UNIVERSIDADE FEDERAL DO ESTADO DO RIO DE JANEIRO
CENTRO DE CIÊNCIAS EXATAS E TECNOLOGIA
PROGRAMA DE PÓS-GRADUAÇÃO EM INFORMÁTICA

APLICAÇÃO E COMPARAÇÃO DE AMBIENTES DE VIRTUALIZAÇÃO UTILIZANDO SOFTWARES LIVRES PROXMOX E XCP-NG

Michel Augusto Queiroz Morais

Orientador

MSc. Maximiliano Martins de Faria

Co-orientador

Prof. Sidney C. de Lucena

RIO DE JANEIRO, RJ - BRASIL
AGOSTO DE 2019

Augusto Queiroz Morais, Michel
A827 Aplicação e comparação de ambientes de virtualização utilizando softwares livres Proxmox e XCP-ng / Michel Augusto Queiroz Morais. – Rio de Janeiro, 2019.
69 f.

Orientador: Maximiliano Martins de Faria.
Co-orientador: Sidney Cunha de Lucena
Trabalho de Conclusão de Curso (Graduação) - Universidade Federal do Estado do Rio de Janeiro, Graduação em Sistemas de Informação, 2019.

1. Virtualização 2. Hipervisor. 3. Proxmox 4. XCP. I. Martins de Faria, Maximiliano, orient. II. Cunha de Lucena, Sidney. coorient. III .Título.

APLICAÇÃO E COMPARAÇÃO DE AMBIENTES DE VIRTUALIZAÇÃO UTILIZANDO SOFTWARES LIVRES PROXMOX E XCP-NG

Michel Augusto Queiroz Morais

Projeto de Graduação apresentado à Escola de Informática Aplicada da Universidade Federal do Estado do Rio de Janeiro (UNIRIO) para obtenção do título de Bacharel em Sistemas de Informação.

Aprovada por:

DSc. Sidney Cunha de Lucena — UNIRIO

MSc. Maximiliano Martins de Faria — UNIRIO

DSc. Morganna Carmem Diniz — UNIRIO

DSc. Leonardo Luiz Alencastro Rocha — UNIRIO

RIO DE JANEIRO, RJ - BRASIL
AGOSTO DE 2019

Agradecimentos

Dedico esse trabalho aos meus pais Anibal e Sandra, que muito se dedicaram para que eu concluísse essa etapa de meus estudos, participando dessa caminhada com seu apoio e dedicação, para que os obstáculos fossem vencidos durante minha trajetória acadêmica. Dedico esse momento também aos meus coordenadores Sidney e Max que, com sua riqueza de conhecimentos, de maneira impecável, acompanharam a execução desse processo, despertando em mim o gosto pela pesquisa e o interesse pela perfeição, eficiência e clareza na exposição de ideias.

RESUMO

Escolher um *software* de virtualização de qualidade para uma organização é uma tarefa árdua em virtude dos variados tipos de *softwares* de virtualização presentes no mercado. Em virtude de haver poucos estudos técnicos comparativos sobre o funcionamento desses *softwares*, este trabalho busca comparar e analisar duas soluções de virtualização livres: o *Proxmox* e *XCP-ng*. Através de uma análise comparativa, buscamos identificar qual deles seria a melhor opção para substituir o *software* atualmente usado no ambiente de *datacenter* do Centro de Ciências Exatas e Tecnologia (CCET) da Universidade Federal do Estado do Rio de Janeiro - UNIRIO, cujo custo de atualização se tornou proibitivo. Devido a essa complicação para se obter licenças de utilização de *softwares* de virtualização pagos, esta pesquisa se baseia na comparação das funcionalidades dos dois *softwares* de virtualização gratuitos citados, almejando assim o benefício de se ter um *software* de virtualização de alta qualidade sem precisar arcar com custos de licença.

Palavras-chave: Virtualização, Hipervisor, Proxmox, XCP.

Abstract

Choosing a good virtualization software for an organization, is a daunting task because of the many types of virtualization software on the market. Due to the few technical studys conducted to compare the functioning of these softwares, this paper seeks to compare and analyze two existing free virtualization solutions: *Proxmox* and *XCP-ng*. Through a comparative analysis, we sought to identify which one would be the best option to replace the software currently used in the datacenter of the Center for Exact Sciences and Technology (CCET) of the Federal University of the State of Rio de Janeiro - UNIRIO, in which the cost for upgrades have become prohibitive. Due to these expenses in obtaining licenses for commercial virtualization softwares, this research is based on comparing the functionalities of the two free virtualization software cited above, thus aiming the benefit of having a high-quality virtualization software without the costs of license fees.

Palavras-chave: Virtualization, Hypervisor, Proxmox, XCP.

Sumário

Lista de Figuras

Lista de Tabelas

1. Introdução

A técnica de virtualização de estruturas de TI é um grande marco em serviços de tecnologia. Hoje, diversas empresas oferecem esse tipo de arquitetura, garantindo aos seus clientes serviços virtualizados e em nuvem. Como exemplo desses serviços, podemos citar o de *streaming* de vídeo, onde existem *datacenters* mantendo toda uma estrutura virtualizada disponível vinte quatro horas por dia, de forma escalável e sob demanda. Trata-se, portanto, de uma técnica de gerenciamento que visa distribuir os recursos computacionais disponíveis numa máquina física entre diversas máquinas "lógicas" coexistentes, de maneira a reduzir custos e otimizar o funcionamento desses múltiplos sistemas operacionais rodando sobre um mesmo *hardware* (Patel, Daftedar, Shalan, & El-Kharashi, 2015).

A virtualização utiliza um *software* como camada de abstração entre o *hardware* e os sistemas operacionais convidados e suas aplicações. Essa camada, chamada de *hypervisor*, ou hipervisor, gerencia os recursos físicos do servidor virtualizador entre os sistemas operacionais convidados (Sahoo, Mohapatra, & Lath, 2010).

As tecnologias usadas para fins de virtualização vêm sendo aperfeiçoadas ao longo dos anos, possuindo diferentes versões e *softwares* disponíveis no mercado, competindo entre si. De forma geral, a virtualização visa fornecer um ambiente seguro para desenvolvimento e infraestrutura com disponibilidade, confiabilidade e escalabilidade de recursos. Com o passar dos anos, essas tecnologias vêm sendo melhoradas objetivando a facilidade no seu manuseio e gerenciamento.

1.1 Motivação

Diante do surgimento de novos cenários de TI com ambientes tecnológicos diversos, cada vez mais desenvolvidos e com tecnologias novas surgindo, testes entre soluções de virtualização se fazem necessários. Nesse sentido, é importante executar testes comparati-

vos para um melhor embasamento na escolha da solução mais adequada. Existem diversas opções de *softwares* para virtualização e a escolha da melhor solução não é simples, já que a mesma deve levar em conta vários fatores que abrangem preço, gerenciamento, licenciamento, instalação entre outros. Na medida que a virtualização de servidores começou a ser usada nos *datacenters* ao redor do mundo, a mesma tornou-se praticamente obrigatória, uma vez que propiciou um melhor uso dos recursos computacionais de um servidor, possibilitando uma melhor escalabilidade do seu poder de processamento, melhor gerenciamento de recursos de memória e mais segurança a partir do isolamento entre recursos, dentre outras melhorias. Diante disso, para a adoção de uma tecnologia de virtualização adequada a um dado cenário é crucial que a escolha por uma solução seja baseada em testes experimentais, frente à quantidade de soluções já disponíveis no mercado.

Outro ponto importante é o custo de licenciamento de um *software* de virtualização, uma vez que as melhores soluções são proprietárias e o custo de aquisição e manutenção são muito altos. Como as soluções de virtualização são úteis e, algumas vezes, imprescindíveis para qualquer tipo de ambiente, mesmo com os custos de licenciamento sendo reduzidos ao longo do tempo isso ainda pode ser um fator limitante para algumas organizações, como por exemplo uma universidade pública. Essa limitação pode ser, no entanto, contornada com a utilização de *softwares* de virtualização com licenças de uso livre, uma vez que existem diversas opções desse tipo no mercado.

A partir dessa realidade, este trabalho se propõe a comparar dois *softwares* de virtualização livres, escolhidos mediante alguns critérios e analisar qual melhor se comportaria segundo determinados parâmetros. Através da implementação e comparação desses dois softwares de virtualização livres, e com a consequente análise das informações obtidas, podemos entender as diferenças, vantagens e desvantagens na adoção dos mesmos. Dessa forma, entende-se que a análise obtida poderá ser benéfica para aqueles interessados em utilizar esses dois *softwares* e para possíveis trabalhos futuros.

1.2 Objetivos do trabalho

Este trabalho tem a intenção de estudar e entender a viabilidade de instalação e a eficácia de uso de duas soluções de virtualização, baseadas em *softwares* livres, no *datacenter* de uma universidade pública brasileira, a UNIRIO. Atualmente, a UNIRIO tem um ambiente de virtualização utilizando a solução *VMWare* versão 5.5, que está desatualizada. Seu *upgrade*, no entanto, é caro e, somado ao custo de manutenção, torna sua aquisição inviável. Nesse sentido, este estudo se propõem analisar duas soluções de virtualização

open source, que se apresentam como melhores alternativas à substituição de todas as funcionalidades da ferramenta proprietária instalada atualmente: as soluções *Proxmox* e *XCP-ng*.

Para tal, pretende-se analisar diversos quesitos técnicos dessas soluções, desde instalações até configurações e gerenciamentos. As instalações partirão do "zero" até o ambiente estar pronto para ações de configuração de *clusters*, criação de máquinas virtuais, adição de *storages*, rotina de *backups*, migração de máquinas e resposta a desastres, entre outras. Além disso, este trabalho é uma oportunidade para se estudar os aspectos técnicos de soluções de virtualização, como também criar um guia para que profissionais ou estudiosos da área possam usá-lo na escolha da melhor solução para seus *datacenters*.

A escolha do ambiente na universidade se mostrou muito útil para a realização das instalações e comparação dos dois *softwares* escolhidos. O resultado desse estudo pode ser usado como base para a tomada de decisão de outros gestores que desejem reduzir custos com licenças dos *softwares*, sem abrir mão de um ambiente robusto com todas as funcionalidades de um *datacenter* moderno.

1.3 Estrutura do texto

O presente trabalho está estruturado em capítulos e, além desta introdução, organiza-se da seguinte forma:

- O Capítulo II consiste na revisão da bibliografia das tecnologias de virtualização existentes, com intenção de prover mais conhecimento sobre o assunto e mapear as principais tendências e funcionalidades de um ambiente de virtualização.
- O Capítulo III trata da metodologia utilizada para realização do estudo. Esta capítulo é a base conceitual para os testes que foram usados na comparação dos dois *softwares*.
- O Capítulo IV trata da instalação, configuração e dos resultados obtidos para as comparações entre as funcionalidades mais importantes das soluções testadas.
- O Capítulo V traz as considerações finais, assinalando as contribuições da pesquisa e sugerindo possibilidades de aprofundamento posterior.

2. Revisão Bibliográfica

2.1 Visão geral

A tecnologia de virtualização teve inicio em 1960 elencada pela IBM. Nesta década surgiram os primeiros computadores de grande porte com algoritmos de programação, que dividia o tempo de execução de processos, agindo como gerenciador de recurso e que possibilitava um mesmo sistema rodar mais de um processo em pedaços de tempo alternados, um pedaço para cada processo (Sahoo et al., 2010). Este recurso possibilitou a abstração e o mapeamento de processador e memória reais para uma forma virtual, ou seja, os processos do sistema poderiam compartilhar os recursos destes. A partir destas técnicas surgiram as primeiras formas de virtualização, onde inicialmente uma máquina com sistema operacional **A** executa uma aplicação que roda outra máquina com sistema operacional **B**, encapsulada como um processo, como mostra a figura 2.1. Cada máquina virtual é semelhante à estação real, mas seus dispositivos físicos são virtuais (emulação de *hardware* via *software* de *hardwares*) e oriundos da máquina física, chamada de estação *host*.

Essa nova forma de implementação de máquinas virtuais, com a abstração de recursos de *hardware*, permitia a emulação completa de processadores, memória e outros dispositivos físicos. Apesar do grande avanço e a possibilidade de um *mainframe* rodar outros sistemas embutidos, em geral, essa técnica era desvantajosa, já que o desempenho do sistema convidado (virtual) era sacrificado. Nessa técnica existe todo um processo, que ocorre em modo de usuário, de um sistema dentro do outro. É como se o sistema operacional **B** fosse uma aplicação rodando dentro do sistema operacional **A**, e com isso fica claro que, na medida que novas máquinas convidadas forem implementadas, os sistemas tanto real quanto virtual sofrerão uma grande perda de desempenho. Em virtude dessa perda, surgiram os monitores de máquinas virtuais(*Virtual Machine Monitor* - VMM) conhecidos também como *hipervisores*. Eles operam como uma camada entre o sistema

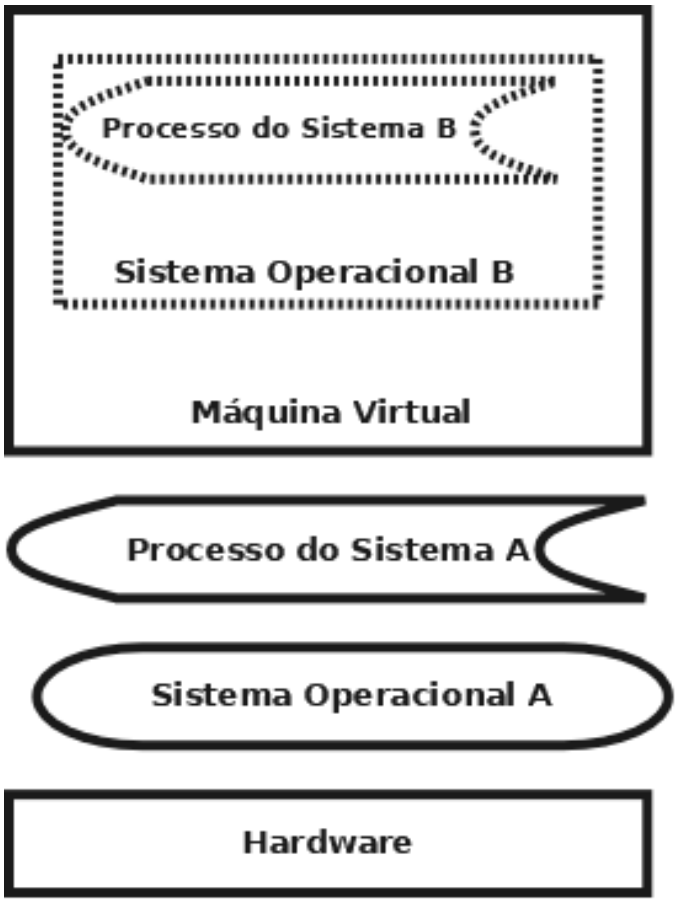


Figura 2.1: Representação de Máquina virtual

operacional convidado e o *hardware* físico do *host*, vide figura 2.2.

