguir serão apresentadas algumas soluções de sistema de armazenamento.

2.3.0.1 *Direct Attached Storage* (DAS)

O DAS é um equipamento conectado diretamente aos servidores, para persistência de dados, sua conexão é feita por *Hot Bus Adapter*s (HBAs), que fornecem acesso a nível de bloco ao volume de armazenamento (TOSADORE, 2012), (COOPER, 2019). "O acesso por blocos é o mecanismo básico de acesso aos discos" (VERAS, 2011).

A proteção de dados em um DAS se dá pelo emprego de *Redundant Array of Independent Disks* ($RAID$), em que vários discos são configurados para trabalhar em conjunto, formando um único volume conhecido como *array* de discos (matriz de discos) (TOSADORE, 2012).

Figura 14 – Armazenamento DAS



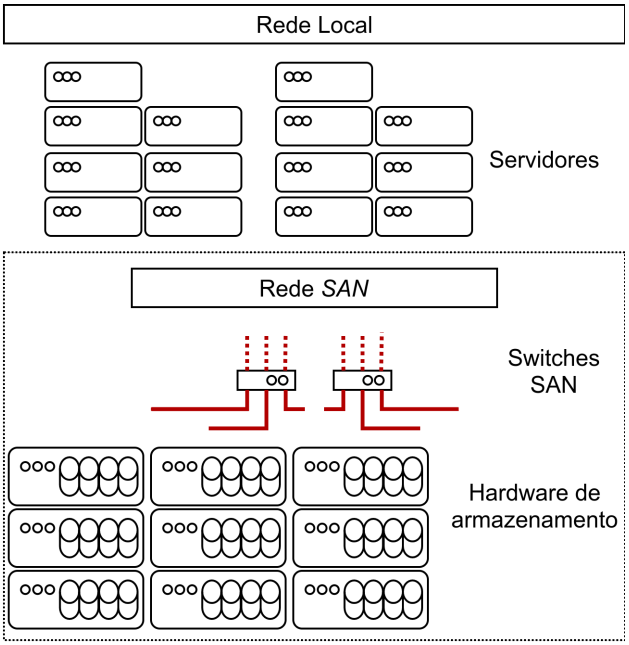
Fonte: Adaptado de Veras (2011, p. 141).

Por não ser um equipamento compartilhado na rede, o *DAS*, acaba tendo seu número de conexões limitada ao número de portas disponíveis, sendo ideal para pequenas e médias empresas, onde o número de servidores conectados geralmente é pequeno (TOSADORE, 2012). A Figura 14 representa um ambiente *DAS*.

2.3.0.2 *Storage Area Network* (*SAN*)

Para Veras (2011), a *SAN* é considerada uma evolução do *DAS*, eliminando a necessidade de conexão direta do servidor com o *storage*, com uma rede de *storage* independentes que utilizam protocolos específicos. A Figura 15 mostra sua organização.

Figura 15 – Topologia *SAN*



Fonte: Adaptado de Chalkias (2020).

O padrão *Gigabit Ethernet* e *Fibre Channel* (*FC*) podem ser empregados em uma infraestrutura *SAN*, criando uma rede de armazenamento dedicada e escalável onde vários servidores se conectam a vários dispositivos de *storage* em nível de bloco (VERAS, 2011).

O planejamento de uma *SAN* requer redundância de equipamentos e conexões, configuração de *RAID* e soluções de *backup* para garantir a disponibilidade do ambiente (TOSADORE, 2012).

Veras (2011) comenta sobre considerar a homologação dos produtos envolvidos em uma rede *SAN* e a solução de virtualização, uma vez que parte das funcionalidades obtidas pela virtualização dependem do sistema de armazenamento.

O custo de uma solução baseada em rede *SAN* é considerável e leva ao *vendor lock-in*, onde apenas os equipamentos do fornecedor escolhido são compatíveis, limitando as opções no momento de escalar a rede.Também deve ser considerado o fim do suporte

por parte do fornecedor, que pode acarretar em uma migração com custos elevados (CHALKIAS, 2020).

2.3.0.3 *Network Attached Storage* (NAS)

O NAS é um equipamento independente, conectado a uma rede *Ethernet*, que possui discos configurados em um sistema de $RAID$ e fornece espaço para armazenamento de dados (SHAIKH et al., 2019). A Figura 16 representa este tipo de solução.

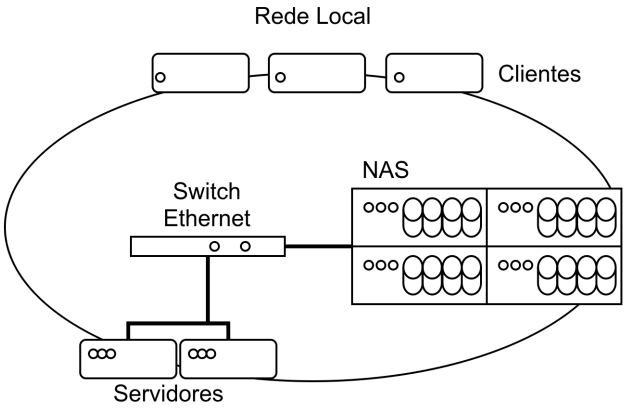
Vários clientes podem ser atendidos por um NAS, que diferentemente de um servidor de arquivos convencional possuí um sistema operacional próprio e otimizado para fluxos de entrada/saída, suportando vários tipos de sistemas operacionais dos clientes (VERAS, 2011, p. 148).

Um sistema NAS utiliza, basicamente, protocolos de rede e de compartilhamento de arquivos como *Transmission Control Protocol* (TCP)/*Internet Protocol* (IP), *Common Internet File System* ($CIFS$), *Server Message Block* (SMB) e *Network File System* (NFS) para prover seus serviços (VERAS, 2011), (SHAIKH et al., 2019).

Dispositivos NAS são suportados pelos softwares de virtualização, permitindo otimizar sua utilização (VERAS, 2011).

Soluções NAS mais sofisticadas podem contar com diversas unidades interligadas, promovendo a redundância do sistema como um todo, o que garante maior disponibilidade, mesmo em coso de falha de um equipamento (CONTROLENET, 2017).

Figura 16 – Topologia NAS



Fonte: Adaptado de Shaikh et al. (2019, p. 1415).

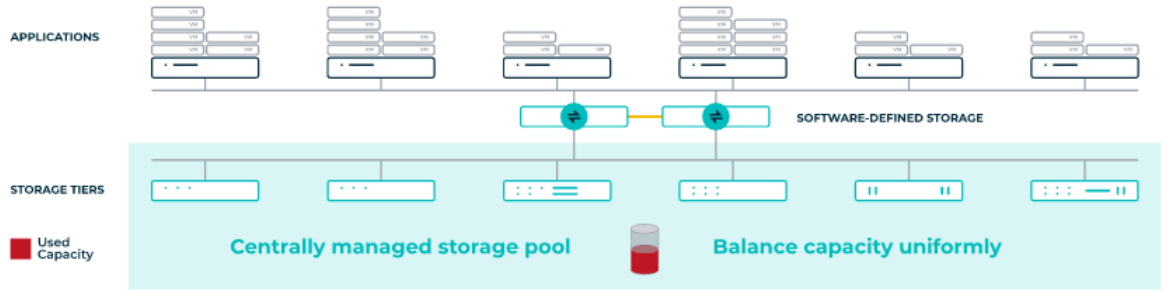
2.3.0.4 *Software Defined Storage* (SDS)

O SDS é uma solução que abstrai a camada de hardware do sistema, fornecendo um ambiente mais dinâmico e flexível (VMWARE, 2021). Este tipo de armazenamento está mudando o paradigma nos *data centers* pela flexibilidade e redução dos custos operacionais (CHALKIAS, 2020).

Em um *SDS* existem duas camadas: a controle, que gerencia os recursos de armazenamento e controla o fluxo de informações; e a de dados, onde estão dispositivos de armazenamento de dados. A camada de controle fica entre as aplicações e a infraestrutura (MORAIS, 2019).

Diversos dispositivos de armazenamento podem ser combinados em *pools* com gerenciamento centralizado (DATACORE, 2021), como mostra a Figura 17. Os *pools* podem ser criados com base no dispositivo de armazenamento empregado. Por exemplo, para uma aplicação de banco de dados, um *pool* de unidades de armazenamento em estado sólido oferece melhor desempenho que um composto por unidades de armazenamento mecânicas.

Figura 17 – Solução *SDS*



Fonte: Datacore (2021).

A flexibilidade do *SDS* é tamanha que é possível utilizar soluções de armazenamento já existentes, como *SAN*, *NAS* ou *DAS* (VMWARE, 2021), além de servidores comuns (MORAIS, 2019).

Para Sant'Ana (2018) o *SDS* abre caminho para a Hiperconvergência, em que os servidores de armazenamento podem executar tarefas de computação, como máquinas virtuais, além das funções de armazenamento, uma vez que o software controla toda a estrutura.

2.4 Redes

Um conjunto de equipamentos de informática e software conectados formam uma rede, que utiliza dispositivos físicos para emitirem e receberem impulsos elétricos, ondas eletromagnéticas ou qualquer outro meio a fim de transportar dados (BORQUE, 2013).

Conforme apontado por Veras (2011), a interconexão é fundamental para um *cluster* de alta disponibilidade e para tal, comumente, são aplicadas redes *Ethernet*.

O padrão *Ethernet* tornou-se dominante no âmbito de redes locais cabeadas, com larguras de banda que variam de 10 *Megabits per second* (*Mbps*), 100 *Mbps*, 1 *Gigabit per second* (*Gbps*) e 10 *Gbps* (COULOURIS et al., 2013).

As redes *Local Area Network*s (*LAN*s) utilizam meios físicos, como o cabo de par trançado, o cabo coaxial ou a fibra óptica, para transporte de mensagens entre computadores (COULOURIS et al., 2013).

Estas mensagens trafegam em formato de pacotes, que são formados por um campo de cabeçalho e um campo de dados, este último com tamanho variável e limitado pelo *Maximum Transfer Unit* (*MTU*). Quando uma mensagem tem tamanho maior que o *MTU* suportado pela camada de rede esta é fragmentada em tamanho adequado e sua sequência é identificada com números no cabeçalho do pacote, para a correta montagem no destino (COULOURIS et al., 2013).

Em uma rede a comunicação é efetiva pela existência de protocolos, que definem as sequências de mensagens que devem ser trocadas e o formato de dados das mesmas, os protocolos são implementados por módulos de software na origem e destino. A necessidade de padrões que atendessem aos requisitos de sistemas abertos levou a *International Organization for Standardization* (*ISO*) a adotar o modelo de referência *Open System Interconnection* (*OSI*), que é um conjunto de protocolos estruturado em camadas (COULOURIS et al., 2013).

Dentro de uma rede cada elemento possui um endereço *IP* e um endereço *Media Access Control* (*MAC*), que o identifica na camada de rede e enlace, do modelo OSI, respectivamente. A quantidade de elementos permitidos em uma rede é definida pela máscara de *subrede* (BORQUE, 2013).

Os elementos que integram uma rede são: os *hosts*, elementos finais da rede que fornecem ou consomem serviços; hubs, utilizados para interconexão de equipamentos onde todo sinal recebido por uma porta é repassado para todas as outras, sem nenhuma inteligência aplicada; *switch*, que é um dispositivo digital lógico que interconecta dois ou mais segmentos de rede e trabalha na camada de enlace de dados do modelo *OSI*, dirigindo o tráfego até o endereço *MAC* de destino; e o roteador, que permite a interconexão de redes *IP*, proporcionando conectividade ao nível de rede do modelo *OSI* (BORQUE, 2013).

No âmbito de *clusters* de virtualização é comum a utilização de *Virtual Local Area Network*s (*VLans*) e *Ethernet Bonding*.

Com as *VLans* é possível separar o tráfego de dados criando redes locais virtuais, delimitando contextos dos dispositivos, tal técnica permite solucionar diversos problemas do *datacenter* (SANTOS, 2019).

Com o *Ethernet Bonding* é possivel vincular dois ou mais canais *Ethernet* em um único canal, permitindo o balanceamento de carga e alta disponibilidade deste canal. A norma *Institute of Electrical and Electronic Engineers* (*IEEE*) 802.3ad intitulada *Link Aggregation* (*LAG*) define o *Link Aggregation Control Protocol* (*LACP*) como protocolo

padrão de conexão entre dispositivos com links agregados (GUEDES, 2018).

No mercado de protocolos existem soluções proprietárias, geralmente atreladas aos fabricantes e seus respectivos equipamentos e outras soluções de software, inclusive livres, que atendem à finalidade de agregação de link (ROCHA JUNIOR, 2017).

A biblioteca nativa e livre de agregação de links do Linux, *Linux Bonding* é um exemplo de solução baseada em software. Esta implementa o protoclo $LACP$; operando nas camadas de enlace, de rede e de transporte do modelo OSI. Em redes onde os equipamentos não fornecem ao protocolo $LACP$ é possível utilizar algum modo de agregação baseado apenas em software, estes atuam na camada de elance do modelo OSI (ROCHA JUNIOR, 2017). Dentre os modos de operação possíveis existem duas opções: a $LACP$, que segue a norma $IEEE$ 802.3ad; e o *active-bakcup*, que utiliza apenas um link do conjunto como ativo e os demais para redundância.

2.5 *Backup*

O *backup* é uma técnica para realização de cópias de segurança de dados específicos com a finalidade de recuperação em caso de problemas ou necessidades específicas (GODOY, 2011). No entanto a realização de *backup* não é simples, deve haver um processo que garanta a eficácia do mesmo (RODRIGUES, 2017).

A forma e a frequência de realização de backups devem observar as necessidades do ambiente. Desse modo é possível estabelecer um processo de *backup* e uma estratégia de recuperação de dados com testes que garantam a eficácia do processo, antevendo uma real necessidade, uma vez que não é incomum a ocorrência de falhas em tentativas de recuperação de dados (VERAS, 2011).

O *backup* pode ser realizado de forma completa, incremental e diferencial. Na forma completa, sempre que um *backup* é realizado todos os arquivos são copiados. Nas formas incremental e diferencial somente o que foi modificado após a execução do último *backup* é copiado e ambas iniciam de um *backup* completo. A diferença entre as duas últimas é que a incremental tem por resultado somente a cópia dos arquivos modificados desde o último *backup*, enquanto a diferencial é acumulativa tendo como resultado a cópia de todos os arquivos modificados desde o último *backup* completo. Para a restauração do *backup* incremental é preciso restaurar o *backup* completo e todos os fragmentos incrementais até o ponto desejado, na restauração do *backup* diferencial é necessário somente o *backup* completo e o último *backup* diferencial (GODOY, 2011).

As formas podem ser combinadas dentro de uma programação de *backup* a fim de se ter o melhor desempenho, segurança e capacidade de armazenamento (GODOY, 2011).

Com o processo de *backup* uma grande quantidade de dados é produzida, para

poupar espaço de armazenamento podem ser aplicadas técnicas como a deduplicação e compressão de dados. A deduplicação visa eliminar dados redundantes do volume de armazenamento, para isso uma única cópia do dado repetido é salva e as demais são substituídas por apontamentos a esta cópia (VERAS, 2011). A compressão utiliza algoritmos para a codificação de dados a fim de ocupar menos espaço de armazenamento, a compressão aumenta o tempo de processamento do *backup* (RODRIGUES, 2017).

O aspecto de segurança dos *backups* é relevante e técnicas de encriptação podem ser empregadas, no entanto o desempenho e tempo de *backup* e restauração podem ser afetados (VERAS, 2011).

O *backup* é uma tarefa de um processo amplo que visa garantir a disponibilidade do *datacenter* e da infraestrutura. A combinação do *cluster* de HA, a replicação de dados e o *backup* forma o plano de recuperação de desastres que por sua vez visa a continuidade do negócio (VERAS, 2011).

3 Estudo de caso

Dadas as necessidades apontadas na motivação do trabalho e requisitos levantados no Capítulo 2, foi realizada uma busca por soluções que contemplassem a virtualização, alta disponibilidade e armazenamento.

O presente trabalho baseou-se em pesquisa científica acerca dos temas: virtualização, alta disponibilidade e soluções de armazenamento. Foram consultados artigos, manuais e livros para realização da mesma.

De acordo com a pesquisa realizada, um sistema de virtualização em alta disponibilidade deve formar um *cluster*, com múltiplos servidores interconectados via rede *Ethernet*, um volume para armazenamento de dados ou *storage* e o software hipervisor com gerenciamento do *cluster*.

Para a tecnologia inferida na pesquisa ser implantada no ambiente do Cartório foram disponibilizados os seguintes equipamentos:

- 1 servidor Dell® R620 - 2 processadores Intel® Xeon® CPU E5-2658, 2 *Power Supply Unit*s (*PSU*s) 750 *watts*, 96 *Gigabyte* (*GB*) de *RAM*, 8 unidades *Hard Disk Drive* (*HDD*) de 600 *GB* com taxa de transferência de 6 *Gbps* e velocidade de 10.000 Rotação por minuto (RPM);
- 2 servidores Dell® R620 - 2 processadores Intel® Xeon® CPU E5-2650, 2 *PSU*s 750 *watts*, 96 *GB* de *RAM*, 8 unidades *HDD* de 600 *GB* com taxa de transferência de 6 *Gbps* e velocidade de 10.000 RPM;
- *Switch* Dell® *Networking* X1052;
- *Switch* Dell® *Networking* N2048;
- 3 adaptadores de rede convergido Intel® *Ethernet* X520-DA1;
- 3 cabos *Small Form-factor Pluggable* (*SFP*)+ TWINAX;
- 2 *Uninterruptible Power Supply* (*UPS*) Intelbras® DNB 1.500 Voltampere;

Foi realizada a busca pela ferramenta de virtualização mais adequada ao cenário em questão. A preferência inicial era por ferramentas com uma forma de licenciamento simples ou até mesmo *open-source*, sendo avaliado o *VMware vSphere*, o *XenServer* e o *Proxmox*.

A solução da *VMware* é bem completa e robusta, porém a sua versão mais acessível economicamente não contemplava o requisito de *HA* e os custos de licenciamento são

consideravelmente elevados para obter o conjunto de serviços do pacote *VMware vSphere Essentials Plus Kit*.

O *XenServer*, por sua vez, passou apresentar limitações em recursos como *HA* e migração de dados em tempo real a partir de sua versão 7.3 *Free Edition* (SILVA, 2019), sendo assim a ferramenta não despertou maiores interesses.

O *Proxmox* oferece os recursos de *HA*, migração de máquinas virtuais ao vivo além de ser *open-source* com opção de assinatura de plano de suporte (CHENG, 2014), atendendo aos requisitos especificados.

Nos estudos da documentação do *Proxmox* foi encontrado o *Ceph*, um *storage SDS*, que pode ser instalado em um servidor executando o *Proxmox*, formando assim uma solução hiperconvergente, onde os servidores são encarregados do processamento das máquinas virtuais e armazenamento dos dados do *storage*. O *Ceph* trabalha com a replicação de dados entre os discos dos servidores, o que garante a *HA* e elimina o ponto único de falha da solução de armazenamento. (Proxmox Server Solutions Gmbh, 2020).

Verificou-se a exigência de três servidores para a formação do *cluster* de *HA* e do *cluster* de armazenamento de dados do *Ceph* (TOLFO, 2019) e esta foi atendida pelo hardware disponível.

Foram avaliadas soluções de armazenamento baseadas em *DAS* e *NAS*, porém o custo de aquisição foi considerado elevado para atender os mesmos pontos já contemplados pelo *Ceph* e os dispositivos de armazenamento dos servidores seriam subutilizados, uma vez que soluções baseadas em *DAS* ou *NAS* utilizam os servidores hipervisores apenas para processamento das máquinas virtuais.

A solução *Proxmox* e *Ceph* foi escolhida para implantação. Dado o cenário em questão, com três servidores de configurações equivalentes, espaço para armazenamento disponível e interconexão por meio de rede 10 *Gigabit Ethernet* (*GbE*) e 1 *GbE*.

3.1 Soluções utilizadas

3.1.1 *Proxmox*

O *Proxmox* é uma solução de virtualização nativa do Tipo I (Seção 2.1.3.1) que utiliza o *kernel* Linux, é baseado na distribuição GNU/Linux Debian e está sob licença GNU *AGPL* v3, seu código é aberto para inspeção e contribuição (Proxmox Server Solutions Gmbh, 2020), (SILVA, 2019). É capaz de virtualizar sistemas Windows e sistemas com base Unix (CHENG, 2014).

Oferece recursos presentes nas principais ferramentas do mercado, entre elas: migração ao vivo que permite mover uma *VM* em execução de um servidor físico para

outro sem tempo de inatividade; *HA* que garante a interrupção mínima dos serviços, migrando as *VM*s, em caso de falha de um servidor físico, para um servidor disponível; gerenciamento de redes que permite configurações de redes privadas entre máquinas virtuais, inclusive com configurações de *VLan*s; *firewall* integrado que permite o gerenciamento individual de *VM*s, do próprio hipervisor e de grupos de segurança; suporte a várias tecnologias de armazenamento disponíveis no mercado; ferramenta de *backup* que permite criação de planos de *backup* e configuração de retenção de *backups* no *storage*; gerenciamento por linha de comando (*Command Line Interface* (*CLI*)); e gerenciamento centralizado por meio de interface *web* (CHENG, 2014), (SILVA, 2019).

O suporte a virtualização é provido pelo *KVM*, que basicamente é um módulo do *kernel* Linux responsável por oferecer a virtualização assistida por hardware (Seção 2.1.3.5). O *KVM* expõe uma interface para o *Qemu*, responsável por emular o hardware da *VM* (CHENG, 2014). Os dispositivos emulados pelo *Qemu* são apresentados ao SO da *VM*, que utiliza os *drivers* apropriados e os controla como se fossem reais. Contudo os dispositivos emulados requerem trabalho extra da *CPU* do hipervisor, pois as ações que seriam executadas em hardware estão sendo simuladas por software. Para contornar o problema o *Qemu* utiliza a *libvirt* para entregar dispositivos para-virtualizados, estes dispositivos possuem *drivers* próprios que devem ser instalados no SO convidado, com isso o desempenho do hipervisor é acelerado (CHENG, 2014), (Proxmox Server Solutions Gmbh, 2020).