Os *hipervisores* possuem acesso direto aos recursos físicos da máquina real, tornando-o um gerente, que por sua vez melhora o despenho da camada de abstração. Existem duas técnicas de operação utilizadas nos *hipervisores*: **virtualização total** e **paravirtualização**. A diferença entre as duas é que, na **paravirtualização**, o sistema operacional convidado é uma versão modificada específica para a virtualização e, na **virtualização completa**, o sistema operacional convidado funciona sem saber que é uma máquina virtual. A virtualização funciona através dessa camada de abstração chamada *virtual machine monitor* (VMM), ela conecta o sistema virtual aos recursos físicos da máquina real. Para o sistema operacional convidado (ou virtual) é como se a máquina virtual tivesse todos os recursos de uma estação física, mas, na verdade, é tudo gerenciado e simulado pelo *host* através de um *software* chamado *hypervisor*.

Como os recursos que são entregues às estações virtuais são controlados pelo *hipervisor*, é possível ter diferentes sistemas operacionais rodando na mesma máquina física. Cientistas perceberam que a possibilidade de particionar os sistemas permite que muitos processos e aplicações rodem ao mesmo tempo, assim aumentando a eficiência do ambi-

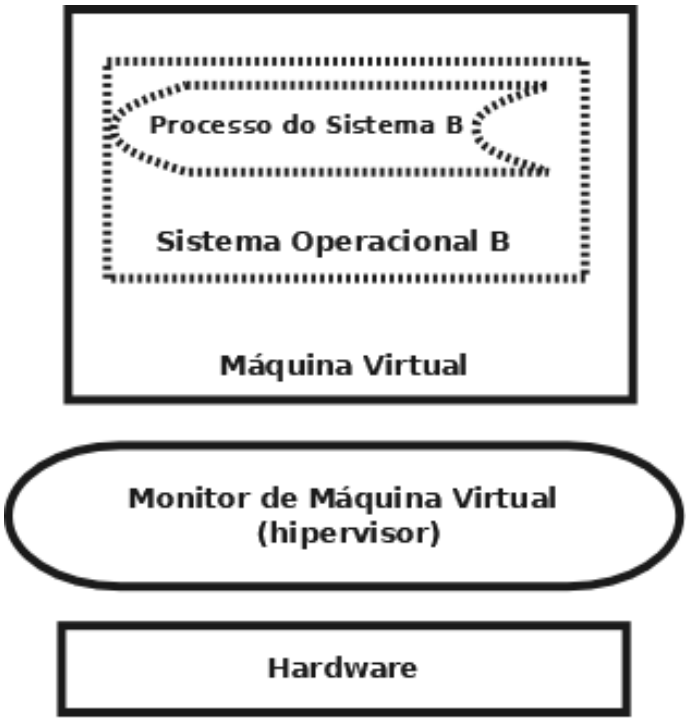


Figura 2.2: Representação de Máquina virtual

ente e diminuindo custos (Sahoo et al., 2010). Com esse fato, surgiu a primeira ideia de Máquina Virtual pela IBM, baseada no conceito de *hipervisor*. Essa visão permitia que vários sistemas operacionais diferentes executassem sobre a mesma grande máquina da época o *mainframe*. No contexto dos anos 1970, quando a maioria dos *mainframes* utilizavam os mesmos sistemas operacionais, foi possível a migração das aplicações legadas ao novo sistema virtual gerenciado pelo *hipervisor*.

Com a popularização dos computadores pessoais na década 80 e a redução do número de sistemas operacionais utilizados, a necessidade de máquinas virtuais foi perdendo importância. Na década de 90, a virtualização ganhou força novamente, principalmente por alguns fatores, como o aumento do poder computacional de processadores, a criação de máquinas com multiprocessamento, a disseminação de sistemas distribuídos e a expansão da internet com modelo de arquitetura cliente-servidor. O marco inicial foi o desenvolvimento da *VMware* em 1998, a primeira a desenvolver o primeiro *hipervisor* capaz de virtualizar servidores em plataformas x86. Outras empresas tratavam da virtualização através de sistemas de parceiros, como o desenvolvido pela empresa Connectix. Nessa época, o mercado crescente de microcomputadores e servidores x86 percebeu que as máquinas estavam operando acima da necessidade e não necessariamente utilizando todo o poder de processamento disponível. Nesse sentido, foi possível elaborar arquiteturas extremamente simples nas quais podiam se dividir recursos entre várias máquinas virtuais e ainda manter um bom desempenho. Nos anos 2000, o cenário da virtualização continuava

ganhando força, muitos *datacenters* começaram a migrar suas soluções de servidores para virtualização. A possibilidade da redução de servidores tinha um apelo forte tanto pela economia de energia e menor espaço físico. Em 2003 a Microsoft comprou a Connectix e também entrou na disputa pelo mercado da virtualização com o lançamento do *Microsoft Virtual Server 2005*, o primeiro produto com foco total na virtualização de servidores. Nos anos seguintes a virtualização se consolidou e atualmente é condição “sine qua non", com a crescente demanda por serviços *on-line*. Os *datacenters* precisam de mais servidores e estes, por sua vez, devem estar preparados para maior escalabilidade e menor tempo de parada possível, o que vai de encontro com a virtualização.

2.2 Características da Virtualização

A virtualização pode ser feita de diversas formas através da emulação de vários componentes dos computadores (Veras & Carissimi, 2019):

- **Virtualização de Servidores**: é a forma mais comum e simples. Diferente da época dos *mainframes*, a virtualização é feita em servidores x86. O servidor trata-se de uma máquina de alto desempenho capaz de fornecer recursos para um grande número de máquinas virtuais. Como se fosse uma grande máquina única oferecendo vários serviços.

- **Virtualização de *desktops***: trata da configuração dos *desktops* dos usuários finais em uma infraestrutura centralizada virtual. Isso significa que as aplicações de *desktop* também passam a executar em um *datacenter*, sob a forma de máquinas virtuais. Esse é o conceito de *Virtual Desktop Infrastructure* (VDI), que permite a montagem dinâmica de *desktops*, oferecendo maior confiabilidade e otimização do uso de espaço em disco com a consolidação do armazenamento e flexibilidade na escolha do sistema operacional. Existem limitações para o uso dessa técnica de forma generalizada. Normalmente, sua adoção é antecedida por um trabalho de levantamento da situação a ser considerada.

- **Virtualização de armazenamento (*storage*)**: a ideia é introduzir um componente que permite às diversas unidades heterogêneas de armazenamento (discos físicos) serem vistas como um conjunto homogêneo de recursos. A virtualização do armazenamento não é tão popular quanto a de servidores, mas atualmente acompanha as soluções de virtualização.

- **Virtualização das aplicações**: trata do conceito de execução do programa por com-

pleto, em um repositório central, permitindo a configuração centralizada do aplicativo, o que melhora seu gerenciamento por permitir que seja feita em um único lugar.

- **Virtualização de redes**: arquitetura que proporciona um ambiente de rede separado para cada grupo ou organização. Esses ambientes lógicos são criados sobre uma única infraestrutura compartilhada de rede. Cada rede lógica fornece ao grupo de usuários correspondente plenos serviços de rede, semelhantes aos utilizados por uma rede tradicional não virtualizada. A experiência da perspectiva do usuário final é a de ter acesso a uma rede própria, com recursos dedicados e políticas de segurança independentes. Assim, envolve a segmentação da rede de transportes, dos dispositivos e de todos os serviços. Como as diversas redes lógicas compartilham uma infraestrutura comum, muitas vezes centralizada e com um conjunto de equipamentos e servidores, os grupos de usuários podem colaborar com maior flexibilidade e capacidade de gerenciamento, o que permite novos processos de negócios que não seriam possíveis (e nem sequer imagináveis) por meio de uma rede tradicional.

2.3 Conceitos e categorização dos *Softwares* de Virtualização

A forma de operação dos *softwares* de virtualização pode ser categorizada em três formas (Veras & Carissimi, 2019):

- **Nível do *Hardware***: A camada de virtualização é posta diretamente sobre a máquina física e a apresenta às camadas superiores como *hardware* similar ao original, vide figura 2.3. Esse é o caso da maioria dos *hipervisores*, como *VMware ESX, Xen e Hyper-V*.

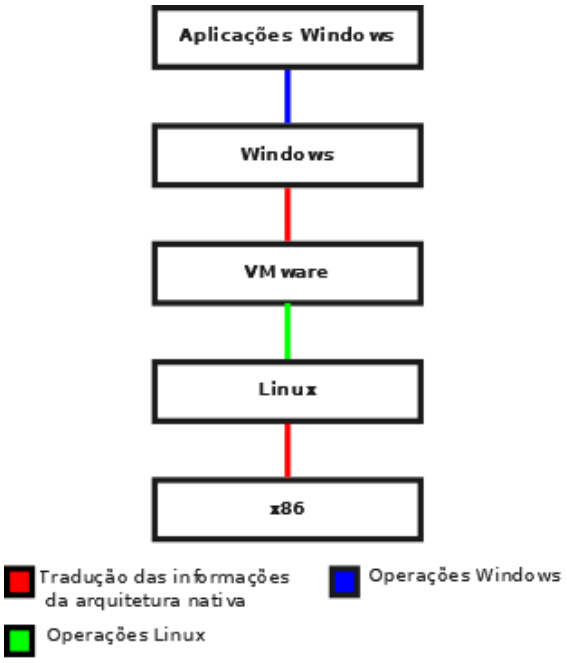


Figura 2.3: Virtualização na forma *Hardware*

- **Nível do Sistema Operacional(NSO)**: A camada de virtualização está inserida entre o sistema operacional e as aplicações, como mostra a figura 2.4. Isso permite a criação de partições lógicas, onde cada uma é uma máquina virtual isolada compartilhando o mesmo sistema operacional. Exemplos: *Jails, OpenVZ, Solaris Zones, Containers, Linux-Vserver, Parallels Virtuozzo, SandBox, KVM e Oracle VirtualBox*).

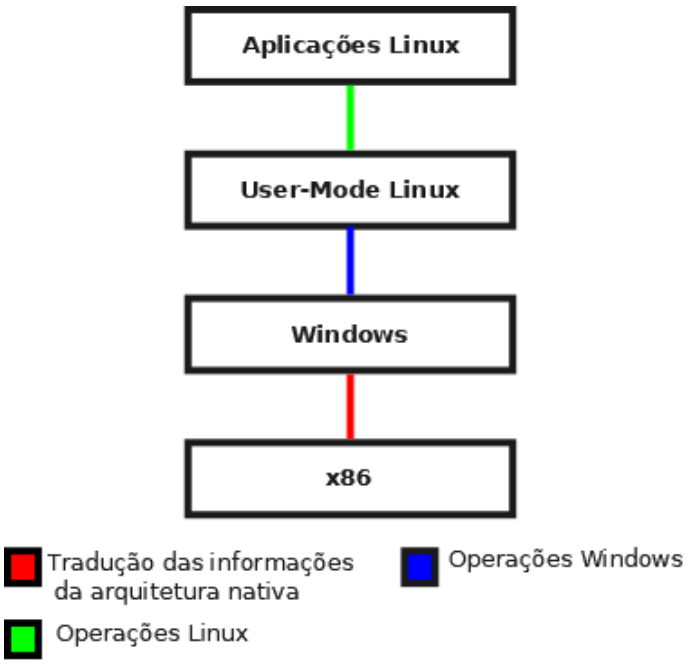


Figura 2.4: Virtualização na forma de sistema operacional

- **Nível de linguagem de programação**: Neste, a camada de virtualização é um programa do sistema operacional da máquina física. Ele cria uma máquina virtual abstrata, sobre a qual executa instruções específicas para rodarem naquela máquina criada em cima do sistema operacional nativo. O programa que for desenvolvido com essas instruções específicas precisa apenas desta máquina virtual para rodar que, por sua vez, fica encarregada de traduzir essas ações em ações do sistema operacional abaixo, como mostra figura 2.5. A máquina virtual *Java* (JVM) e o *DotNet* da *Microsoft* são os exemplos mais marcante.

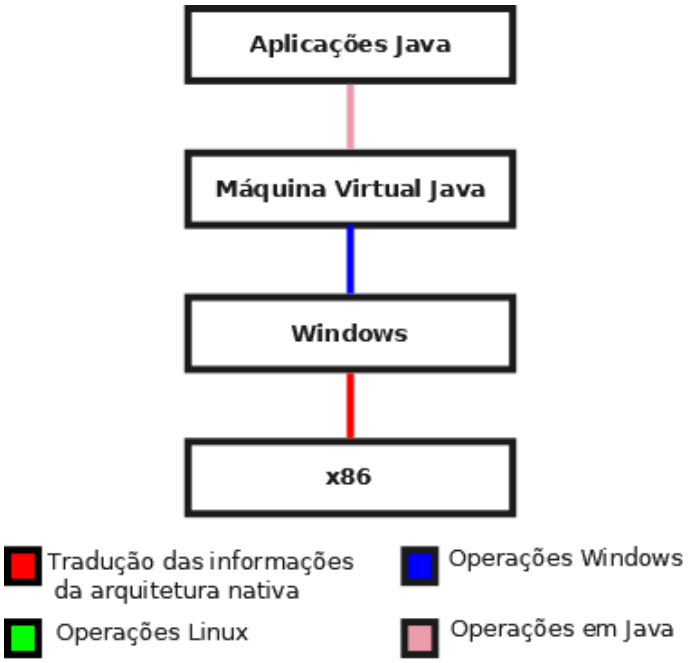


Figura 2.5: Virtualização na forma de Linguagem de Programação

A formas de operação possuem finalidades diferentes, mas o objetivo final são os mesmos:

- Isolar o ambiente das máquinas virtuais, de modo que uma não afete a execução de outra;
- Fornecer compatibilidade de *software*;
- Encapsular a máquina virtual de modo a permitir sua manipulação e a captura de seu estado.

A camada de virtualização (*hypervisor*) foi desenvolvida com intenção não só de melhorar o desempenho, mas também de não impactar o desempenho das aplicações rodando nas máquinas virtuais. Mais a frente vamos detalhar o *modus operandi* dos *hipervisores*.

Neste estudo abordaremos principalmente a categoria de virtualização a nível de *hardware*, a forma mais comum de virtualização. O objetivo principal dessa categoria é um ambiente onde os servidores que estavam operando abaixo da capacidade máxima agora possam ser consolidados em uma única grande máquina operando com carga plena, reduzindo assim o espaço utilizado, facilitando o gerenciamento e economizando energia e refrigeração.

Em termos simples, a virtualização funciona como uma camada de abstração entre os recursos físicos (*hardware*) do servidor (*host*) e o *software*. Essa camada impede o acesso direto do *software* ao *hardware* garantindo que as aplicações e máquinas virtuais sejam isoladas, como mostra a figura 2.6. A forma como a camada de abstração é implementada dá origem às máquinas virtuais de processo e controle, assim como os monitores de máquinas virtuais (*hypervisors*).

Figura 2.6: Exemplificação da Virtualização

2.3.1 Características dos Hipervisores

Os hipervisores possuem classificações que ditam basicamente a sua arquitetura e seu modo de operação. O hipervisor serve como uma interface entre a máquina virtual e os recursos físicos da máquina real. A forma como a arquitetura do hipervisor é implementada impacta em quão efetivo é o gerenciamento da virtualização, já que, o hipervisor é uma ponte entre os recursos de *hardware* e a máquinas virtuais.