Há possibilidade de utilizar *containers*, que são uma alternativa leve às *VM*s, tendo como base a tecnologia *LXC*. Os *containers* são uma virtualização baseada em sistema operacional, onde o *kernel* do hipervisor é compartilhado e dessa forma podem acessar os recursos do sistema hipervisor diretamente, com tempos de execução muito baixos (CHENG, 2014), (Proxmox Server Solutions Gmbh, 2020). Algumas restrições se aplicam, são elas: apenas distribuições Linux podem ser executadas em *containers*; o acesso ao hipervisor é restrito, por questões de segurança, cada *container* é executado em um espaço próprio e algumas *syscalls* não são permitidas. A criação de *containers* é feita por meio de *templates* e o *Proxmox* oferece integração com o repositório *Turnkey*. Os recursos disponíveis às *VM*s se aplicam da mesma forma aos *containers* (Proxmox Server Solutions Gmbh, 2020).

3.1.2 *Ceph*

O *Ceph* é uma solução de armazenamento *SDS*, de código aberto, mantido por grandes empresas de tecnologia. Seus princípios funcionais são a escalabilidade, inexistência de ponto único de falha, base em software e execução em dispositivos comuns de computação (TOLFO, 2019).

A tolerância a falhas deste sistema o torna confiável, pois assume que falhas nos

componentes de armazenamento são constantes e estas devem ser contornadas (SANTOS, 2019).

Esta ferramenta disponibiliza recursos amplamente utilizados em ambientes corporativos: a escala sob demanda do *cluster*; armazenamento em cache; políticas de acesso aos discos; e replicação em diferentes regiões geográficas. Por padrão, os dados armazenados no *Ceph* são redimensionáveis e distribuídos, garantindo segurança e disponibilidade (CHALKIAS, 2020).

Os benefícios de sua implantação incluem simplificar o gerenciamento de dados, reduzir os custos de hardware e garante a independência de fornecedores sem comprometer o desempenho, confiabilidade e escalabilidade (AHMED, 2014).

O *Ceph* pode armazenar dados em formato de arquivos, blocos e objetos (SANTOS, 2019). No armazenamento de arquivos é utilizado o *Ceph File System* ($CephFS$) que possuí compatibilidade com o padrão *Portable Operating System Interface* ($POSIX$), sendo compatível com aplicações legadas ou modernas. O armazenamento em blocos, geralmente utilizado para máquinas virtuais, oferece alto desempenho. O armazenamento de objetos possui compatibilidade com Amazon S3 e OpenStack Swift, amplamente utilizado por aplicações modernas (SCOTT, 2015).

A continuidade desta seção é destinada à descrição técnica da composição do *Ceph* e sua integração com o *Proxmox*.

O *Object Storage Daemon* (OSD) é o serviço responsável pelo armazenamento e replicação dos dados, interagindo diretamente com o dispositivo de armazenamento, a recomendação é utilizar um OSD por disco rígido. É o componente de mais baixo nível do sistema (SANTOS, 2019) e este fornece informações do seu estado para os *monitors* e *managers* (AHMED, 2014).

O *monitor* mantém informações do *cluster* e dos mapas de outros *monitors*, OSDs, *managers* e o mapa *Controlled Replication Under Scalable Hashing* ($CRUSH$) (SANTOS, 2019). O *monitor* autentica os clientes e serviços que desejam acessar ou gravar dados e no processo de autenticação fornece um mapa do *cluster* e autorização para conexão direta aos OSDs (SANTOS, 2019). O algoritmo $CRUSH$ reside neste componente (TOLFO, 2019). Para assegurar a integridade do *cluster Ceph* são necessários no mínimo três *monitors* (AHMED, 2014).

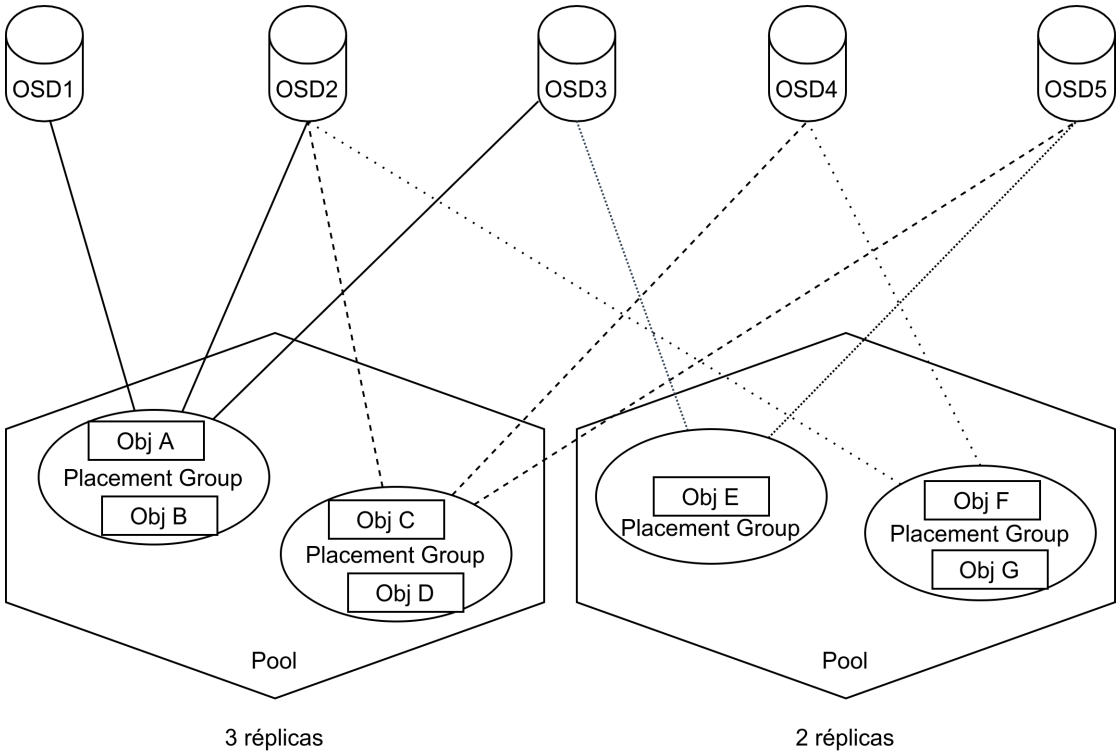
O *manager* é responsável por coletar métricas de execução e estado do *cluster Ceph* e fornecer interface externa para sistemas de monitoramento (SANTOS, 2019).

O *Metadata Servers* (MDS) é utilizado pelo $CephFS$ para armazenar metadados do sistema de arquivos, o armazenamento em bloco e objetos não utilizam o MDS (SANTOS, 2019). Quando utilizado, o MDS deve ser instalado em no mínimo dois servidores, para garantir a disponibilidade do serviço (AHMED, 2014).

O *Placement Group* (*PG*) combina vários objetos em um grupo, o objeto é a menor unidade no armazenamento do *Ceph*, cada *PG* é replicado para vários *OSD*s (AHMED, 2014), (CHENG, 2014).

O *pool* é visto como uma partição lógica onde o *Ceph* armazena os dados (CHENG, 2014). Em cada *pool* é definido o número de réplicas e o tamanho do *PG* (CHENG, 2014). A Figura 18 contextualiza seu funcionamento, com dois *pools* configurados com diferentes números de réplicas.

Figura 18 – *Pool* com diferentes configurações de replicação por *PG*.



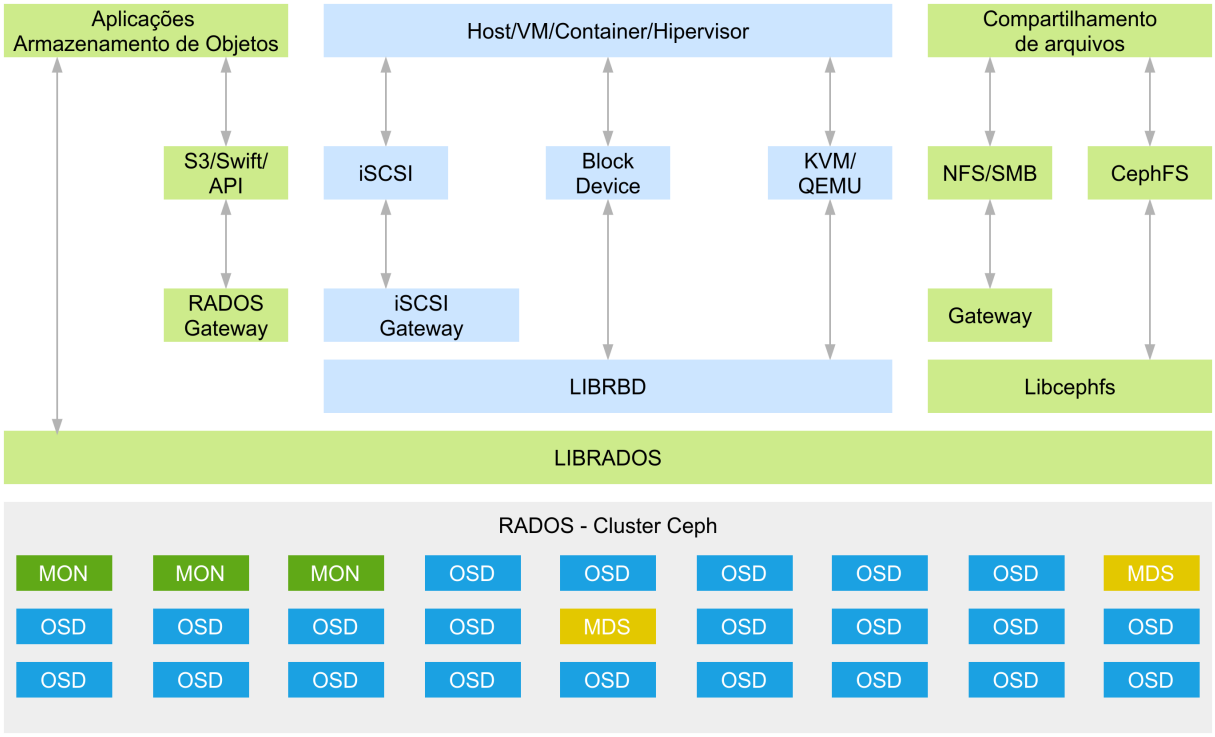
Fonte: Adaptado de Cheng (2014, p. 56)

O algoritmo *CRUSH* garante a eficiência, escalabilidade e confiabilidade do *Ceph* (SANTOS, 2019). Este algoritmo permite o cálculo de localização dos dados dentro do *Cluster Ceph* de forma descentralizada, assim o cliente pode obter a informação pelo caminho mais curto; e também é responsável por coordenar e monitorar os *OSD*s, indicando onde os dados devem ser gravados (TOLFO, 2019). O *CRUSH* permite que o *cluster Ceph* seja escalonado, reequilibrado e recuperado dinamicamente (CEPH, 2021).

A combinação de *OSD*s e *monitors* forma o *Reliable Autonomous Distributed Object Store* (*RADOS*) que é o *cluster* de armazenamento de dados *Ceph* (SANTOS, 2019). O acesso ao *RADOS* é realizado pela *Library to access RADOS* (*LIBRADOS*), que provê por padrão: a *Rados Gateway* (*RGW*), para acesso a nível de objeto; a *Rados Block Device* (*RBD*), para acesso a nível de blocos; e a *CephFS* para acesso a nível de arquivos no padrão *POSIX*, esta última ainda conta com o *MDS* para funcionamento. Além das

interfaces padrão a *LIBRADOS* abre a possibilidade de implementação de soluções de acesso personalizadas (SCOTT, 2015). A Figura 19 ilustra a solução *Ceph*.