Os *hipervisores* podem ser classificados em dois tipos, como mostra a figura 2.7 (Veras & Carissimi, 2019):

- **Tipo 1** (*bare metal*, nativo ou supervisor): Os de tipo 1 são aqueles que rodam diretamente no *hardware* do servidor para fazer controle de acesso dos recursos para o sistema operacional convidado. O hipervisores tipo 1 não possuem um sistema operacional para controlá-lo, pois ele é responsável pelo gerenciamento. O papel

do hipervisor nativo é compartilhar os recursos de *hardware* entre as máquinas virtuais, de forma que cada uma delas imagina ter recursos exclusivos. Exemplos são: *VMware ESX Server, Microsoft Hyper-V e Xen Server*. Uma variação do tipo 1 é o "*embedded hypervisor*"que é instalado no *firmware* (como o *VMwareESXi*). Esse hipervisor é pequeno e tem um impacto mínimo nos recursos e no desempenho do servidor físico.

- **Tipo 2** (*hosted*): Os *hipervisores* tipo 2 são aplicações que fornecem um ambiente de execução para outras aplicações. Executa sob um sistema operacional nativo como se fosse um processo deste. A camada de virtualização é composta por um sistema operacional hóspede e um *hardware* virtual, que são criados sobre os recursos de *hardware*, oferecidos por meio do sistema operacional nativo. Exemplos são: *VMware plaer, Virtualbox e Virtual PC*.

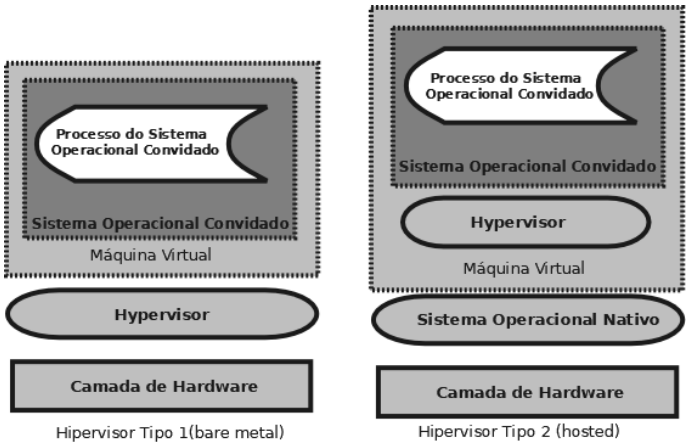


Figura 2.7: Tipos de hipervisores

Os *hypervisors* tipo 1 são a forma uma virtualização mais vantajosa, segura e eficiente pelo fato de ter acesso direto aos recursos de *hardware*, sem que seja necessário um sistema operacional para efetuar seu controle. (Obasuyi & Sari, 2015)

2.3.2 Arquitetura de virtualizadores

Os virtualizadores também podem ser classificados por sua arquitetura. As diferentes arquiteturas de hipervisores são:

- **Completamente monolítica:** Uma arquitetura completamente monolítica é uma única camada de *software* responsável pela emulação no servidor, virtualização da Unidade Central de Processamento (CPU) e emulação de periféricos. Exemplos: *Xvisor* e *VMware ESXi Server*. Na figura 2.8 é apresentada esta arquitetura.

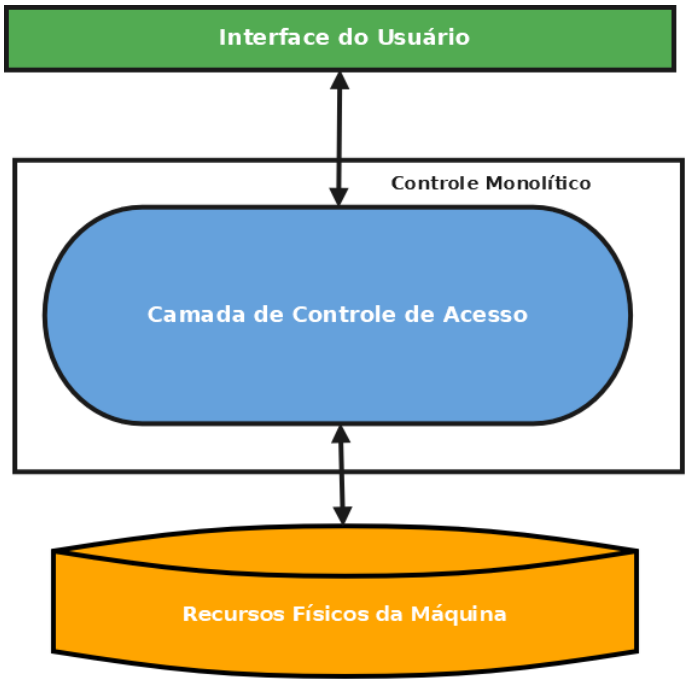


Figura 2.8: Arquitetura completamente monolítica

- **Parcialmente monolítica:** A arquitetura parcialmente monolítica, do tipo *Kernel-based Virtual Machine* (KVM), por exemplo, é uma extensão da versão completamente monolítica. Eles suportam acesso de *hardware* do *host*, virtualização de CPU no *kernel* do sistema operacional e controle de emulação de periféricos na interface de usuário, vide figura 2.9.

- **Micro-*kernel*:** Hipervisores com arquitetura micro-*kernel* têm formas mais leves de virtualizadores que oferecem acesso básico do *hardware* do servidor e virtualização da CPU para o *hipervisor*. Para o resto dos recursos necessários para as máquinas virtuais, o *hipervisor* utiliza uma máquina virtual de gerenciamento com acesso total aos recursos do servidor. Por exemplo, a máquina virtual de gerência Dom0 no *Xenserver*, na figura 2.10, gerencia os acessos das máquinas virtuais aos recursos físicos.

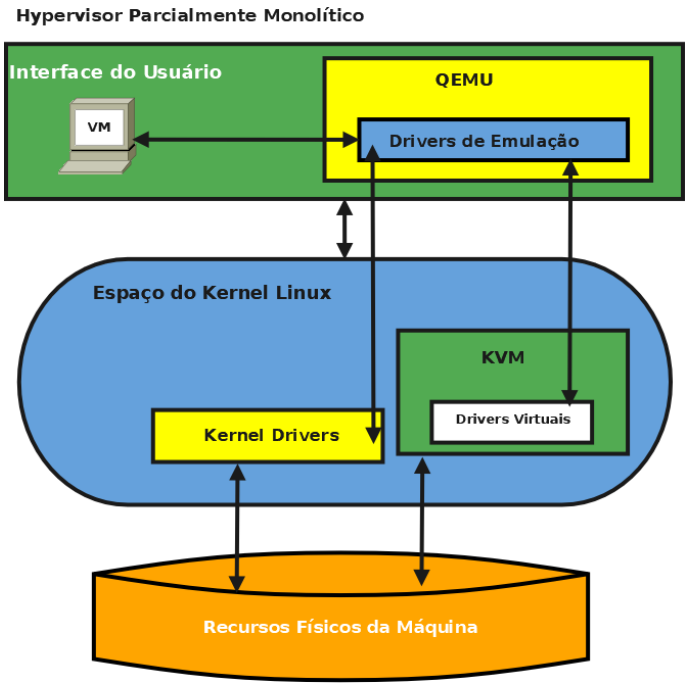


Figura 2.9: Arquitetura Parcialmente Monolítica

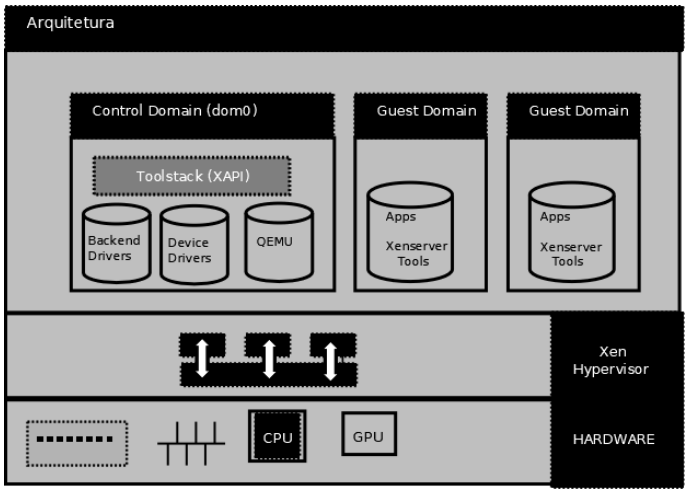


Figura 2.10: Arquitetura *Micro-Kernel*

Nos softwares de virtualização envolvendo *hipervisores*, as máquinas virtuais podem ser implementadas de diferentes maneiras alterando os níveis de privilégio dentro do sistema operacional nativo, cada uma possuindo vantagens e desvantagens. Esses diferentes modos de operação da virtualização são (Veras & Carissimi, 2019):

- **Virtualização completa (VC)**: A virtualização completa é uma forma de virtualização que realiza total abstração do sistema físico de um computador. Não é necessário fazer qualquer tipo de modificação no sistema operacional convidado ou em suas aplicações. Esse método facilita migração das máquinas virtuais, já que

há total independência entre os recursos físicos e a máquina virtual, como mostra figura 2.11.

- **Paravirtualização (PV)**: Os virtualizadores no servidor hospedeiro são formas modificadas de sistemas operacionais que utilizam *hyper*-chamadas para ter acesso aos recursos do servidor. E os sistemas operacionais das máquinas virtuais têm conhecimento de que são virtualizados e que estão compartilhando recursos, como mostra a figura 2.12. Neste modo de virtualização, os sistemas convidados devem ser versões modificadas de seus sistemas operacionais para que funcionem corretamente neste tipo de virtualização.

- ***Hardware Assisted Virtualization*(HAV)**: A virtualização assistida por *hardware* foi uma maneira de tentar solucionar as diferenças de desempenho entre a virtualização completa e paravirtualização, utilizando extensões da arquitetura x86 para suportar virtualização e melhorar o desempenho de ambas soluções como um todo. Basicamente, os processadores destinados a virtualização (Intel VT e AMD-V) alteram o funcionamento de privilégios do processador para que os hipervisores passem a ter acesso total e prioridade sobre o sistema operacional convidado. Assim, as instruções privilegiadas e sensíveis executadas pelo sistema operacional convidado causam um desvio para o *hipervisor*, que agora tem a responsabilidade de tratar adequadamente a ocorrência dessas ações, como mostra figura 2.13

- ***Linux Containers***: Os *containers* compartilham o mesmo *kernel* do sistema operacional e isolam os processos da aplicação do restante do sistema. Os *containers Linux* são extremamente portáteis, mas devem ser compatíveis com o sistema operacional subjacente. Os *containers Linux* são executados de maneira nativa no sistema operacional, compartilhando-o com todos os outros *containers*. Assim, as aplicações e os serviços permanecem leves e são executados em paralelo com agilidade.

2.4 Sistemas operacionais convidados

As máquinas virtuais rodam dentro de servidores dedicados de alto desempenho. Seus sistemas operacionais funcionam em um ambiente dentro dos servidores virtualizadores, como por exemplo no *Proxmox*, *VMware* e *Xenserver*, entre outros. O sistema operacional da máquina convidada utiliza os recursos físicos do servidor que são alocados e controlados dinamicamente pelo *hypervisor* (Padhy, Patra, & Satapathy, 2010). Os sistemas operacionais convidados das máquinas virtuais mais utilizados são o CentOS, Ubuntu, Fedora, BSD e versões *Microsoft Windows* 7, 8 e 10.

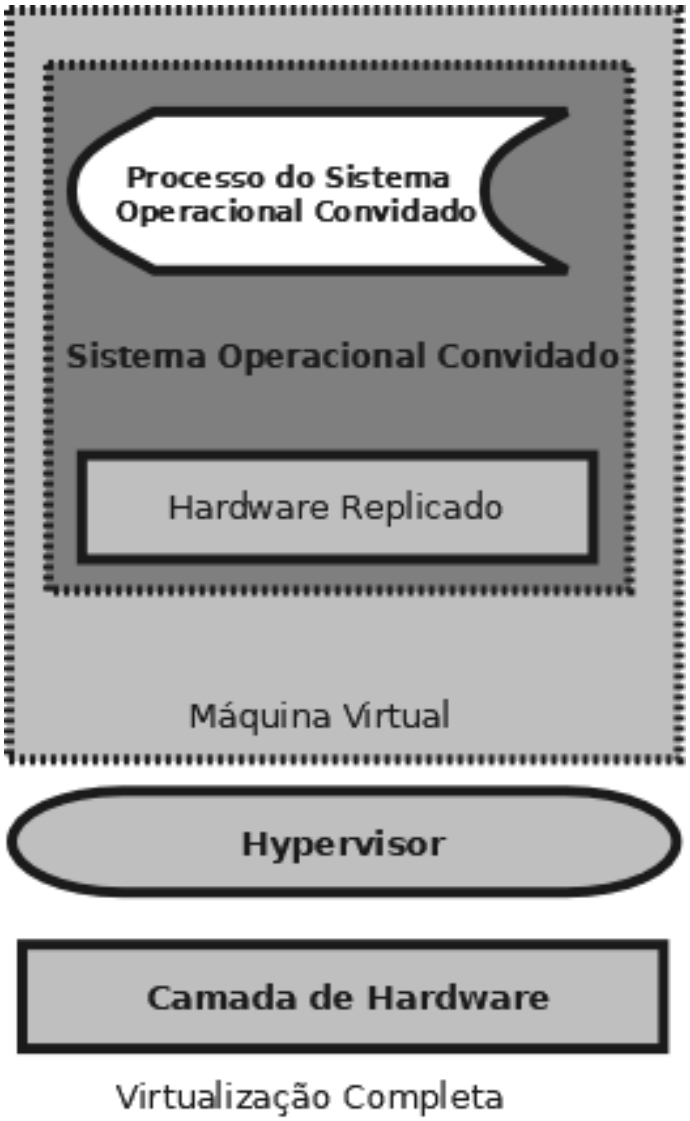


Figura 2.11: Virtualização Completa

2.5 Principais soluções dos virtualizadores

As principais soluções de virtualizadores utilizadas no mercado são o *Microsoft Windows Server Hyper-V, Citrix XenServer, Red Hat Enterprise Virtualization e VMware VSphere*. As licenças utilizadas por esses são pagas e geralmente estes valores são bem altos, pois oferecem uma gama de funcionalidades, além de estruturas de suporte 24 horas por 365 dias no ano. Mas nem todas as organizações conseguem recursos para aquisição destas licenças, como por exemplo as organizações públicas, dentre as quais as universidades brasileiras, que dependem de verba governamental e, muitas vezes, a compra de licenças não são permitidas. Nesse sentido, muitos fabricantes de *softwares* de virtualização optaram por licenças livres, onde o básico do *software* é disponibilizado, com poucas funcionalidades e sem contar com suporte do fabricante. Pode-se destacar, nesse

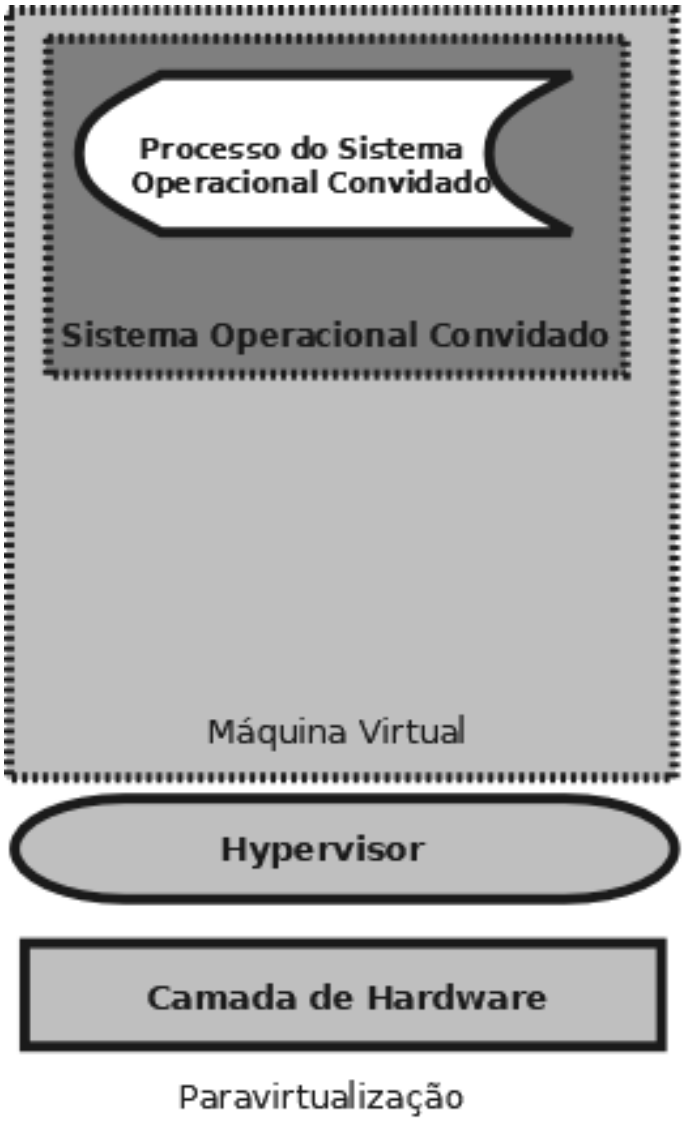


Figura 2.12: Paravirtualização

contexto, o *software* de virtualização o XCP-ng, baseado no *Xenserver*, e o *Proxmox VE*, como sendo as duas únicas opções de fato até o momento da realização deste trabalho. Portanto, neste trabalho serão comparadas as funcionalidades entre essas duas soluções livres disponíveis no mercado: o *Proxmox VE* e o XCP-ng. Abaixo serão descritas as principais funcionalidades de ambas as versões.

2.5.1 Proxmox VE

O *Proxmox Virtual Environment* é uma solução de virtualização de servidores *open source*, baseada no sistema operacional Linux *Debian GNU/Linux* com *kernel Red Hat*, oferecendo virtualização completa e *Linux Container* (LXC) para gerenciamento de *con-*

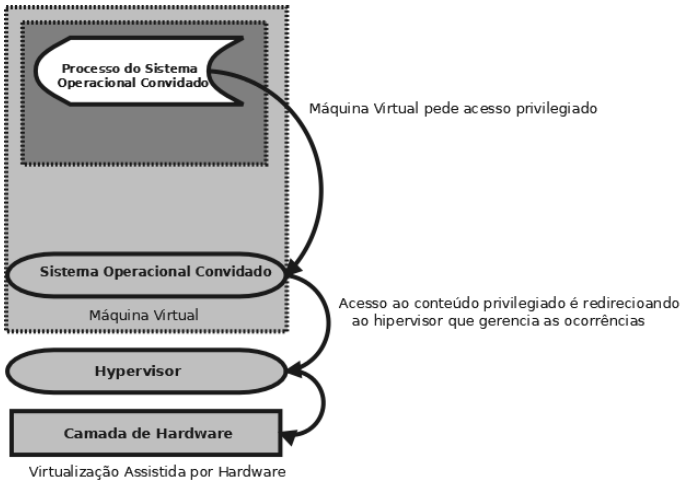


Figura 2.13: Virtualização por Hardware

tainers [1]. Com ele é possível gerenciar máquinas virtuais, *containers*, *clusters* com alta disponibilidade, armazenamentos e configurações de rede através de uma interface *web* ou *Command-Line Interface* (CLI). O código do Proxmox VE é licenciado através da GNU *Affero General Public License*, versão 3. O projeto é desenvolvido e mantido pelo grupo *Proxmox Server Solutions* GmbH. O Proxmox também integra ferramentas para configurar alta disponibilidade, recuperação de desastre, armazenamento por *softwares* e as configurações de rede. A forma de virtualização *Kernel-based Virtual Machine* (KVM) é a técnica predominante de virtualização Linux, capaz de fornecer virtualização completa para processadores x86. KVM é um módulo fundido com a linha principal de *kernel* [2] Linux que permite uma performance próxima da versão nativa em todas as versões de hardware x86 e suporta versões de virtualização Intel VT-x ou AMD-V (KVM contributors, 2016). LXC (Linux *Containers*) é uma forma super leve de virtualização se comparada a uma máquina virtual completa e de baixo custo. LXC é uma virtualização que atua a nível de sistema operacional para rodar múltiplos sistemas Linux isolados em um único *host* Linux (Proxmox Server Solutions GmbH, 2019).

2.5.2 XCP-*new generation* (XCP-ng)

XCP-ng é um *software* de virtualização *open source* baseado no *Citrix Hypervisor do Xenserver* que foi criado após a empresa *Citrix* remover diversas funcionalidades da versão grátis do *Xenserver*. A partir disso, alguns indivíduos e empresas se reuniram para criar um *fork* do projeto chamado *XCP-ng*, baseado na documentação livre da versão *Xenserver* mais atual, mas sem restrições nas funcionalidades e aberta à comunidade para

[1] *Containers* são máquias virtuais que utilizam versões mais simples do sistema operacional do hipervisor

[2] Palavra que designa o núcleo do sistema operacional

desenvolver e melhorar. *XCP-ng* é um software *turnkey*, caracterizado por ser capaz de ser instalado basicamente com a ISO e sem necessidade de configurações especiais, além de ser possível gerenciá-lo completamente *online* através da aplicação *web Xen Orchestra* (compatível com versão paga da *Citrix* e do *XCP-ng*) ou de uma aplicação *windows* chamada *XCP-Center*. A história do *Xen* começou em 2003, quando o *Xen* foi lançado como o primeiro hipervisor *open source*. Em 2007, a empresa responsável pelo *Xen* foi adquirida pela *Citrix* e esse foi renomeado para *Xenserver*, tornando-se uma aplicação fechada (*closed source*). Em 2010, a *Citrix* começou a focar em virtualização de *desktop* ao invés da virtualização de servidores com *XenApp/XenDesktop*. Em 2011, foi lançado a primeira versão do *Xen Cloud Project* (XCP) 1.0, uma versão gratuita e *open source* do *Xenserver*. Em 2013, a ferramenta *Xenserver*, antes *closed source*, se tornou agora *open source* e marcou o fim do XCP e o nascimento da aplicação de gerenciamento *web Xen Orchestra*. Entretanto, em dezembro de 2017, diversas funcionalidades foram removidas da versão grátis do *Xenserver*, chamada (*Xenserver Free Edition*), e todas essas e as novas funcionalidades seriam agora da versão paga (*closed source*). Por conta desses fatos, um grupo de pessoas e empresas se uniu e criou um novo projeto de *fork* do *Xenserver* com todas as restrições removidas, sendo chamado de XCP-*new generation* (*XCP-ng*). Das funcionalidades removidas do *Xenserver free* que o *XCP-ng* possui, as mais importantes são o aumento do número de *clusters* possíveis de 3 para 16, controle de memória dinâmica, possibilidade de fazer *live-patching* sem desligar os servidores para *upgrade*, alta disponibilidade, *storage live migration* e balanceamento de carga dinâmico.

3. Metodologia

3.1 Visão geral

Segundo (Rampazzo, 2005) a pesquisa é uma investigação para coletar informações e obter conhecimento novo em relação a algum fenômeno, relacionando este com algo que já se conhece. Existem diversos métodos científicos, o que está mais presente nas observações humanas e na tentativa de compreender algum fenômeno é a **comparação**. A observação de um fenômeno só pode ser reconhecido e analisado como diferente se for comparado com outro fenômeno (des Ligneris, 2016). Esse projeto tem por objetivo realizar uma pesquisa aplicada e comparativa, uma vez que, utilizará conhecimento de pesquisa básica para então analisar o comportamento entre dois *softwares*.

A comparação de características comuns entre objetos permite a categorização e a diferenciação entre os mesmos. Uma análise comparativa é fundamental em diversos estudos. Estas são frequentemente utilizadas nas fases iniciais de muitas pesquisas. Ao se tentar estudar aquilo que não se conhece ou não se compreende é mais fácil buscar aspectos comuns entre o que se conhece, para então, poder formar algum tipo de entendimento. Neste sentido serão identificados dois *softwares* populares e com versões livres presentes no mercado, capazes de oferecer as funcionalidades comuns necessárias para um ambiente de virtualização. O objetivo será identificar, através da comparação, qual seria o *software* mais adequado para se utilizar dentro de um *datacenter* de uma universidade pública brasileira.

Com esta finalidade em mente, será analisada cada funcionalidade julgada como obrigatória em um ambiente de virtualização e como cada um dos *softwares* executa essas tarefas. Durante as comparações também serão enumerados as diferenças entre os *softwares*, identificando vantagens e desvantagens dos mesmos em um ambiente semelhante ao real. Ao final, deseja-se identificar entre os *softwares*, qual seria a melhor escolha para se instalar dentro do ambiente acadêmico, substituindo a solução usada atualmente.

3.2 Categoria do Estudo

Para alcançar os objetivos deste trabalho, usaremos uma metodologia do tipo exploratória, uma vez que, não temos muitas informações sobre o objeto a ser estudado e não sabemos exatamente o que se comparar. A pesquisa exploratória é caracterizada por estudos que buscam uma aproximação de um fenômeno para então melhorar o entendimento deste. O ponto de partida deste tipo de pesquisa é uma revisão bibliográfica, crucial para o entendimento dos conceitos, técnicas e ferramentas da virtualização, assim como, para embasar na descoberta de métricas e medidas utilizadas na comparação. Assim, a leitura de artigos científicos irá ajudar a formular uma melhor compreensão sobre o assunto e que tipo de comparações poderão ser usadas na instalação, desempenho, funcionalidades, suporte, entre outros.

Este estudo comporta-se de forma semelhante a um estudo de caso, pois, as comparações serão feitas mediante a um ambiente controlado, similar ao ambiente virtualizado encontrado na universidade. Como já dito o procedimento inicial será uma revisão bibliográfica, já que, existe um amplo conhecimento estabelecido e publicado sobre as práticas dos ambientes de virtualização. Serão utilizadas pesquisas por artigos que envolvam tecnologias sobre virtualização e que citem tendências e práticas do mercado. O resultado do estudo irá se assemelhar a um guia técnico, o que torna o trabalho auditável e aplicável por estudiosos e profissionais da área.

Quanto ao tratamento e coleta de dados do estudo, esta será qualitativa, pois é obtida de uma fonte direta, com interpretação dos fenômenos e atribuição dos significados analisados. O estudo qualitativo é formulado por perguntas abertas sujeitas a interpretação do pesquisador e no nosso caso, trata-se da capacidade do *software* em realizar certas tarefas determinadas. Quanto a natureza deste trabalho, este possui um caráter hipotético dedutivo, já que faremos um teste aplicado, analisando o funcionamento dos *softwares* sob uma perspectiva de tentativa e erro, tendo como base e objetivo final, a resolução dos principais problemas encontrados no ambiente real.

A partir de todo o conhecimento obtido desde a revisão, testes e deduções, será possível identificar nos *softwares* quais possuem as tecnologias e funcionalidades necessárias para utilizarmos em nosso ambiente simulado semelhante ao da universidade. Uma vez escolhidos os *softwares*, serão executadas as instalações de forma separada, em seguida as configurações e aplicações das funcionalidades mais importantes para o ambiente de virtualização identificadas na revisão bibliográfica.

3.3 Execução do estudo

Com intenção de identificar qual a melhor solução de virtualização para o *datacenter* da universidade, será feita uma comparação entre os dois *softwares*: o ***Proxmox*** e o ***XCP-ng***. Essa comparação será realizada através de dados obtidos através de testes e de dados secundários fornecidos por outros estudos, uma vez que, já existem muitas análises de cada software na literatura científica. A comparação será executada em ambiente semelhante ao encontrado no *datacenter* da universidade, onde cada software será instalado. Serão observados o funcionamento, os recursos necessários para seu funcionamento, as funcionalidades, particularidades de cada *software* e como se aplicam no ambiente que se deseja implantar. Os testes e comparações serão executados com base na tabela 3.1.

A comparação será iniciada pela instalação de cada um dos virtualizadores escolhidos, em seguida iniciam-se a coleta de dados relativas a instalação e as particularidades de cada *software*. Após as instalações, iniciam-se as configurações do *cluster*, este deverá ser configurado através de interfaces gráficas disponíveis em cada virtualizador. Após a criação dos *clusters*, será configurado a conexão com os *storages*. Esta fase é muito importante pois, segundo a literatura, um ambiente profissional de virtualização deve conter no mínimo um *cluster*(dois ou mais servidores) e um *storage*. Para simplificar a conexão com o *storage* e ainda reduzir custos, optou-se por conexão via protocolo *iSCSI*. Ao fim desta etapa, o ambiente de teste (semelhante ao *datacenter* da universidade) estará montado e pronto para as diversas comparações executadas neste trabalho.

A nova etapa será relativa as instalações das máquinas virtuais (máquinas convidadas) que o ambiente virtualizado é compatível. Nesta etapa podem ser realizadas instalação de diferentes tipos de máquinas virtuais e com sistemas operacionais diferentes, assim estabelecendo um comparativo entre os virtualizadores e sua compatibilidade com sistemas operacionais do mercado. Também serão executados testes e configurações de *Virtual Local Area Networks* (*VLANs*), tanto entre máquinas virtuais quanto na integração com *VLANs* de *switches* físicos do ambiente simulado. Também nesta fase testes de clonagem e edição das configurações das máquinas virtuais serão testadas.

Em seguida serão executados testes mais avançados de redes como por exemplo: alta disponibilidade (*High Availability* ou HA), *backup*, agregação de rede e balanceamento de carga entre outros. A configuração de HA é suma importância para um ambiente de virtualização, neste teste serão configurados o HA e verificados as diferenças de configuração em ambos os virtualizadores. Após esta configuração poderão ser simulados desastres (desligamento de um dos virtualizadores) e como acorrem as migrações das máquinas

virtuais para o servidor disponível. Também testes de recuperação serão executados com a criação de processos de *backup* e *restore* em cada virtualizador.

Por fim, outros testes avançados de rede serão configurados, como por exemplo, agregação de tráfego de rede (*Network Bonding*) com o *switch* físico do ambiente simulado. Este teste será configurado via interface gráfica e também via linha de comando. O teste tem a intenção de verificar se cada virtualizador é capaz de agregar placas de redes aumentando sua capacidade de tráfego, assim como, fazer a medição do tempo de migração das máquinas virtuais. Também serão testados procedimentos de migração à quente, ou seja com máquina virtual ligada (*Live Migration*). Este tipo de migração é muito importante na recuperação em desastres, pois, quando um virtualizador falhar, suas máquinas virtuais migram em tempo real para os virtualizadores ativos, diminuindo o máximo o tempo de parada das máquinas virtuais.