Figura 19 – A solução *Ceph*



Fonte: Adaptado do site da Ambedded Technology Co., LTD[1].

O *Ceph* possui integração com o *Proxmox*, formando uma solução hiperconvergente. Para a instalação do *Proxmox* e *Ceph* são necessários três servidores, de preferência idênticos. O *Ceph* foi projetado para trabalhar diretamente com os discos dos servidores, sendo assim, não é recomendável o uso de configurações *RAID*, pois pode reduzir a performance e afetar a disponibilidade dos dados (Proxmox Server Solutions Gmbh, 2020).

Os requisitos de hardware para o *Ceph* podem ser divididos de acordo com a função no *cluster*, conforme a página oficial do projeto <https://docs.ceph.com/en/latest/start/hardware-recommendations/>. Para cada *OSD* é recomendado de 2 a 4 *GB* de memória *RAM* e 1 núcleo de processador. Para *monitor* é recomendado 24 *GB* de memória e 2 núcleos de processamento. Para *MDS* é recomendado 2 *GB* de memória e 2 núcleos de processamento.

É recomendada a utilização de rede 10 *GbE*, para *clusters* menores é possível utilizar redes de 1 *GbE* ou a agregação de links. Para os discos é recomendada quantidade e distribuição uniforme entre os servidores. Por exemplo, quatro discos de 500 *GB* em cada servidor é melhor que três discos de 250 *GB* e um disco de 1 *TB* (Proxmox Server Solutions Gmbh, 2020).

[1] Disponível em: <https://www.ambedded.com.tw/en/technology/Ceph-Block-Storage/tech_block-storage.html>

3.2 Instalação do ambiente

A instalação do ambiente deu-se em dois momentos, a instalação do hardware e a instalação do software.

3.2.1 Instalação do hardware

O primeiro passo para a implantação foi a instalação e conexão do hardware. Os servidores e *switches* foram instalados em um *rack* próprio.

Cada servidor teve sua placa Intel X520 conectada a uma porta $SFP+$ do *switch* Dell x1052, por meio de cabos de conexão direta TWINAX. Estas conexões trafegam dados a uma velocidade de 10 *Gbps* e são destinadas ao *cluster Proxmox* e ao *Ceph*.

Cada servidor teve duas portas de conexão *Ethernet* 1 *GbE* destinadas ao *backup* da conexão principal do *cluster*, conectadas ao *switch* Dell N2048, utilizando agregação de links $LACP$ (802.3ad).

Os *switches* N2048 e x1052 foram interconectados utilizando duas portas 1 *GbE*, com $LACP$. Foi verificada a ativação do *Spanning Tree Protocol* (STP) nos *switches*, a fim de evitar *loops* na rede. Uma $VLan$ foi criada nos *switches*, esta separa o tráfego da rede do *cluster* do da rede local, contendo as portas mencionadas anteriormente.

Por fim, as duas portas de conexão *Ethernet* 1 *GbE* restantes em cada servidor foram conectadas, uma em cada *switch*, e utilizadas na rede LAN das VM e gerenciamento dos servidores.

A alimentação do conjunto ficou a cargo de duas unidades UPS Intelbras DNB 1.5Kva TW, com capacidade para 1.350 *watts* de carga. Cada servidor dispõe de uma fonte principal, que consome em média 160 *watts* de potência com pico máximo registrado de 380 *watts*, e uma fonte secundária em *stand-by*. A primeira unidade UPS recebeu a conexão da fonte principal de dois servidores e a fonte secundária do terceiro, tendo um consumo médio de 320 *watts*. A segunda unidade UPS recebeu a conexão de duas fontes secundárias e uma fonte principal, tendo consumo médio de 160 *watts*.

Tal conexão foi realizada para distribuição de carga e garantia da redundância no fornecimento de energia para os servidores. Considerando um cenário de falha de uma unidade UPS a outra manteria a alimentação de todo o conjunto, mesmo em pico máximo, onde o consumo é aproximadamente 1.140 *watts*. O pico de consumo ocorre apenas em momentos de inicialização dos servidores, durante o uso a média de consumo é de 480 *watts* para todo o conjunto. A instalação elétrica conta com gerador de energia que entra em funcionamento após 15 segundos de eventual falha na rede de fornecimento.

3.2.2 Instalação do software

Antes de iniciar a instalação foram necessárias configurações na controladora $RAID$ dos servidores. Como visto anteriormente, o *Ceph* precisa ter acesso aos discos de forma individualizada, sem a abstração de um volume $RAID$. Desta forma, a configuração de $RAID$ foi desabilitada, nas controladoras PERC H710p.

Foi verificado o suporte a virtualização assistida por hardware no *setup* de cada servidor. Sem este recurso ativado há penalidades de desempenho, mencionadas nesta pesquisa.

O processo de instalação foi simples e é descrito no manual do *Proxmox* <https://pve.proxmox.com/pve-docs/pve-admin-guide.pdf>. Durante a instalação foi solicitada a escolha do disco rígido destinado ao sistema, localidade e horário, definição de senha do usuário *root* e configuração do endereço de rede para gerenciamento.

A instalação foi feita no HDD 0 de cada servidor, utilizando-se o sistema de arquivos Ext4; apenas um disco foi utilizado sem configuração de $RAID$, a fim de não onerar a capacidade de armazenamento do *storage*, formado pelos discos restantes. A não utilização de um arranjo $RAID$ para a instalação do sistema é um risco assumido, considerando a quantidade de servidores no *cluster* e o processo de reinstalação do hipervisor, abordado na Seção 3.5.

3.3 Configuração do ambiente

Todo gerenciamento do *Proxmox* foi feito pela interface *web*, acessando o endereço configurado na instalação de cada servidor, na porta padrão 8006. Após a instalação foram configuradas as interfaces de rede de todos os servidores, foi feita a criação do *cluster*, instalação e configuração do *Ceph* e criação de máquinas virtuais de teste.

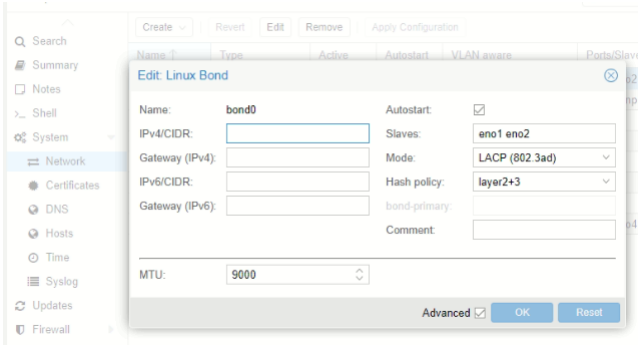
3.3.1 Configuração de rede e criação do *cluster*

Em cada servidor foi criada uma *Linux Bond* em modo $LACP$ (802.3ad) com duas portas 1 GbE, denominada *bond0*. Outra *Bond* foi criada, *bond1*, em modo *active-backup*, composta pela interface 10 GbE e a *bond0*, sendo a interface 10 GbE definida como primária. O tamanho do MTU das interfaces e *bonds* foi ajustado para 9.000 bytes, tendo em vista que estes componentes formam a rede do *cluster* com maior largura de banda, sendo possível enviar menor quantidade de pacotes com tamanhos maiores. Na *bond1* foi configurado manualmente o endereço IP em uma faixa escolhida para ser a rede do *cluster*.

As Figuras 20 e 21 mostram a criação da *bond0* e da *bond1*, por meio da opção *network* no menu do servidor *Proxmox*; o endereço de rede, na Figura 21, foi encoberto

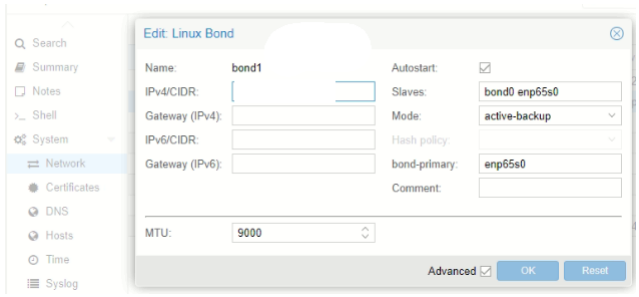
para preservar a integridade do ambiente, cada servidor tem um endereço próprio nesta configuração.

Figura 20 – Criação da *bond0*



Fonte: Elaborado pelo autor.

Figura 21 – Criação da *bond1*



Fonte: Elaborado pelo autor.

As interfaces que formam a *bond0* foram conectadas às portas do *switch* N2048, a interface 10 *GbE* foi conectada a porta *SFP+* do *switch* X1052, estas portas estão configuradas para a *VLan* do *cluster*.

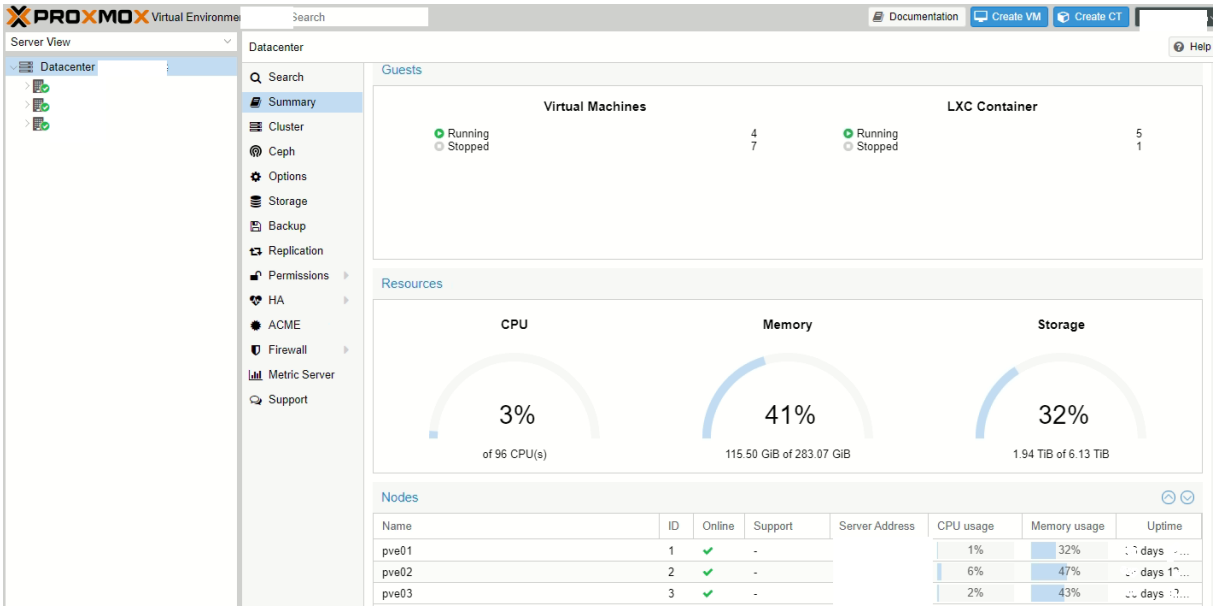
Em seguida foi criado o *cluster Proxmox*, acessando o item *cluster* do menu *datacenter* do primeiro servidor, executando a ação *create cluster*. Em sua configuração o *link0* recebeu o endereço de rede definido no passo anterior, o *link1* não foi utilizado pois a *bond1* garante a disponibilidade do *link0*. Os demais servidores foram adicionados ao *cluster*, por meio do mesmo menu, utilizando a ação *join cluster* e informando as credenciais de ingresso obtidas na criação do *cluster*.