Os testes e comparações serão executados com base na tabela 3.1, utilizando um ambiente controlado semelhante ao ambiente encontrado na universidade. Serão executadas todos os testes da tabela, como também serão simuladas as situações de emergência para que se observe comportamento de cada software em situações de recuperação de desastre presente na tecnologia de virtualização. Após todas as instalações, configurações e testes, os ambientes serão comparados para que, então, tenhamos uma base das diferenças de cada software. A intenção é ter uma base para pode se escolher qual a melhor opção para a Universidade e qual melhor atenderá os requisitos de um ambiente virtualizado presentes no ambiente real. Existem muitos fatores que podem justificar a escolha de um em detrimento do outro como: desempenho, formato dos arquivos, tamanho dos arquivos assim como, qual está mais apto a receber a migração de ambiente caso, a partir do estudo, seja feita a mudança.

Funcionalidades para comparação
Complexidade da instalação
Atribuição do *storage* compartilhado
Configuração de *Cluster*
Migração de máquina virtual
Métodos de Virtualização
Arquiteturas do servidor
Sistema Operacional Hóspede/Convidado
Quantidade de Recursos por Hipervisor
Gerenciamento de Rede
Diferenças na criação de *Snapshot*
Complexidade da rotina *Backup*/*Restoration*
Monitoramento de Desempenho
Clonagem e *Templates*
Formatos de importação/exportação
Alta disponibilidade
Live Migration
Balanceamento de carga
Network Bonding

Tabela 3.1: Tabela de funcionalidades para comparação

4. Comparação Experimental

4.1 Visão Geral

Nesta seção será executada as comparações entre os dois *softwares* escolhidos, *Proxmox* e *XCP-ng*. As comparações foram feitas em um ambiente controlado, semelhante ao de uma universidade pública brasileira. Neste ambiente foram instalados os dois hipervisores e executados todos os testes para então serem comparados. Os testes e comparações foram baseados na tabela 3.1.

4.2 Topologia

Para realizar a comparação, foi montada uma estrutura composta de dois servidores físicos com duas placas de rede, um servidor adicional necessário para testar alta disponibilidade em uma das soluções e uma estação de trabalho usado para gerência. Foi criado uma rede própria com os servidores e um *storage* físico de pequeno porte interligados por *switch* físico de camada três. O ambiente foi ligado a uma rede externa com várias *vlans* já configuradas, o objetivo é interligar as estações virtuais criadas com *vlans* externas de outras redes.

A topologia foi montada e inspirada no ambiente existente dentro da universidade, esta possui uma solução de virtualização proprietária com diversas funcionalidades conhecidas do mercado de trabalho e a ideia é analisar como seria a migração desse ambiente para um ambiente utilizando os dois *softwares* livres que estão sendo aqui comparados. O ambiente montado é constituído de dois servidores *DELL PowerEdge R300* com processadores *Single Intel Xeon* com 8 GB de memória RAM, ambos com duas placas de redes ligados a uma rede externa com várias *vlans* configuradas. Também no ambiente existe um *storage* ligado ao *switch* físico que será apresentado ao ambiente de virtualização

através do protocolo ISCSI [1]. Na figura 4.1 é apresentado a topologia montada.

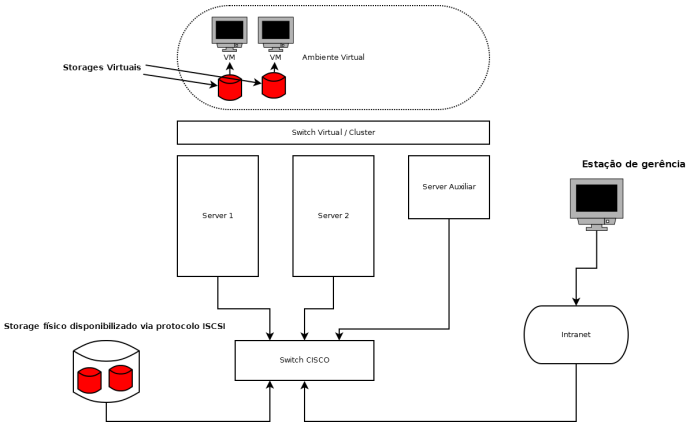


Figura 4.1: Topologia do ambiente

4.3 Testes a serem executados

Os principais testes que serão executados no ambiente são: instalação dos *softwares* (virtualizadores) no ambiente controlado, criação de um *storage* compartilhado para ser apresentado ao ambiente virtual, configuração de alta disponibilidade, migração de máquinas virtuais, importação e exportação de máquinas virtuais através das ferramentas dos *softwares*, gerenciamento do ambiente virtual via interface gráfica e criação de rotina de *backup/restore* automáticos. Também serão executados testes mais complexos como, balanceamento de carga, *NIC bonding*, *live migration* e *thin provising* assim como uma análise geral das principais características de cada *software*. Os testes e comparações estão baseadas na tabela 3.1, cada item será testado com os dois *softwares*, para então serem comparados. Os *softwares* serão instalados separadamente para que se possa conhecer as particularidades de cada um. Cada *software* será instalado mais de uma vez. A ideia é entender o funcionamento e aplicação primeiro, para então, serem feitas as medições e comparações na segunda instalação.

4.4 Instalações dos *softwares*

4.4.1 Instalação *XCP-ng*

A primeira instalação foi do *software XCP-ng*, as configurações durante a instalação são simples e intuitivas. Vale citar na instalação: a configuração do disco local (ou tam-

[1] Sigla de *Internet Small Computer System Interface*. O ISCSI é protocolo de transporte que transporta comandos SCSI entre um computador anfitrião (*initiator*) e um dispositivo de destino (*target*)

bém *storage* local) é fornecido a opção para habilitar o *thin provision* e nas configurações de rede é possível configurar o número IP do servidor via DHCP ou atribuir manualmente. Esses dois passos são apresentados nas figuras 4.2 e 4.3.

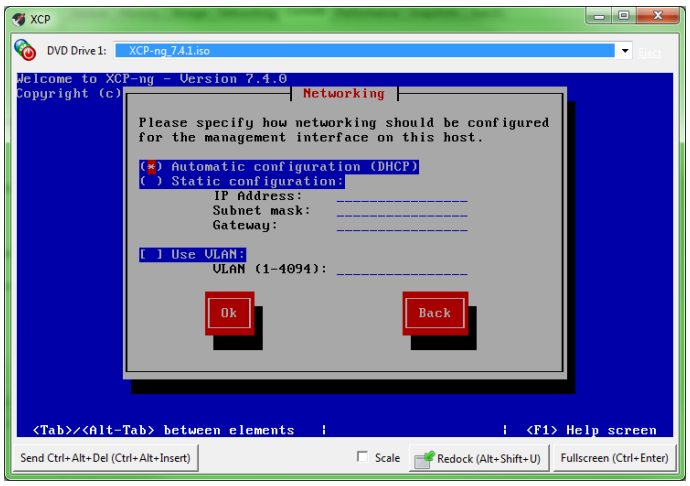


Figura 4.2: Configurar via DHCP

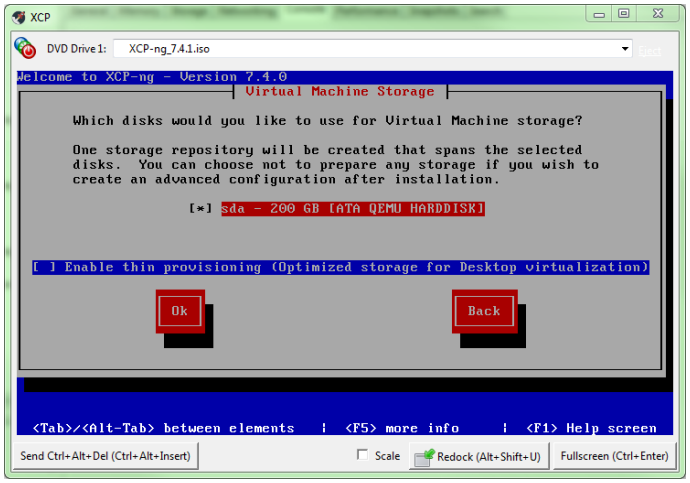


Figura 4.3: Opção de habilitar *Thin Provisioning*

Com a opção *thin provision* habilitada, o sistema fornece espaço do disco local de acordo com a necessidade do *software*, em vez de alocar todo espaço do disco de uma vez só. Este processo torna o uso do disco local mais eficiente pois a quantidade alocada será apenas a requerida pelo sistema, evitando que este seja usado completamente, podendo, a parte não alocada, ser usada para outros fins (Qinlu He, Zhanhuai Li, & Xiao Zhang, 2010).

Após a instalação e *reboot*, é apresentada a tela inicial com as principais configurações e, ao lado, um menu com outros recursos do sistema. A partir desta etapa, todo gerenciamento do *software* deve ser feito de forma remota, com a aplicação de gerência instalada em uma estação de trabalho ou via linha de comando executado no próprio

servidor via procolo *SSH*[2]. Para configuração remota utilizamos a aplicação *XCP-Center* em outra máquina, com sistema operacional *MS-Windows* para que seja possível iniciar as configuração do virtualizador. Através do *XCP-Center*, depois de efetuar a instalação nos três servidores, a primeira coisa que precisamos fazer é conectar os servidores ao *XCP-Center* pela rede utilizando seus endereços IP e suas senhas de administração, como mostra figura 4.4. Ao conectarmos todos os três servidores ao *XCP-Center*, o sistema está pronto para iniciar a configuração.

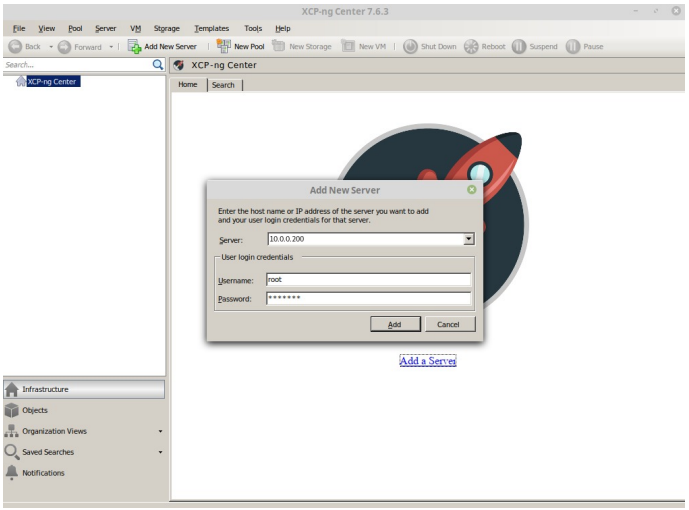


Figura 4.4: Adição de servidor no *XCP-ng*

4.4.2 Instalação do *Proxmox*

A instalação do *Proxmox* possui a maioria das opções de configurações semelhantes ao do *XCP-ng* até o ponto onde se configura os endereços de IP do servidor. Não é possível, durante a instalação, optar por configuração automática via DHCP. É necessário estabelecer manualmente os endereços e por consequência é necessário ter conhecimento sobre a rede em que se realizará a instalação, como mostra figura 4.5.

[2]Protocolo usado para acesso remoto a servidores Linux

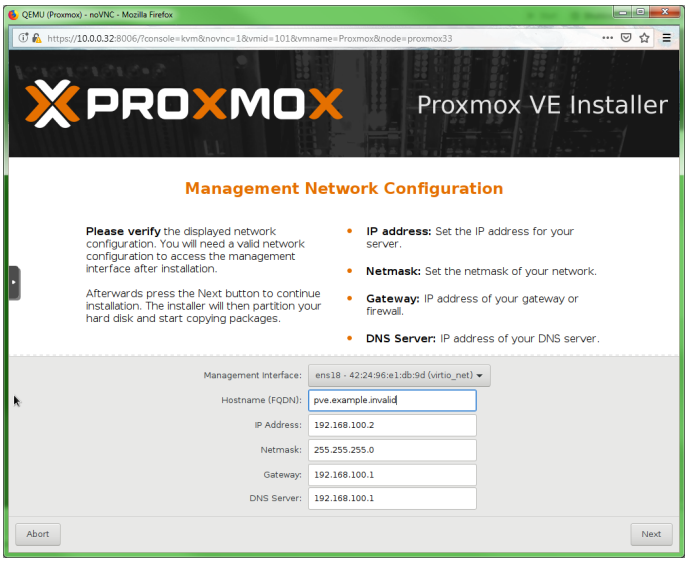


Figura 4.5: Configuração de rede Proxmox

As opções de configuração após configuração de rede são simples e intuitivas. Ao final é feito o *reboot* e a máquina liga com o menu solicitando o login e senha administrativa para iniciar configuração ou acessar informações via linha de comando. Para o correto funcionamento do *Proxmox*, são instalados três servidores. Após a instalação, os três servidores devem ser acessados remotamente via interface *web*.

4.5 Configuração do ambiente de virtualização

Com os virtualizadores instalados e prontos para configuração partimos para a implementação de uma réplica do ambiente real e das funcionalidades necessárias para testarmos o funcionamento do *software* e analisarmos como ele se comportaria se estivesse operando em um ambiente real.

4.5.1 Configuração do XCP-ng

A configuração do *XCP-ng* é executada com o *XCP-ng Center*, onde é possível gerenciar, via interface gráfica, a rede do sistema, criação das *VLAN's* necessárias e todas as outras configurações. Assim como na configuração de rede, a criação de *clusters* e conexão ao *storage* pode ser feita pela interface facilmente, uma vez que todos os servidores sejam adicionados ao *XCP Center*.

Para a instalação de máquinas virtuais, primeiro criamos uma conexão ao *storage* virtual disponível via protocolo ISCSI, este previamente configurado. As figura 4.6 e 4.7

mostram como se faz a união dos servidores conectados ao *XCP-Center* e o ambiente unido em um *cluster* e com o *storage* virtual conectado.

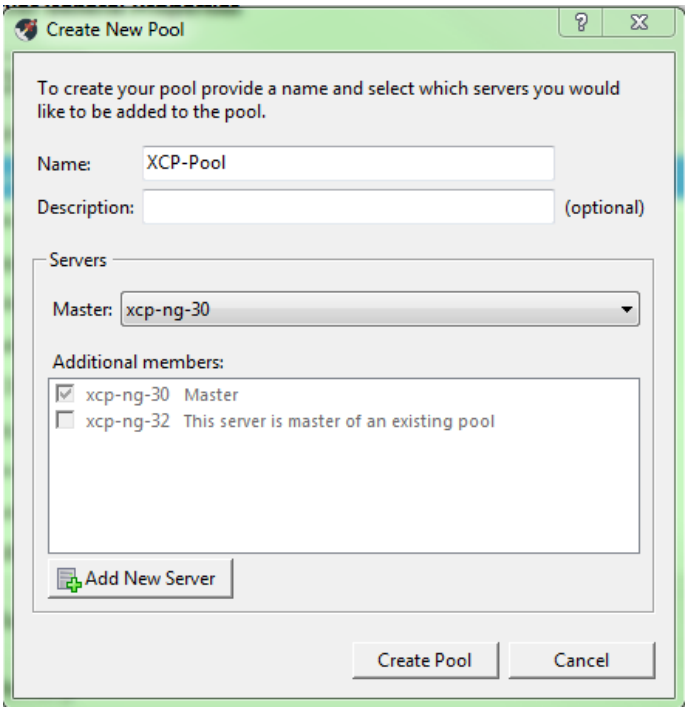


Figura 4.6: Criação de *Cluster* no XCP

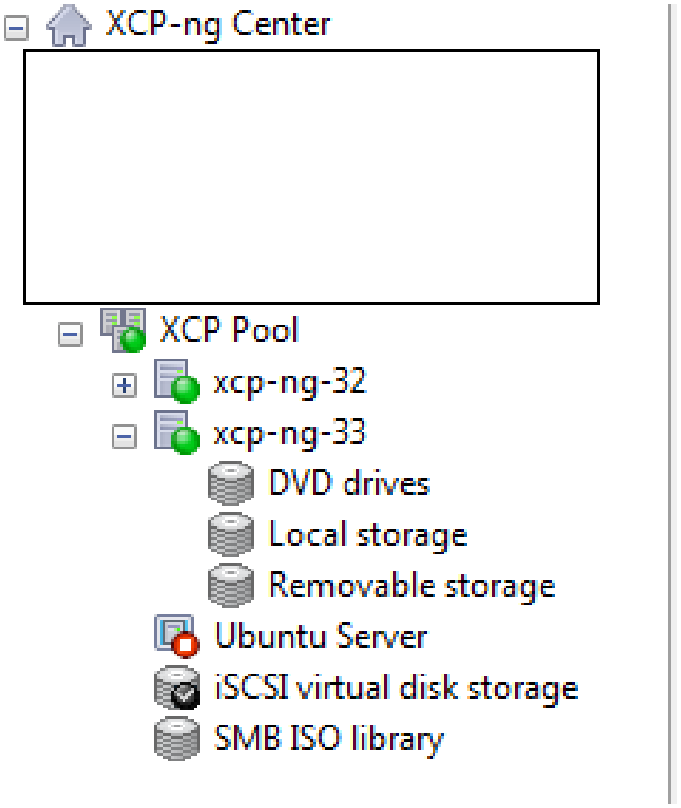


Figura 4.7: Cluster com *storage* virtual compartilhado

Com o *cluster* criado, a rede configurada e *storage* virtual conectado, foi adicionado as primeiras máquinas virtuais para aplicarmos posteriormente a alta disponibilidade e iniciar os testes de migração. No *XCP-ng* é capaz de fornecer virtualização completa, então, é possível instalar máquinas *Windows* e *Linux* como mostra as figuras 4.8 e 4.9. Neste ponto já é possível mover máquinas facilmente entre os servidores físicos através da interface gráfica. Na figura 4.10 é apresentado como ocorre esta movimentação de uma máquina virtual para outro servidor físico.

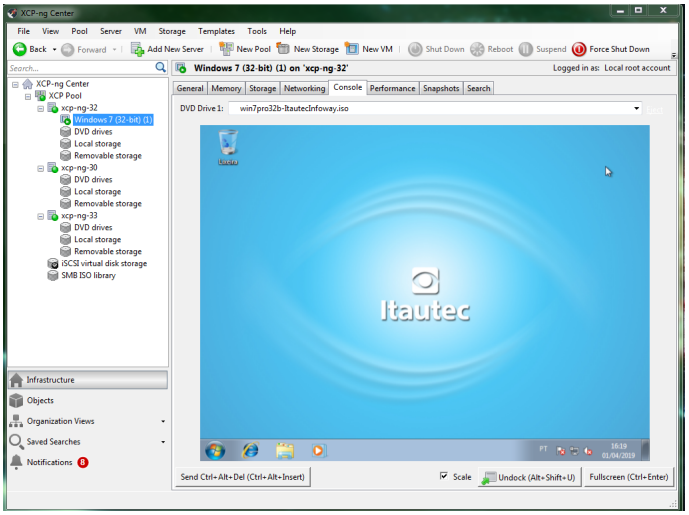


Figura 4.8: XCP com *Windows 7*

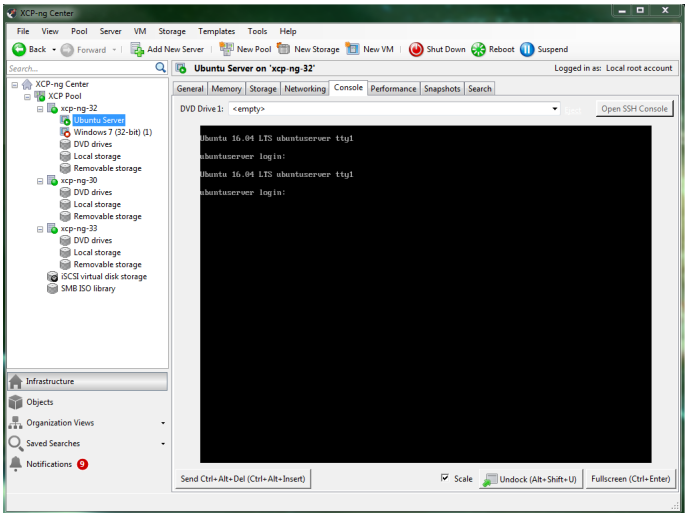


Figura 4.9: XCP com *ubuntuserver*

A figura 4.11 mostra o ambiente instalado e configurado. Esta topologia esta baseada no ambiente utilizado na universidade, a partir daqui, serão executados os testes mais avançados, como por exemplo, a alta disponibilidade. O ambiente ficou configurado com dois servidores físicos, um *storage*, um *switch* e uma estação de gerência.

Com o ambiente em funcionamento e com as funcionalidades aplicadas, iniciamos

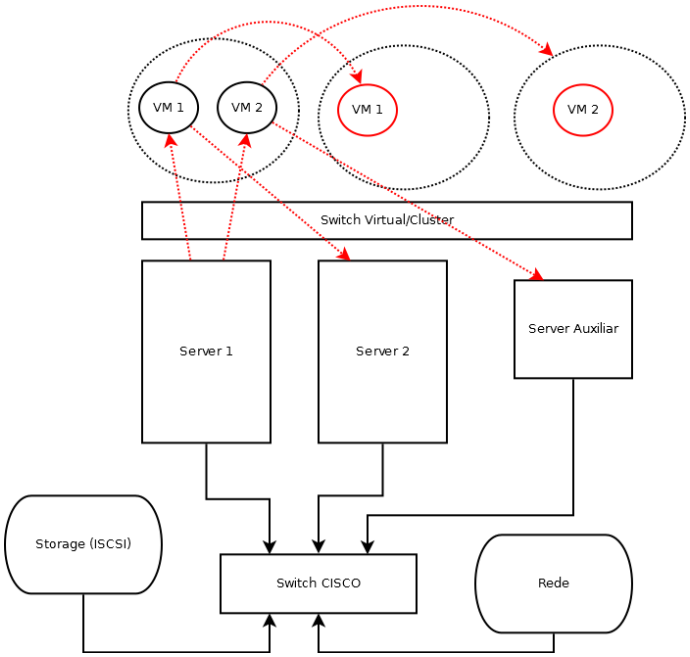


Figura 4.10: Migração no ambiente virtual

alguns testes mais avançados que podem ocorrer no ambiente real e o sistema precisa estar pronto pra lidar com eles. O mais importante é a solução de desastres de servidor. Para isso foi simulado um desastre de energia removendo o cabo de força do servidor e com a alta disponibilidade configurada, como mostra a figura 4.12. Em alguns minutos as máquinas no servidor desligado deverão ser migradas para outro servidor disponível, alterando apenas um ponteiro que direciona a máquina a outro servidor. A figura 4.13 exemplifica esse processo.

4.5.2 Configuração do *Proxmox*

Após a instalação do *Proxmox* nos 3 servidores, é possível controlar as configurações pela interface *web* e acessando com *login* e senha administrativa. As figuras 4.14 e 4.15 mostra as telas de *login* e a página inicial.

A primeira configuração executada é a configuração de rede, apesar de ser possível editar via interface *web*, ao criarmos as VLAN's ou criar agregações de interfaces de rede, perdíamos a conexão com os servidores. Foi necessário então editar as configurações de rede via linha de comando. Diferentemente do *XCP-ng*, na interface gráfica não foi possível criar as VLANs necessárias para os testes de rede. Para isso, foi necessário buscar na documentação como configurar via linha de comando, pois as mudanças na interface não estavam funcionando como esperado. As máquinas virtuais não estavam se conectando apropriadamente à VLAN's. Depois de um longo tempo de estudo e tentativas, consegui-

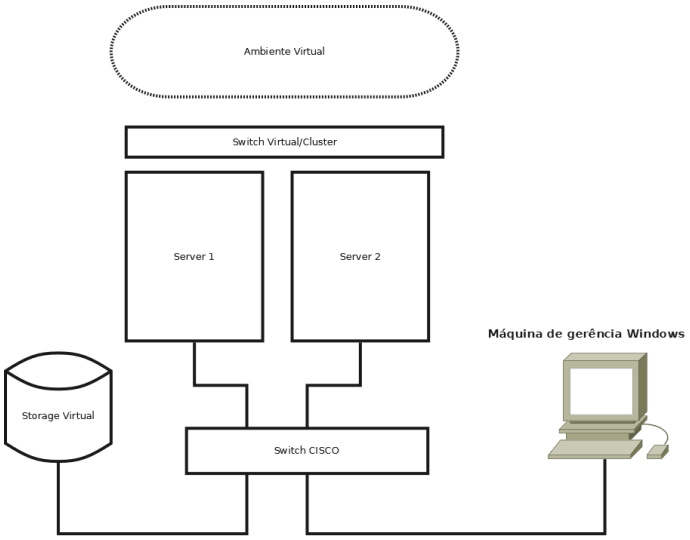


Figura 4.11: Topologia do ambiente XCP

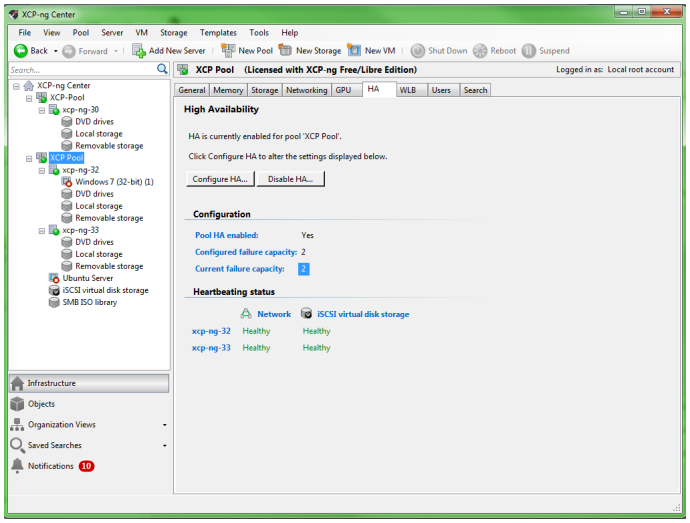


Figura 4.12: *Cluster* com alta disponibilidade

mos alterar a configuração de rede manualmente, para obtermos conexão das máquinas virtuais com as VLAN's. Com as VLAN's funcionando podemos criar as máquinas virtuais nas redes apropriadas. De acordo com a documentação, o software *Proxmox VE* suporta virtualização completa *Kernel-based Virtual Machine* (KVM) e *Containers Linux*. O *container* é uma versão mais simples das máquinas virtuais que usam o mesmo sistema operacional do virtualizador (servidor físico), economizando espaço em disco. O próximo passo é a instalação de máquinas para testar migração e foi testado com uma máquina virtual completa *windows* e um *linux container*. As figuras 4.16 e 4.17 mostram mas máquinas *windows* e uma máquina *linux Ubuntu* respectivamente.

No *Proxmox*, para configuração do *storage* virtual, foi necessário criar uma conexão com a *Logic Unit Number* (LUN) do *storage* e a partir dessa conexão que se pode criar

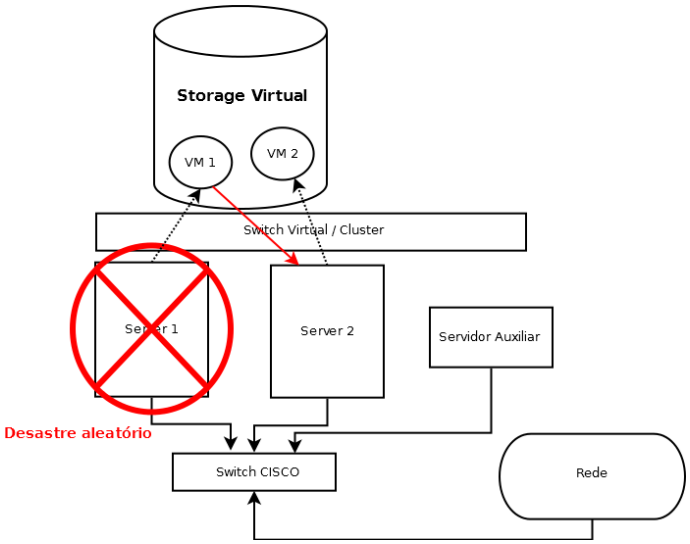


Figura 4.13: Migração após desastre

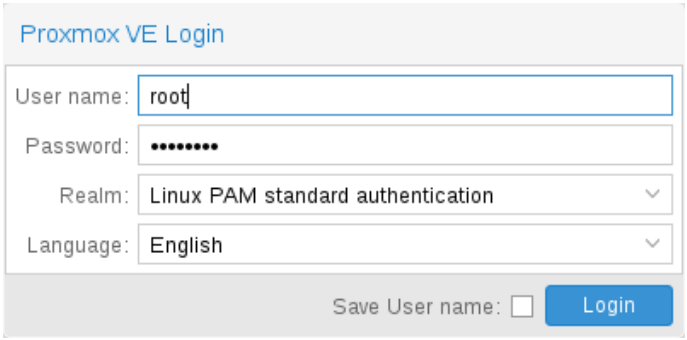


Figura 4.14: Tela de *login*

um volume de disco para ser utilizada pelas máquinas virtuais. Durante a criação de volume é possível optar pela opção de *Thin Provisioning*, como mostra a figura 4.18. Já no *XCP-ng* esta é fornecida durante a instalação no disco local. Com *storage* conectado podemos criar máquinas virtuais e executar as configuração de rede. Apenas após esse procedimento, conectamos os servidores em *cluster* através da rede. Cada servidor possui uma informação de identificação obrigatória para se juntar em um *cluster*, como mostra a figura 4.19. Com isso já se pode fazer a migração de máquinas virtuais entre servidores, no *Proxmox* basta selecionar opções na máquina e clicar para migrar como mostra a figura 4.20. Para alcançarmos o cenário básico ideal basta configurarmos alta disponibilidade e um procedimento de *backup*. A configuração de alta disponibilidade do *Proxmox* é mais elaborada que no *XCP-ng*. É possível criar grupos de servidores num mesmo *cluster* que vai disponibilizar a funcionalidade e qual servidor possui prioridade, como mostra a figura 4.21, e quais máquinas irão usufruir do procedimento, como mostra a figura 4.21. Uma vez configurado, caso um dos servidores do grupo de alta disponibilidade seja desligado, as máquinas selecionadas do procedimento, que forem afetadas, vão migrar para outro servidor disponível, procedimento este testado com sucesso. No *Proxmox* ao

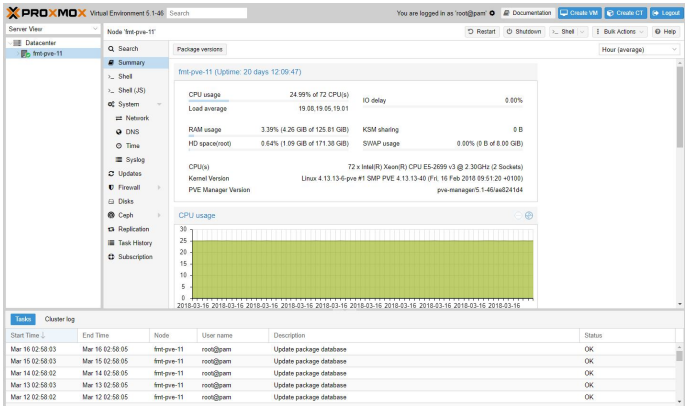


Figura 4.15: Tela inicial *Proxmox*

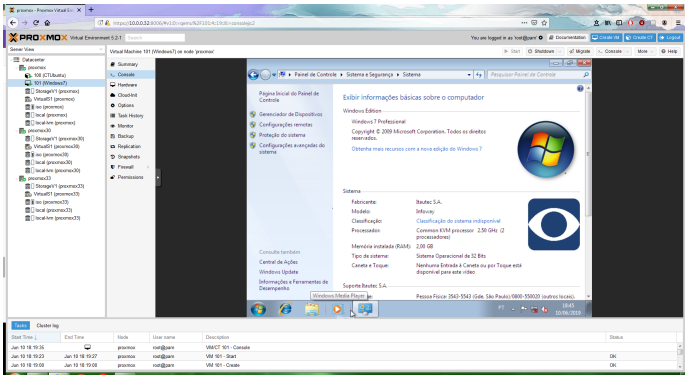


Figura 4.16: Proxmox com *Windows* 7

contrário do *XCP-ng* é necessário um quorum de no mínimo três servidores para que alta disponibilidade seja garantida. Enquanto que no *XCP-ng* pode-se testar com duas máquinas, no *Proxmox* foi necessário utilizar uma terceira máquina apenas para quorum. Para que a migração seja possível, entretanto, é necessário o *storage* compartilhado.