A Figura 22 mostra o resumo do *cluster* criado, as informações sensíveis do ambiente foram preservadas.

3.3.2 Instalação e configuração do *Ceph*

Para utilização do *Ceph* foi necessária a instalação e configuração inicial, criação dos *monitors* e *managers*, criação dos *OSD* e por fim a criação do *pool* com adição ao *storage* do *Proxmox*.

Figura 22 – Resumo do *Cluster Proxmox*



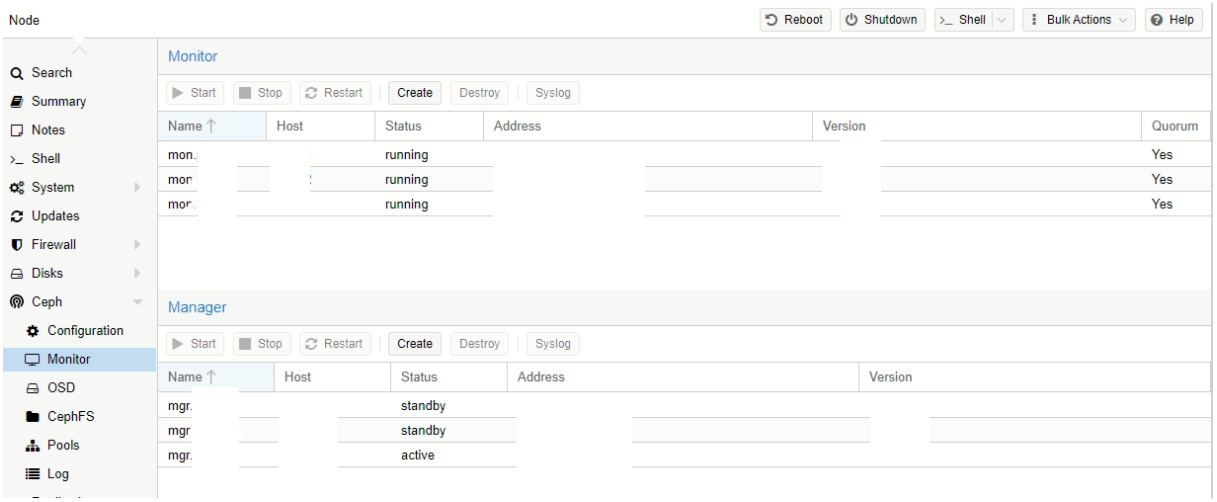
Fonte: Elaborado pelo autor.

O *Ceph* não vem instalado por padrão no *Proxmox*, a instalação foi feita em cada servidor do *cluster* através item *Ceph* do menu. Apenas na instalação do primeiro servidor foi solicitada a informação da rede utilizada para o *Ceph*, nos demais a configuração é sincronizada pelo *Proxmox*.

A mesma rede do *cluster* do *Proxmox* foi utilizada para a rede pública e rede de *cluster* do *Ceph*.

Com a instalação concluída, em todos os servidores, foi feita a criação dos *monitors* e *managers*, um para cada servidor, o resultado é exibido na Figura 23.

Figura 23 – *Monitors* e *managers* criados

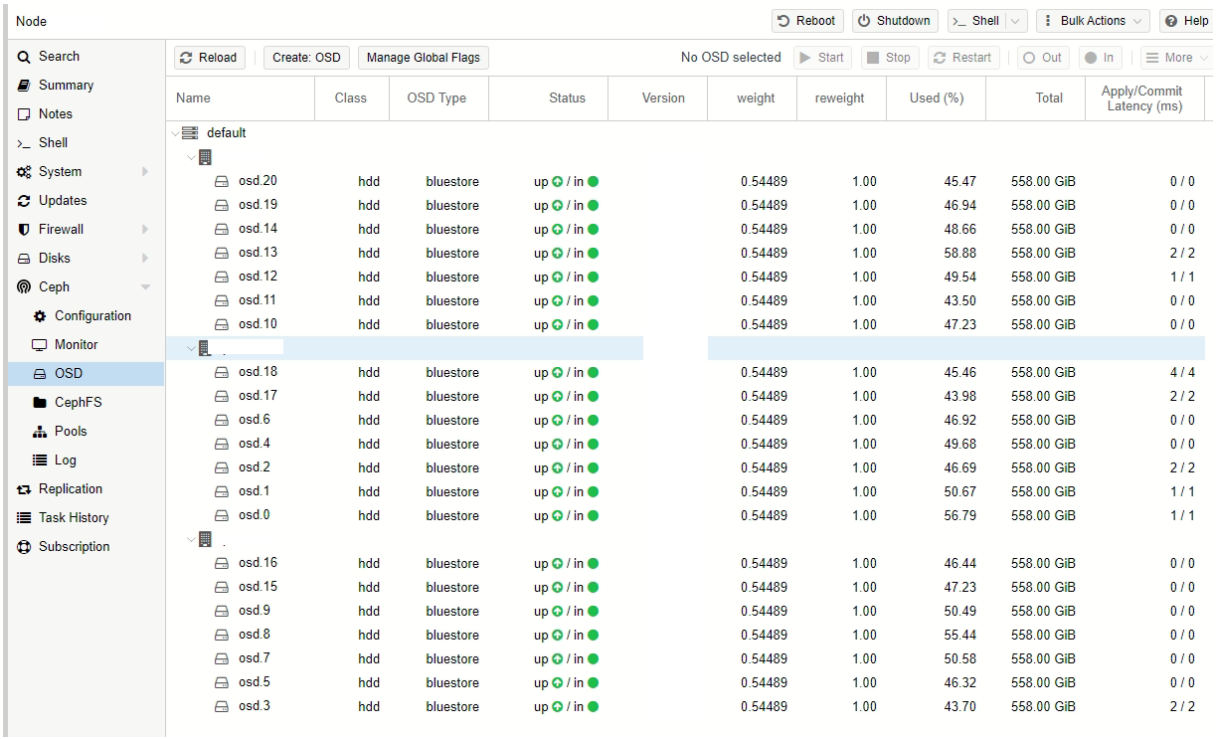


Fonte: Elaborado pelo autor.

Em seguida, foram criados os *OSD* em cada servidor, um para cada disco disponível,

totalizando 21 no *cluster*. A Figura 24 apresenta o resultado desta criação.

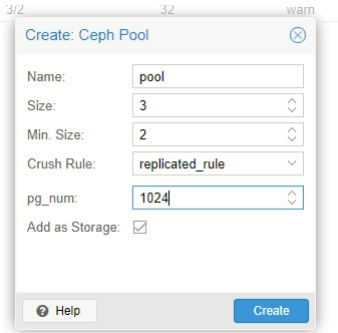
Figura 24 – Resultado da criação dos *OSD*s



Fonte: Elaborado pelo autor.

Por fim, foi criado um *pool* configurado para três replicas e o *PG* com o número 1024, recomendado pela documentação do *Ceph* devido a quantidade de *OSD* estar entre dez e cinquenta. A Figura 25 mostra o processo de criação do *pool*. Com a opção *Add as Storage* o volume fica disponível imediatamente para uso do *Proxmox*.

Figura 25 – Criação do *pool Ceph*



Fonte: Elaborado pelo autor.

O resultado obtido foi um volume de armazenamento de 3.80 *TB*, com três réplicas dos dados distribuídas entre os 21 *HDD*s gerenciados pelos *OSD*s do *Ceph*.

3.3.3 Máquinas virtuais e *HA*

Com a configuração do *cluster* pronta e o *storage* disponível, foram realizadas instalações de *VM*s e *containers*. Os mesmos foram adicionados ao *cluster* de *ha* do *Proxmox*.

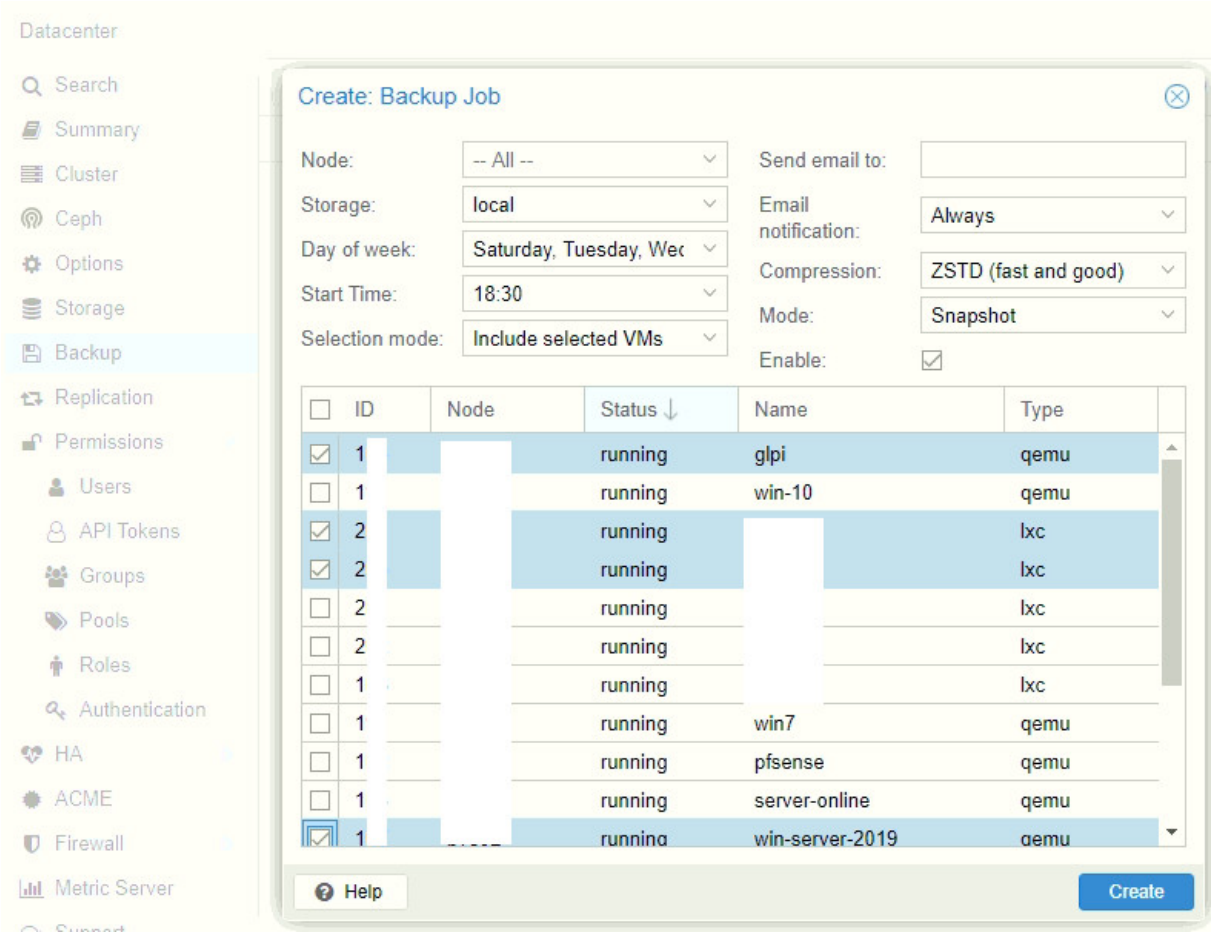
A criação da máquina virtual foi simples e rápida, guiada pelo gerenciador de criação do *Proxmox*. Na primeira etapa foi selecionado um servidor do *cluster Proxmox*, responsável pelo processamento computacional, o identificador da máquina virtual VM ID e um nome. Na segunda etapa foi informado o local da mídia de instalação, o tipo do sistema operacional e sua versão. Na terceira etapa foi feita a escolha do *driver* de vídeo. Na quarta etapa foi feita a criação do disco virtual e escolha do *storage* de armazenamento, no caso o *storage Ceph*. Na quinta etapa foi configurada a CPU com a quantidade de *sockets*, a quantidade de *cores* e o tipo do processador; para o tipo do processador foi selecionado o *host*, que utiliza a ISA do processador físico. Na sexta etapa foi configurada a quantidade de memória. Na sétima etapa foi configurada a rede, com a escolha da interface de rede padrão. Por fim foi feita a revisão e confirmação das configurações da VM e sua primeira inicialização para instalação do SO.

Foram instaladas e testadas máquinas virtuais utilizando os sistemas Windows 7, Windows 10, Windows Server 2016 e Windows Server 2019 em suas versões de avaliação e os sistemas operacionais de base Linux Ubuntu Server nas versões 16.04, 18.04 e 20.04. Os dispositivos de hardware selecionados sempre foram os para-virtualizados, para os sistemas Windows foi preciso baixar uma imagem no padrão ISO 9660 contendo os *drivers* de dispositivos, através do endereço <https://pve.proxmox.com/wiki/Windows_VirtIO_Drivers> e adicionar o arquivo como unidade removível da VM. Para que o HDD da VM fosse reconhecido pelo instalador do Windows foi preciso carregar o *driver virtio*, contido na imagem.

Testes com *containers* foram realizados, a implantação dos mesmos conta com um gerenciador de criação similar ao da VM, porém por se tratar de uma virtualização a nível de SO não existem componentes de hardware no processo. Para a criação do *container* foi necessária a criação da senha de usuário *root* do *container*, a escolha do *template* de criação baixado do repositório, a definição de memória e CPU alocada, a escolha do *storage* e a configuração do endereço de rede.

Para funcionamento dos *containers* e *VM*s em alta disponibilidade estes foram adicionados ao *cluster* HA, por meio de configuração realizada através de acesso ao menu *Datacenter->HA*, acessível também no painel de gerenciamento individual de cada um.

Figura 26 – Configuração do plano de backup



Fonte: Elaborado pelo autor.

3.4 *Backup*

Para o backup foi utilizada a ferramenta do próprio *Proxmox*, nos modos *snapshot* e *stop*. O modo *suspend* não foi utilizado por não ser considerado vantajoso, uma vez que não garante a consistência, como o modo *stop*, e ainda pode ter um tempo de parada da *VM* maior. O modo *snapshot* é o padrão da ferramenta, permite a realização de backups ao vivo, ao custo de uma eventual inconsistência, sem tempo de parada da *VM*.

Foi possível criar planos de *backup* na configuração do *datacenter*, não convém detalhar a criação dos planos, uma vez que estes dependem das necessidades definidas pelas regras de negócio para cada aplicação, contudo a Figura 26 mostra o painel de configuração do plano de *backup*, onde podem ser definidos os dias de execução, horário, seleção de *containers* e *VM*, tipo de compressão de dados, modo de operação e a possibilidade de notificação via e-mail sobre o estado de execução do *backup*.

3.5 Testes

Foram realizados testes de *HA* e migração ao vivo, de *container* e *VM*, recuperação do *storage*, recuperação de servidor físico e *backup*.

Nos testes de *HA* foram simuladas falhas em um servidor executando o processamento de máquinas virtuais, com a desconexão de rede e desligamento forçado. Neste processo também foi possível testar a recuperação do *storage* pois a sincronização de dados ocorre quando um nó volta a integrar o *cluster*.

Nos testes de recuperação do *storage*, alguns *OSD*s foram removidos, simulando uma falha de discos. O algoritmo *CRUSH* atuou fazendo o rebalanceamento do *cluster*, inclusive na recriação dos *OSD*s, simulando a substituição de discos.

A migração ao vivo foi testada utilizando o botão de ação *migrate*, disponível no painel de controle da *VM*, ao executar a ação foi solicitada a escolha do servidor de destino.

No teste de recuperação de um servidor físico, o *Proxmox* foi reinstalado, simulando uma falha no disco de instalação do sistema. Neste teste foi feita uma nova instalação e configuração do servidor, seguindo os passos anteriores, com diferenças da Seção 3.3.2, onde foi feita apenas a instalação do *Ceph* e criação dos *monitors* e *managers*. A criação dos *OSD* não foi necessária, pois os mesmos utilizados na instalação anterior foram restaurados, com o comando "ceph-volume lvm active –all" executado no *shell* do servidor recriado. A criação do *pool* não foi necessária, pois o mesmo continuou preservado nos outros servidores.

O *backup* foi testado com os algoritmos de compressão *Lempel Ziv Oberhumer* (*LZO*), *GZIP* e *ZSTD*, sendo realizado com êxito em *HDD* externo e diretório compartilhado na rede.

Entre as opções de compressão de dados disponíveis o algoritmo *ZSTD* foi o que teve melhor desempenho, realizando o *backup* de uma *VM* de 200 *GB* em uma hora e quarenta e nove minutos, produzindo um arquivo de 45 *GB*. O Algoritmo *GZIP* obteve o pior desempenho, levando cinco horas e quatorze minutos para realizar o *backup* da mesma *VM* e tendo como resultado um arquivo de 52 *GB*. O Algoritmo *LZO* produziu um arquivo de 66 *GB* em uma hora e quarenta e seis minutos. Sem a utilização de compressão o tempo de *backup* foi de uma hora e quarenta e cinco minutos. Neste teste o destino do *backup* foi um dispositivo compartilhado na rede, a *VM* utilizava o SO Windows Server 2019 e era destinada a bancos de dados de teste, estes fatores podem interferir na velocidade e tamanho do *backup*.

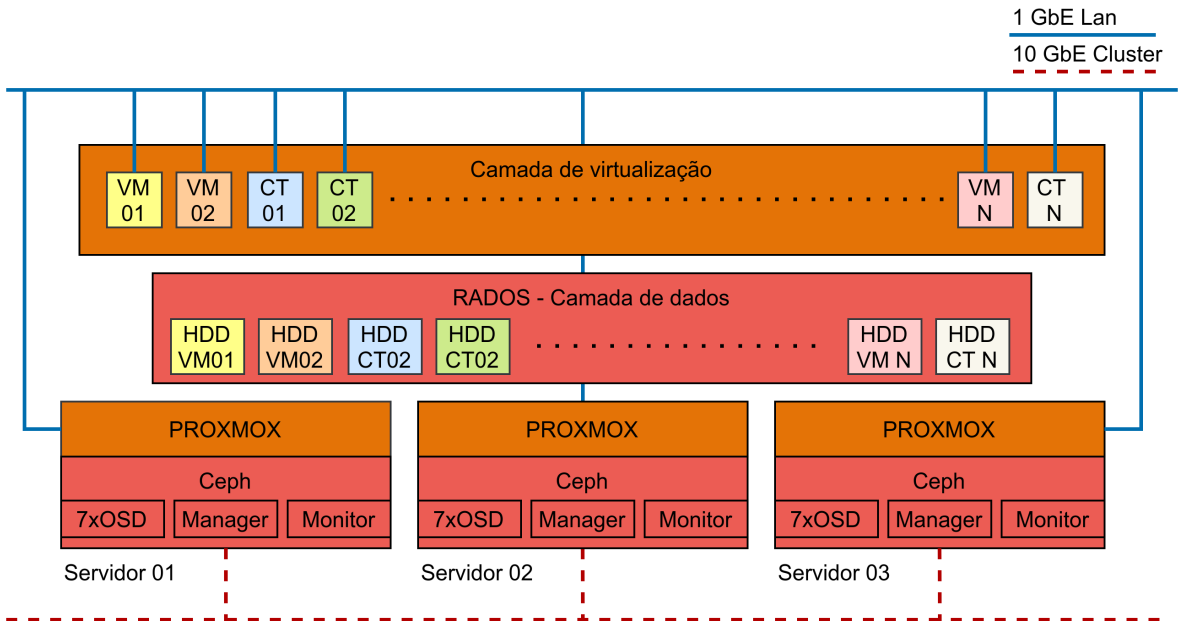
4 Resultados e discussões

Neste Capítulo são abordados os resultados provenientes da aplicação do estudo de caso, apoiado pela pesquisa bibliográfica norteada pelos objetivos deste trabalho. O cumprimento de tais objetivos culmina na implantação da tecnologia de virtualização com ferramentas *open-source*.

Como sabido os objetivos deste trabalho visam solucionar as necessidades de um cartório de Mato Grosso em atender requisitos legais e modernizar a infraestrutura de tecnologia da informação. Tendo em vista as exigências do provimento 74/2018 CNJ foi provida a infraestrutura de tecnologia da informação, solucionando consequentemente as demandas legais e da modernização.

Foi implantada a tecnologia de virtualização com ferramentas *open-source*, por meio de uma solução hiperconvergente, formada pelo *Proxmox* e o *Ceph*, apresentada na Figura 27. A camada de virtualização representa o *Cluster* provido pelo *Proxmox* e a camada de dados representa o *storage* fornecido pelo *Ceph*. Conforme apontado por Sant'Ana (2018) o emprego de uma solução *SDS*, como o *Ceph*, abre portas para a hiperconvergência e esta foi obtida combinada à ferramenta de virtualização.