Com alta disponibilidade configurada, a rotina de *Backup/Restore* é facilmente configurada pela interface com opções bem mais simplificadas e automáticas. Basta escolher as máquinas que deseja fazer *backup* e configurar a frequência como mostra a figura 4.23.

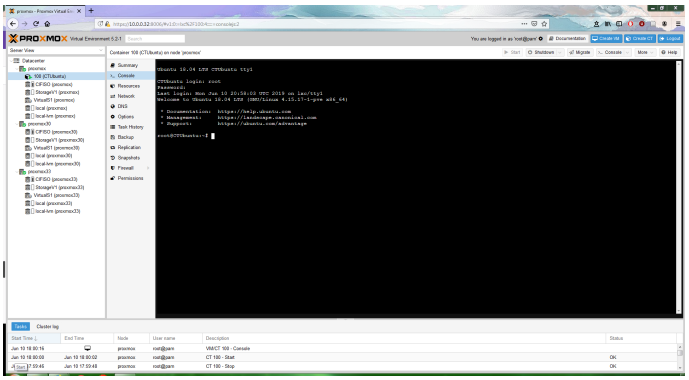


Figura 4.17: Proxmox com *ubuntuserver*

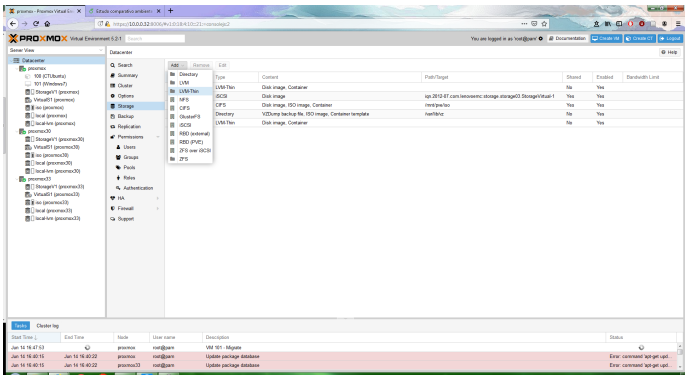


Figura 4.18: A opção *LVM-Thin* configura o disco com a tecnologia *Thin Provision*

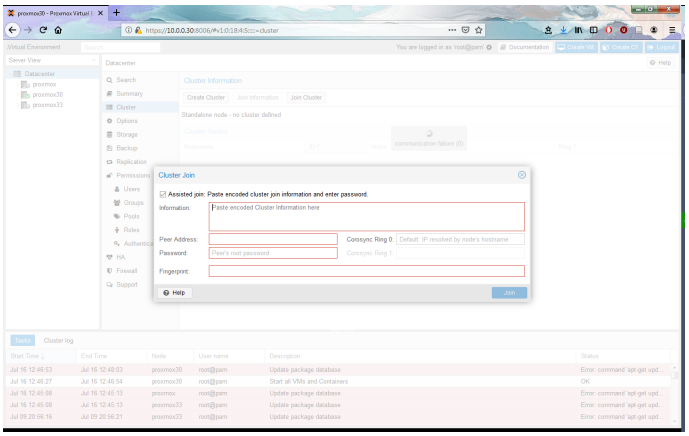


Figura 4.19: Cada *cluster* possui um código para realizar a união

4.6 Análise Comparativa entre os Virtualizadores

Após termos os dois ambientes instalados e configurados para atender os parâmetros estabelecidos e inspirados em um ambiente real, partimos para uma análise de cada software.

4.6.1 Métodos de Virtualização

O *XCP-ng* possui documentação aberta e suas configurações estão baseadas no *Xenserver* 7.6. De acordo com a documentação o *XCP-ng* é capaz de oferecer virtualização completa. Esta é uma forma de emular sistemas operacionais compatíveis onde o hipervisor tem acesso total ao recursos de *hardware* e gerencia o acesso das máquinas virtuais. Essas máquinas virtuais compartilham seus recursos de *hardware* com outras máquinas virtuais, mas sem saber da existência delas, achando que são máquinas "completas"(Ally, 2018). O *XCP-ng* também é capaz de oferecer paravirtualização. Nesta, as máquinas virtuais sabem que estão utilizando recursos compartilhados com outras máquinas por usarem um *kernel* especializado e modificado para o ambiente. Essa forma possui um

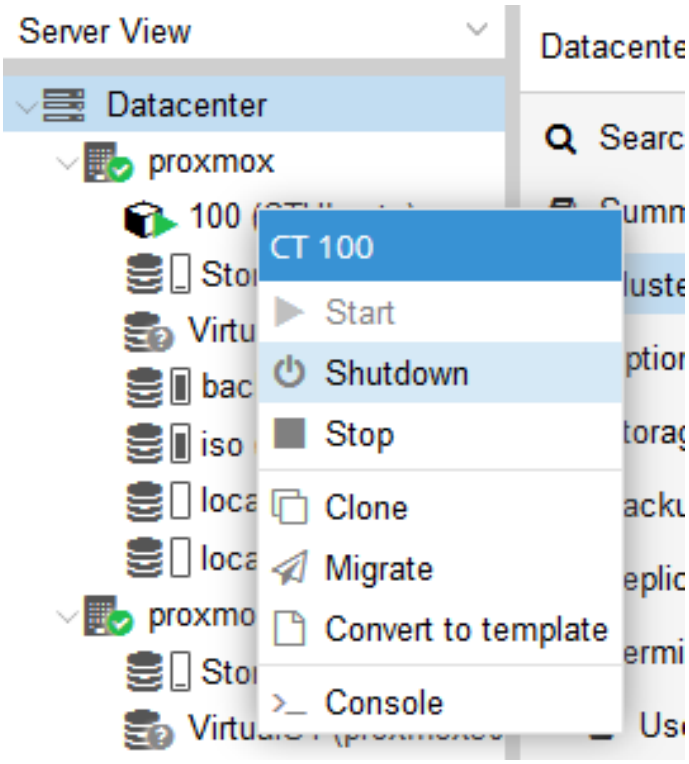


Figura 4.20: Migração no *Proxmox*

desempenho mais econômico pois o sistema utiliza os recursos como uma entidade única (Ally, 2018). O *Proxmox*, além de oferecer virtualização completa e paravirtualização, possui a possibilidade de utilizar *Linux Containers*. *Linux Containers* são formas de virtualização onde a máquina virtual usa o mesmo sistema operacional do virtualizador hospedeiro, economizando espaço, processamento e memória. *Linux Containers* é uma variação da paravirtualização, estes não possuem a mesma facilidade de *upgrades* existentes em máquinas virtuais completas, o que pode acarretar com o tempo, sistemas operacionais dos *containers* mais desatualizados que os sistemas das máquinas completas (Felter, Ferreira, Rajamony, & Rubio, 2015). A tabela 4.1 exemplifica os formatos de virtualização presentes em cada *software* onde **VC** quer dizer virtualização completa, **PV** quer dizer paravirtualização, **NSO** representa virtualização a nível de sistema operacional e **HAV** representa *hardware assisted virtualization*.

Com utilização de *container* é possível rodar diversos processos de maneira isolada utilizando apenas um *kernel* em vários sistemas operacionais e ambientes (Joy, 2015). Além disso, como os *containers* utilizam o mesmo *kernel*, eles podem acessar os recursos do hóspede, diretamente, sem intermédios do hipervisor. E aqui temos uma vantagem do *Proxmox*.

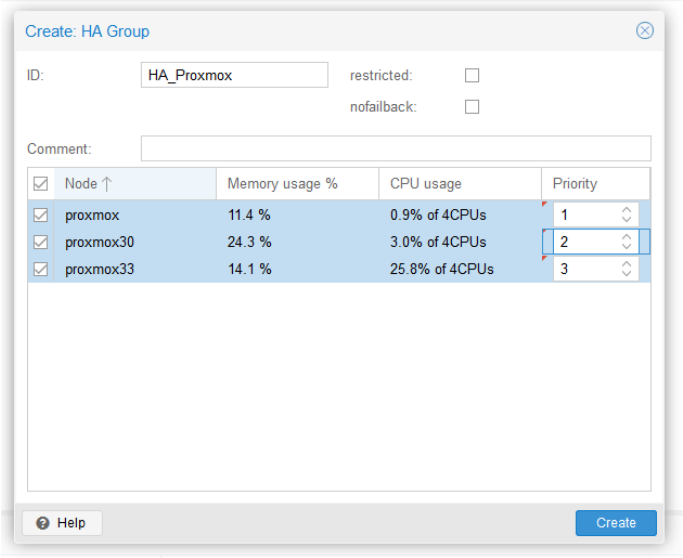


Figura 4.21: Criação do grupo de alta disponibilidade

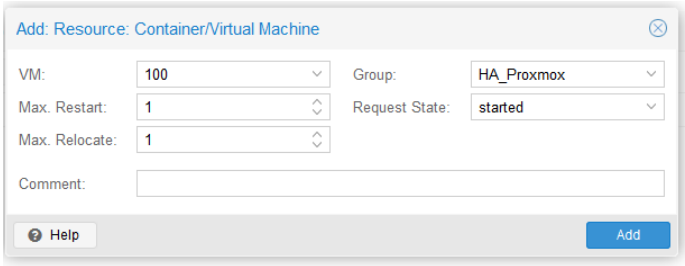


Figura 4.22: Adição da máquina virtual ao grupo de alta disponibilidade

SN	Hipervisor	Método de Virtualização			
		VC	PV	NSO	HAV
1	Proxmox VE	✓	✓	✓	✓
2	XCP-ng	✓	✓		✓

Tabela 4.1: Tabela de Métodos de Virtualização

4.6.2 Arquiteturas do servidor

A arquitetura do *XCP-ng* é baseada no *Xenserver*, que foi uma das pioneiras em virtualização e muitos virtualizadores utilizam as tecnologias da *Citrix* desenvolvedora do *Xenserver*. As figuras 2.10 e 4.24 mostram a arquitetura.

A camada *Hardware* são os recursos físicos do servidor. A camada *Xen Hypervisor* é a camada do hipervisor que permite múltiplos sistemas operacionais funcionarem em conjunto. O *Control Domain* (Dom0) é a camada de *software Linux* privilegiada que roda o serviço *XAPI*, que controla o gerenciamento de recursos e *drivers* e faz a conexão das máquinas virtuais com os recursos físicos através do hipervisor. Na imagem o

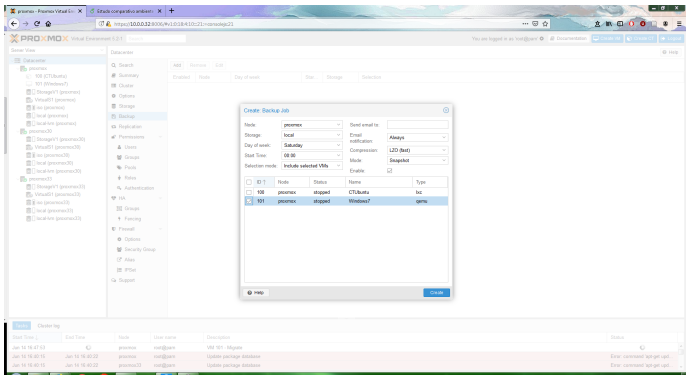


Figura 4.23: *Setup* de *backup* do *Proxmox*

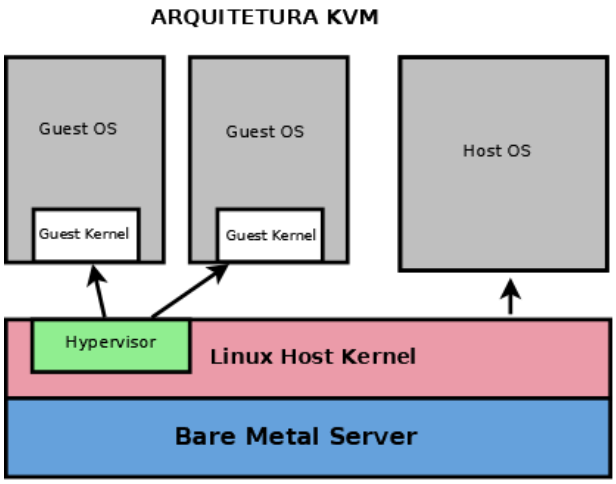


Figura 4.24: Arquitetura do Proxmox

Guest Domain (DomU) é camada de *software* não privilegiada, que roda as máquinas virtuais criadas pelos usuários. O *Proxmox* utiliza uma arquitetura baseada em *Kernel Based Machine* (KVM). A máquina virtual baseada em *kernel* (KVM) é uma tecnologia de virtualização *open source* baseada em *Linux*. Especificamente, com a KVM, você pode transformar o *Linux* em um *hipervisor*, permitindo que uma máquina *host* execute vários ambientes virtuais isolados. A KVM converte o *Linux* em um hipervisor tipo-1 (*bare-metal*). Para executar VMs, todos os hipervisores precisam de alguns componentes em nível de sistema operacional, como gerenciador de memória, agendador de processos, *stack* de entrada/saída (E/S), *drivers* de dispositivo, gerenciador de segurança, um *stack* de rede e muito mais. A KVM tem todos esses componentes por fazer parte do *kernel* do Linux. Toda máquina virtual é implementada como um processo regular do *Linux* que é programado pelo agendador do Linux padrão (Red Hat, 2019). A figura 4.24 mostra a arquitetura KVM e a tabela 4.2 exemplifica as arquiteturas de CPU compatíveis de cada *software*.