Figura 27 – Solução de virtualização hiperconvergente implantada



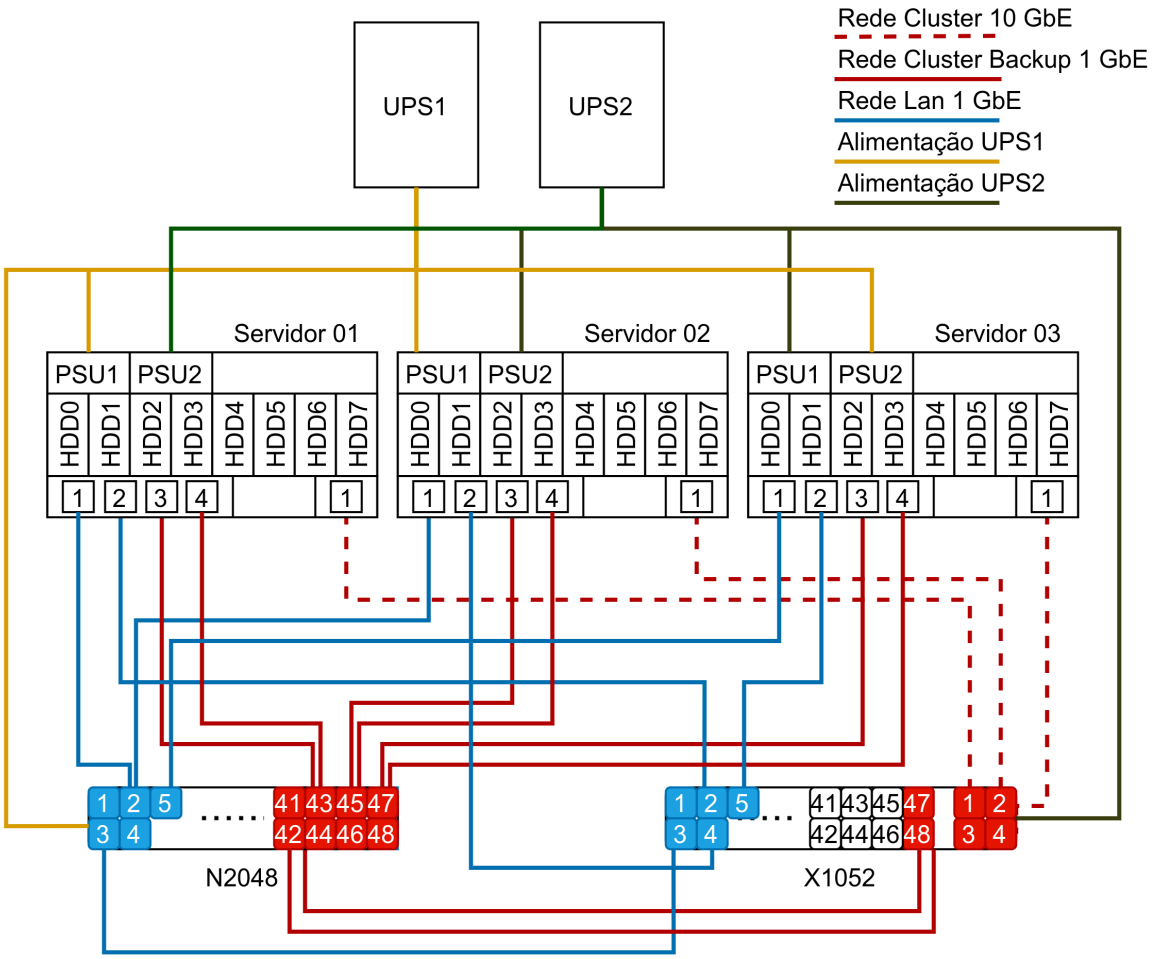
Fonte: Elaborado pelo autor.

A solução hiperconvergente entregue possibilitou economia de recursos financeiros e o aproveitamento do hardware existente.

A instalação do ambiente físico e a configuração lógica efetuada na rede são

representadas sequencialmente nas Figuras 28 e 29. A combinação da instalação física, configuração da rede e a solução hiperconvergente fornecem a HA das VMs, atingindo um dos objetivos deste trabalho. A instalação física e de rede conta com equipamentos e conexões redundantes, conforme apontado por Veras (2011) e Caciato (2015), para eliminar pontos únicos de falha. Veras (2011) define como necessário para a HA o software de gerenciamento do *cluster* e o volume de armazenamento de dados, ambos requisitos contemplados pelo *Proxmox* e *Ceph* respectivamente.

Figura 28 – Ambiente físico obtido após instalações

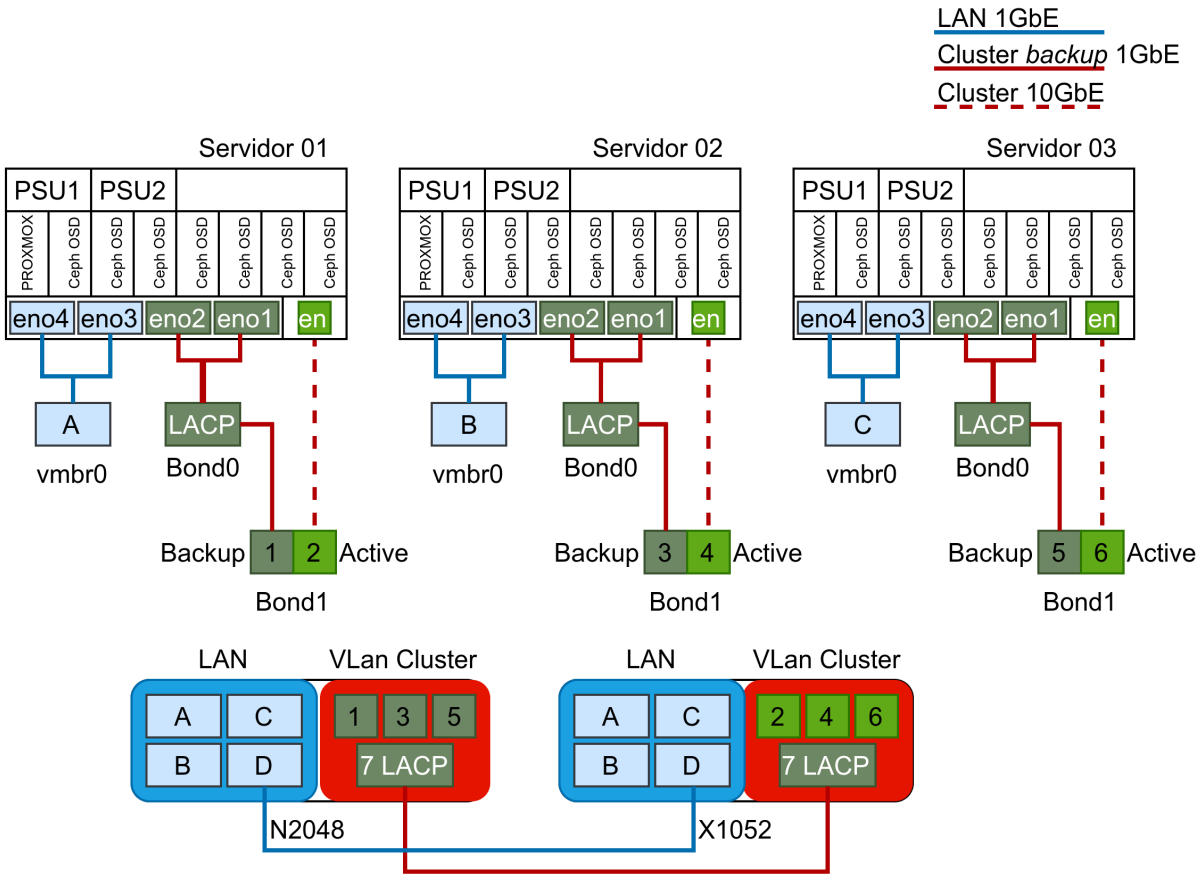


Fonte: Elaborado pelo autor.

Foi provido um volume de armazenamento, no *Proxmox*, para hospedagem das VMs, fornecido pelo *Ceph*, com capacidade total de 3.8 TB e a definição de três réplicas dos dados armazenados. A camada de dados na Figura 27 o representa.

Foram criados processos de *backup*, a partir das necessidades de cada aplicação, utilizando-se da ferramenta de *backup* fornecida pelo *Proxmox*. Foi possível realizar *backup* de *containers* e VMs em dispositivos de compartilhamento de arquivos em rede e HDD externo.

Figura 29 – Organização de rede implantada



Fonte: Elaborado pelo autor.

O ambiente foi disponibilizado para produção e grande parte dos serviços já utiliza-o como base tecnológica. Houve ganho de produtividade e velocidade ao provisionar serviços e aplicações com a utilização de *VM*s e *containers*. O suporte a aplicações legadas foi facilitado.

O *Proxmox* mostrou-se confiável no gerenciamento das *VM*s, além do gerenciamento de *containers*, que promovem economia de espaço em disco e mais agilidade no provisionamento de aplicações. Aliado ao *cluster* de *HA*, que garante, a nível de software, a disponibilidade das *VM*s e *containers*, em caso de falha em algum servidor físico; e sua ferramenta de *backup* que possibilita uma operação mais segura.

O *Ceph* garantiu a disponibilidade e consistência de dados em todos os testes realizados, nenhuma aplicação enfrentou problemas de lentidão, mesmo as de banco de dados, onde existia um receio sobre desempenho.

A virtualização trouxe flexibilidade e desempenho ao ambiente disponibilizado, contribuindo para um plano de recuperação de desastres objetivo, uma vez que a infraestrutura está posta em componentes lógicos e estes podem ser rapidamente restaurados, desde que adotadas as medidas pertinentes, como o *cluster* de *HA*, a replicação de dados

do *storage* e a realização de um plano de *backup*.

A virtualização promove melhor aproveitamento do hardware existente, no entanto deve ser aplicada com cautela. Ao projetar uma solução de virtualização deve ser observada a redundância de componentes de hardware ou a criação de *clusters* de HA. Uma vez que virtualizar vários servidores em apenas um servidor físico pode torná-los indisponíveis em uma eventual falha de hardware.

Considerando os resultados expostos, que foram homologados pelo Cartório, os objetivos deste trabalho foram satisfatoriamente atendidos, validando a solução implantada com o *Proxmox*, o *Ceph* e a configuração realizada no ambiente físico.

Conclusão

Este trabalho apresentou a implantação de um ambiente virtualizado com ferramentas *open-source*, realizada a fim de atender os objetivos estabelecidos. Sua execução seguiu os preceitos extraídos da pesquisa bibliográfica, tendo como resultado final uma solução hiperconvergente que atendeu a todos os objetivos propostos.

Não havia a previsão inicial da hiperconvergência, no entanto o estudo aprofundado das ferramentas utilizadas fez com que esta se concretizasse. Dessa forma, a solução entregue viabilizou a economia de recursos financeiros com o aproveitamento do hardware existente, além da possibilidade de escala e facilidade de manutenção.

A solução implantada encontra-se em produção até a data de redação deste trabalho, fornecendo a base tecnológica para as operações do Cartório.

As técnicas e soluções aqui apresentadas podem ser reproduzidas em ambientes controlados ou reais, havendo qualquer divergência de configurações ou necessidades, os conceitos apresentados podem embasar as mudanças necessárias para funcionamento da solução.

As ferramentas de software escolhidas e os equipamentos de hardware empregados atenderam ao cenário proposto, não sendo uma solução absoluta para todos os casos. As necessidades de cada ambiente devem ser avaliadas no projeto de implantação da virtualização.

Como proposta de evolução e aprimoramento do trabalho realizado há possibilidade de trabalhos futuros no processo de *backup* e no volume de armazenamento.

Para o aprimoramento do processo de *backup* é viável a implantação da ferramenta *Proxmox Backup Server*, que é um servidor dedicado a tarefas de *backup* de *VM*s e *containers* do *Proxmox*, com funcionalidades como a deduplicação e replicação remota de dados.

Prevendo a necessidade de expansão do volume de armazenamento há possibilidade de incrementar a capacidade de armazenamento do *Ceph*, sem comprometer a velocidade e otimizando custos, implementando o mecanismo de *cache* com dispositivos de estado sólido e substituição dos HDDs por outros de maior capacidade e menor rotação por minuto, reduzindo o custo por GB.

Referências

AHMED, W. *Mastering Proxmox*: Master proxmox ve to effectively implement server virtualization technology within your network. Birmingham - Reino Unido: Packt Publishing, 2014. ISBN 978-1-78398-082-6. Citado 2 vezes nas páginas 50 e 51.

BORQUE Álvaro P. *Virtualização de redes*. Monografia (TCC) — UNICAMP, 2013. Disponível em: <https://academica-e.unavarra.es/handle/2454/13383>. Acesso em: 11 mar. 2021. Citado 2 vezes nas páginas 42 e 43.

CACIATO, L. E. *Alta disponibilidade em serviços essenciais utilizando virtualização*. Dissertação (Mestrado) — Universidade Estadual De Campinas, 2015. Disponível em: <http://repositorio.unicamp.br/bitstream/REPOSIP/259640/1/Caciato_LucianoEduardo_M.pdf>. Acesso em: 24 jan. 2021. Citado 4 vezes nas páginas 36, 37, 38 e 62.

CARISSIMI, A. Virtualização: da teoria a soluções. In: *Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos*. [s.n.], 2008. p. 173–207. Disponível em: <https://www.gta.ufrj.br/ensino/CPE758/artigos-basicos/cap4-v2.pdf>. Acesso em: 9 dez. 2020. Citado 8 vezes nas páginas 23, 25, 26, 27, 28, 31, 32 e 33.

CEPH. *Intro to Ceph*. [S.l.], 2021. Disponível em: <https://docs.ceph.com/en/latest/start/intro/>. Acesso em: 17 mar. 2021. Citado na página 51.

CHALKIAS, A. *Ceph storage on VMware*. [S.l.], 2020. Disponível em: <https://ubuntu.com/blog/ceph-storage-on-vmware>. Acesso em: 22 fev. 2021. Citado 3 vezes nas páginas 40, 41 e 50.

CHENG, S. M. *Proxmox High Availability*. Birmingham - Reino Unido: Packt Publishing, 2014. Citado 5 vezes nas páginas 35, 36, 48, 49 e 51.

CONNOR, D. *Citrix acquires XenSource for desktop and server virtualization*. [S.l.], 2007. Disponível em: <https://www.networkworld.com/article/2293987/citrix-acquires-xensource-for-desktop-and-server-virtualization.html>. Acesso em: 1 fev. 2021. Citado na página 36.

CONTROLENET. *Sistemas para armazenamento de dados: Saiba mais sobre DAS, NAS e SAN*. [S.l.], 2017. Disponível em: <https://www.controle.net/faq/sistemas-para-armazenamento-de-dados-das-nas-san>. Acesso em: 17 fev. 2021. Citado na página 41.

COOPER, N. *What is the difference between an HBA and a NIC*. [S.l.], 2019. Disponível em: <https://www.promax.com/blog/what-is-the-difference-between-an-hba-and-a-nic>. Acesso em: 25 fev. 2021. Citado na página 39.

COULOURIS, G. et al. *Sistemas Distribuídos*: Conceitos e projeto. 5. ed. Porto Alegre: Bookman, 2013. ISBN 978-85-8260-054-2. Citado 2 vezes nas páginas 42 e 43.

DATACORE. *Armazenamento Definido por Software*. [S.l.], 2021. Disponível em: <https://www.datacore.com/software-defined-storage/>. Acesso em: 12 fev. 2021. Citado na página 42.

GODOY, M. B. *Segurança em Banco de Dados*. [S.l.], 2011. Disponível em: <https://livrozilla.com/doc/424455/segurança-em-banco-de-dados.indd---n>. Acesso em: 17 mar. 2021. Citado na página 44.

GUEDES, R. *Ethernet bonding – Link Aggregation*. [S.l.], 2018. Disponível em: <https://ricardoguedes.eng.br/wp/blog/2018/01/13/ethernet-bonding-link-aggregation/>. Acesso em: 17 mar. 2021. Citado na página 44.

LAUREANO, M. *Máquinas Virtuais e Emuladores*. Novatec Editora, 2008. Disponível em: <http://www.mlaureano.org/aulas_material/so/livro_vm_laureano.pdf>. Acesso em: 8 fev. 2021. Citado 3 vezes nas páginas 27, 30 e 31.

LAUREANO, M. A. P.; MAZIERO, C. Virtualização: Conceitos e aplicações em segurança. In: *Minicursos do Simpósio Brasileiro de Segurança da Informação e Sistemas - SBSeg*. Sociedade Brasileira de Computação, 2008. p. 151–200. ISBN 978-85-7669-190-7. Disponível em: <https://www.researchgate.net/publication/237681120_Virtualizacao_Conceitos_e_Aplicacoes_em_Seguranca>. Acesso em: 10 dez. 2020. Citado 3 vezes nas páginas 25, 26 e 29.

MATTOS, D. M. F. Virtualização: VMWare e Xen. UFRJ, 2008. Disponível em: <https://www.gta.ufrj.br/grad/08_1/virtual/artigo.pdf>. Acesso em: 7 fev. 2021. Citado 5 vezes nas páginas 23, 27, 29, 35 e 36.

MICROSOFT. *Visão geral da tecnologia Hyper-V*. [S.l.], 2016. Disponível em: <https://docs.microsoft.com/pt-br/windows-server/virtualization/hyper-v/hyper-v-technology-overview>. Acesso em: 10 fev. 2021. Citado na página 35.

MICROSOFT. *Evaluation Center*. [S.l.], 2021. Disponível em: <https://www.microsoft.com/pt-br/evalcenter/evaluate-hyper-v-server-2019>. Acesso em: 10 fev. 2021. Citado na página 35.

MORAIS, R. da S. L. *Utilização de Armazenamento Definido por Software em uma Arquitetura Hiperconvergente de Containers*. Dissertação (Mestrado) — Universidade de Brasília, Brasília, jul. 2019. Disponível em: <https://repositorio.unb.br/handle/10482/36759>. Acesso em: 20 fev. 2021. Citado na página 42.

NASCIMENTO, F. A. et al. Alta disponibilidade em virtualização de servidores. estudo de caso: Universidade estadual de ciências da saúde de alagoas UNCISAL. In: *VII SEGeT – Simpósio de Excelência em Gestão e Tecnologia*. Resende: [s.n.], 2010. Disponível em: <https://www.aedb.br/seget/arquivos/artigos10/26_Virtualizacao_SEGET_2010.pdf>. Acesso em: 22 fev. 2021. Citado 2 vezes nas páginas 37 e 38.

PAVAN, A. J. et al. Alta disponibilidade em servidores virtualizados. FATEC, Bauru, 2014. Disponível em: <http://www.fatecbauru.edu.br/mtg/source/AltaDisponibilidadeemServidoresVirtualizados.pdf>. Acesso em: 22 fev. 2021. Citado 2 vezes nas páginas 36 e 39.

POPEK, G. J.; GOLDBERG, R. P. Formal requirements for virtualizable third generation architetures. In: *Commuications of the ACM*. [s.n.], 1974. v. 17, n. 7, p. 412–421. Disponível em: <http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.141.4815&rep=rep1&type=pdf>. Acesso em: 6 jan. 2021. Citado na página 27.

Proxmox Server Solutions Gmbh. *PROXMOX VE administration guide*. [S.l.], 2020. Disponível em: <https://pve.proxmox.com/pve-docs/pve-admin-guide.pdf>. Acesso em: 12 dez. 2020. Citado 4 vezes nas páginas 36, 48, 49 e 52.

ROCHA JUNIOR, R. R. *Agregação dinâmica de enlaces em ambientes de redes definidas por software*. Monografia (Mestrado) — Universidade Federal de Minas Gerais, 2017. Disponível em: <https://repositorio.ufmg.br/bitstream/1843/JCES-AVPP3P/1/ronaldoresenderochajr.pdf>. Acesso em: 21 fev. 2021. Citado na página 44.

RODRIGUES, W. F. *Análise dos procedimentos de backup dos Institutos Federais*. Dissertação (Mestrado) — Universidade Federal de Pernambuco UFPE, 2017. Disponível em: <https://repositorio.ufpe.br/handle/123456789/25612>. Acesso em: 17 mar. 2021. Citado 2 vezes nas páginas 44 e 45.

SANT'ANA, A. *o que é Software Defined Storage (SDS) e hiperconvergência?* [S.l.], 2018. Disponível em: <https://www.arles-santana.com.br/2018/02/18/o-que-e-software-defined-storage-sds-e-hiperconvergencia/>. Acesso em: 22 fev. 2021. Citado 2 vezes nas páginas 42 e 61.

SANTOS, M. M. d. *Proposta de armazenamento distribuído baseado em contêineres em infraestrutura hiperconvergente*. Monografia (TCC) — Instituto Federal de Santa Catarina - IFSC, 2019. Disponível em: <https://wiki.sj.ifsc.edu.br/images/d/d9/TCC_2_MatuzalemMullerDosSantos_Digital.pdf>. Acesso em: 9 dez. 2020. Citado 3 vezes nas páginas 43, 50 e 51.

SCOTT, S. *Ceph – Part 3 – Technical architecture and components*. [S.l.], 2015. Disponível em: <https://www.supportsages.com/ceph-part-3-technical-architecture-and-components/>. Acesso em: 22 fev. 2021. Citado 2 vezes nas páginas 50 e 52.

SHAIKH, K. et al. Network attached storage. *International Research Journal of Engineering and Technology IRJET*, v. 06, p. 1413–1416, jan 2019. Disponível em: <https://irjet.com/archives/V6/i1/IRJET-V6I1259.pdf>. Acesso em: 4 mar. 2021. Citado na página 41.

SILVA, C. A. R. d. *Estudo de caso da implementação da ferramenta PROXMOX no gerenciamento de recursos do datacenter do 52º Centro de Telemática*. Monografia (TCC) — Escola de aperfeiçoamento de oficiais escola de formação complementar do exército, 2019. Disponível em: <https://bdex.eb.mil.br/jspui/handle/123456789/5257>. Acesso em: 9 dez. 2020. Citado 2 vezes nas páginas 48 e 49.

SILVA, R. F. d. *Virtualização de Sistemas Operacionais*. Monografia (TCC) — Instituto Superior de Tecnologia em Ciências Da Computação - ISTCC, 2007. Disponível em: <https://www.lncc.br/~borges/doc/VirtualizacaodeSistemasOperacionais.TCC.pdf>. Acesso em: 6 jan. 2021. Citado 2 vezes nas páginas 35 e 36.

TAKEUTI, A. Y. Virtualização em ambientes corporativos. *Perspectivas em Ciências Tecnológicas*, v. 3, 2014. Disponível em: <https://fatece.edu.br/arquivos/arquivos-revistas/perspectiva/volume3/2.pdf>. Acesso em: 6 jan. 2021. Citado 2 vezes nas páginas 23 e 35.

TOLFO, A. *Implantação de uma infraestrutura de hiperconvergência open source com PROXMOX*. Monografia (TCC) — Antonio Meneghetti Faculdade, 2019. Disponível em:

<http://repositorio.faculdadeam.edu.br/xmlui/handle/123456789/647>. Acesso em: 9 dez. 2020. Citado 5 vezes nas páginas 23, 48, 49, 50 e 51.

TOSADORE, R. de A. *Virtualização: alta disponibilidade, performance e redução de custos*. Monografia (TCC) — Faculdade POLITEC, 2012. Disponível em: <https://cupdf.com/document/tcc-virtualizacao-rafaeltosadore-2012.html>. Acesso em: 2 fev. 2021. Citado 2 vezes nas páginas 39 e 40.

VERAS, M. *Virtualização*: Componente central do datacenter. Rio de Janeiro: Brasport, 2011. Citado 20 vezes nas páginas 23, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 36, 37, 39, 40, 41, 42, 44, 45 e 62.

VMWARE. *VMware ESX e VMware ESXi*. [S.l.], 2009. Disponível em: <https://www.vmware.com/files/br/pdf/products/VMW_09Q1_BRO_ESX_ESXi_BR_A4_P6_R2.pdf>. Acesso em: 17 jan. 2021. Citado na página 35.

VMWARE. *Armazenamento definido por software (SDS)*. [S.l.], 2021. Disponível em: <https://www.vmware.com/br/products/software-defined-storage.html>. Acesso em: 17 fev. 2021. Citado 2 vezes nas páginas 41 e 42.