SN	Hipervisor	x86	x64	POWER	SPARC
1	Proxmox VE	✓	✓	**X**	**X**
2	XCP-ng	✓	✓	**X**	**X**

Tabela 4.2: Tabela com arquitetura de CPU

4.6.3 Sistema Operacional do Hóspede e Convidado

De acordo com a documentação do *Proxmox* e do *XCP-ng*, ambos são compatíveis com diversos sistemas operacionais convidados. Os mais conhecidos são os sistemas *windows* versões 7, 8 e 10, assim como diversas versões de *Linux*, como mostra a tabela 4.3. A figura 4.25 mostra a relação das versões de *containers* disponíveis no *Proxmox*.

N	Sistema Operacional	Proxmox	XCP-ng
1	Ubuntu	✓	✓
2	CentOS	✓	✓
3	LinuxSUSE	✓	✓
4	MS Windows 7	✓	✓
5	MS Windows 8	✓	✓
6	MS Windows 10	✓	✓
7	REDHAT	✓	✓

Tabela 4.3: Tabela com os principais SOs compatíveis

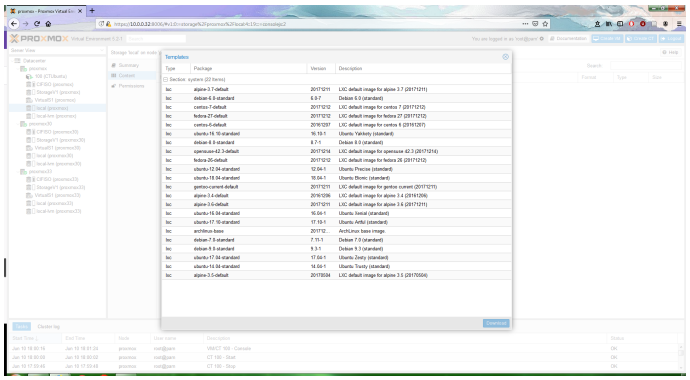


Figura 4.25: *Containers* disponíveis no Proxmox

4.7 Quantidade de recursos por hipervisor

A determinação dos recursos máximos que um hipervisor pode suportar é importante nas medições de carga e na confecção de planejamento de capacidade. Muitas vezes, em cenários de virtualização, máquinas virtuais precisam ser escaladas para melhorar seu desempenho, então, quanto mais recursos puder ser disponibilizado as essas máquinas

melhor. A tabela 4.4 ilustra as principais informações tirada da documentação dos hipervisores (Citrix, 2019) (Proxmox Server Solutions Gmbh, 2019).

SN	Característica	Proxmox VE	XCP-ng
1	Nº Máximo de VM's	Variável	1000
2	Nº Máximo de vCPU por convidado	Variável	32
3	Nº CPU's virtuais	160	160
4	Nº Máximo de RAM no *host*	12TB	5TB
5	Nº Máximo de RAM por VM	2000GB	1.5TB

Tabela 4.4: Tabela de escalabilidade de recursos

4.8 Gerenciamento de Rede

Os virtualizadores *XCP-ng* e *Proxmox* possuem diferentes formas de gerenciamento de rede. Mas ambos oferecem as mesmas funcionalidades como *switch* virtual, VLAN's, migração e agregação de interfaces de rede.

4.8.1 XCP-ng

O *XCP-ng* é capaz de suportar quatro interfaces vinculadas por servidor e até sete interfaces virtuais (lógicas) por máquina virtual. O *XCP-ng* durante a instalação cria uma interface lógica vinculada a placa de rede física para cada servidor. Quando os servidores são unidos num *cluster*, as interfaces lógicas são replicadas entre eles, como mostra as figuras 4.26 e 4.27. As alterações realizadas na ferramenta *XCP-Center* funcionam sem causar problemas na conexão.

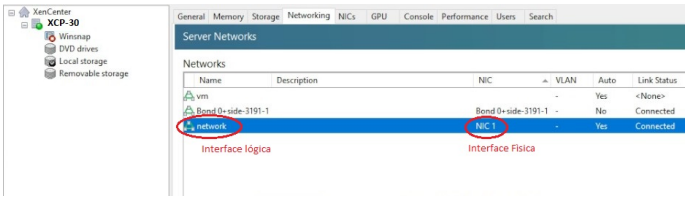


Figura 4.26: Agregação lógica no *XCP-ng*

4.8.2 Proxmox

No *Proxmox* também ocorre a vinculação de interface lógica e placa física durante a instalação, assim como a replicação entre servidores unidos no *cluster*. Entretanto, foi observado que nas mudanças de configuração via interface *web* ocorre perda de conexão, sendo necessário a configuração via linha de comando no próprio servidor.

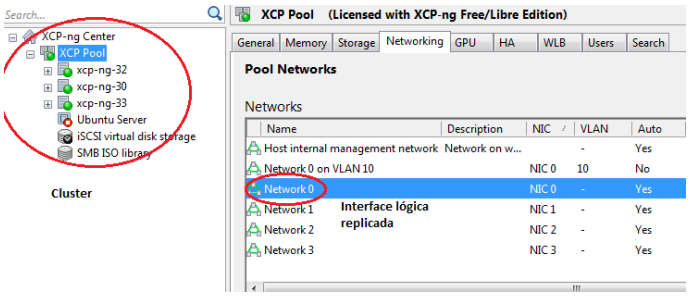


Figura 4.27: Agregação lógica no *Cluster*

4.8.3 VLAN's (IEEE 802.1q)

O *XCP-ng* e *Proxmox* são compatíveis com a tecnologia de VLANs definido pelo protocolo do *Institute of Electrical and Electronics Engineers*(IEEE) 802.1q. Este protocolo permite a criação de redes virtuais locais (VLANs) dentro de uma rede *ethernet*[3], propiciando que a rede física seja dividida em várias redes virtuais.

4.8.4 NIC *Bonding*

NIC *Bonding* é uma tecnologia de agregação de placas de rede. Esta tecnologia possibilita a união de velocidades de cada placa e ainda pode ser usada para garantir mais segurança nas transmissões em caso de falha de alguma placa de rede. As saídas disponíveis são unidas em único canal compartilhado, facilitando o envio e recebimento de pacotes, consequentemente, melhorando o desempenho e segurança (Aust, Kim, Davis, Yamaguchi, & Obana, 2006).

4.8.4.1 *XCP-ng*

O *XCP-ng* fornece um tipo de rede agregada (*bonded network*) que cria uma união entre duas ou mais placas físicas para criar um único canal com mais desempenho e segurança.

4.8.4.2 *Proxmox*

O *Proxmox* fornece sete formas de redes agregadas:

- ***Round-robin* (*balance-rr*):** Transmite os pacotes em ordem sequencial pelas placas de rede disponíveis. Esse modo garante *load balance* e tolerância a falha.
- ***Active-backup*(*active-backup*):** Nessa forma das placas de rede disponíveis apenas

[3]É uma arquitetura de interconexão para redes locais

uma é usada e as outras disponíveis apenas são usadas caso a primeira sofra uma falha. Esse modo oferece tolerância a falhas.

- ***XOR* (*balance-xor*):** Transmite os pacotes entre as mesmas origens e destinos. Selecionando as mesmas placas de rede para destinos fixos. Essa forma garante *load balancing* e tolerância a falha.

- ***Broadcast*:** Transmite os pacotes em todos as placas de rede disponíveis. Essa forma garante tolerância a falha

- **IEE 802.3ad *Dynamic link aggregation*(LACP):** Cria grupos de agregação que compartilham as mesmas velocidades e configurações *duplex*. Ao transmitir pacotes utiliza todas as placas físicas vinculadas no canal de agregação de acordo com a especificação 802.3ad.

- ***Adaptive transmit load balancing*(*balance-tlb*):** Essa forma é um *driver linux* de agregação que não necessita de configuração no *switch* conectado. O tráfego de saída é distribuído pelas placas de rede. Já o tráfego de chegada é recebido por apenas uma placa de rede e caso essa falhe outra irá substituí-la.

- ***Adaptive load balancing* (*balance-alb*):** Inclui a mesma configuração do *balance-tlb* e recebe *load balancing* para tráfego IPV4 e não necessita de configuração no *switch*. O *loud balancing* é feito através de negociação com protocolo *Address Resolution Protocol*(ARP). O *driver* de agregação intercepta o pacote ARP de resposta enviado pelo sistema local na hora de sair e reescreve o endereço de destino entre as placas de rede disponíveis para receber.

Pode-se notar que o *Proxmox* oferece muito mais opções de agregação de placas de rede.

4.9 Funcionalidades Básicas dos hipervisores

Nessa seção será descrito funcionalidades básicas encontradas em cada um dos hipervisores *Proxmox* e *XCP-ng*.

4.9.1 *Snapshot*

Uma *snapshot* de uma máquina virtual é um mecanismo que permite ao administrador fazer uma cópia da máquina virtual, salvando suas configurações e informações em disco

num determinado momento. Caso seja necessário, é possível retornar a máquina a esse estado salvo (Hanqian Wu, Yi Ding, Winer, & Li Yao, 2010). O *snapshot* no *XCP-ng* é feito através via propriedades da máquina virtual, como mostra a figura 4.28. O *snapshot* no *Proxmox* é realizado nas configurações de *backup* como mostra a figura 4.29.

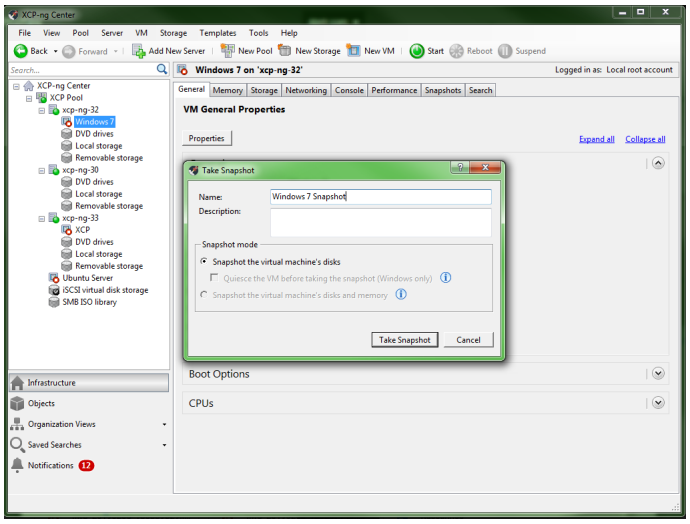


Figura 4.28: *Snapshot* no *XCP-ng*

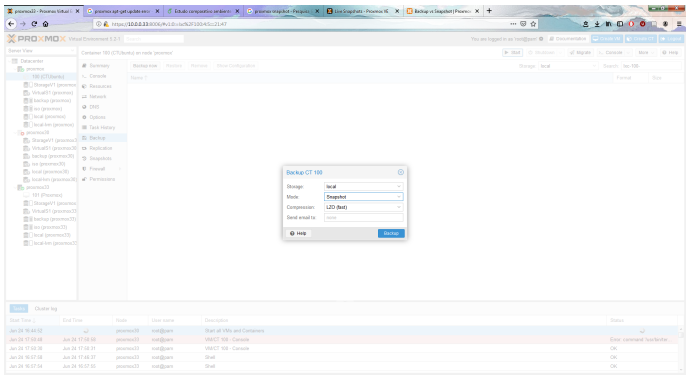
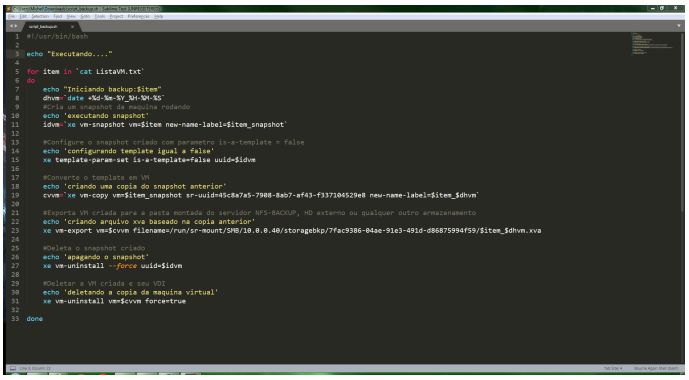


Figura 4.29: *Snapshot* no *Proxmox*

4.9.2 *Backup/Restore*

A funcionalidade de criar *backups* permite salvar as informações das máquinas em caso de desastres, sendo possível a recuperação desta a partir do último *backup*. Uma boa prática é criar procedimentos de *backups* periódicos de acordo com a política de *backup* da organização. A ideia de se guardar *backups* foi iniciado pela IBM com as aplicações de gerenciamento de *storage*. O *backup* e *restore* foram as primeiras formas de gerenciamento de *storage* (Barnoschi, 2019). Para fazer o *backup* no *XCP-ng*, usando a ferramenta *XCP-Center*, precisamos fazê-lo por meio de *scrips*, conforme mostrado na imagem 4.30. No *Proxmox* existe uma funcionalidade chamada *backup* que permite customizar as rotinas de *backup* mais facilmente, visto na figura 4.31.



```
#!/usr/bin/bash

echo "Executando...."

for item in `cat ListaVM.txt`
do
    echo "Iniciando backup:$item"
    dhvm="date +%d-%m-%Y_%H-%M-%S"
    #Cria um snapshot da maquina rodando
    echo 'executando snapshot'
    idvm=`xe vm-snapshot vm=$item new-name-label=$item_snapshot`

    #Configure o snapshot criado com parametro is-a-template = false
    echo 'configurando template igual a false'
    xe template-param-set is-a-template=false uuid=$idvm

    #Converte o template em VM
    echo 'criando uma copia do snapshot anterior'
    cvvm=`xe vm-copy vm=$item_snapshot sr-uuid=45c8a7a5-7908-8ab7-af43-f3371045298 new-name-label=$item_$dhvm`

    #Exporta VM criada para a pasta montada do servidor NFS-BACKUP, HD externo ou qualquer outro armazenamento
    echo 'criando arquivo xva baseado na copia anterior'
    xe vm-export vm=$cvvm filename=/run/sr-mount/SMB/10.0.0.40/storagebkp/7fac9386-04ae-91e3-491d-d86875994f59/$item_$dhvm.xva

    #Deleta o snapshot criado
    echo 'apagando o snapshot'
    xe vm-uninstall --force uuid=$idvm

    #Deletar a VM criada e seu VDI
    echo 'deletando a copia da maquina virtual'
    xe vm-uninstall vm=$cvvm force=true

done
```

Figura 4.30: *Script* do *xenserver* para *backup*

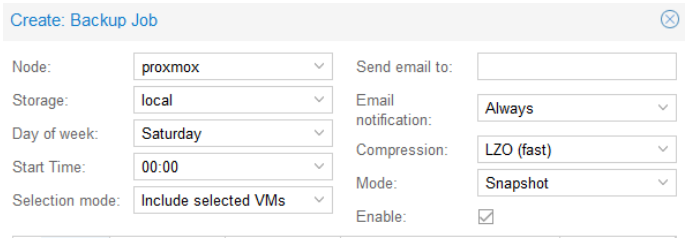


Figura 4.31: Configuração da rotina de *Backup* do Proxmox

4.9.3 Relatório de Desempenho

As ferramentas de virtualização proprietárias possuem uma funcionalidade chamada de análise de desempenho. Este tipo de relatório apresenta informações de uso de processador, memória, número de acessos, tráfego de rede, consumo de energia, entre outros. A interface *web* do *Proxmox* possuí uma seção que mostra análises do sistema, como mostra a figura 4.32. A ferramenta *XCP-Center* possui uma aba monitoramento, mas por ser uma versão adaptada da aplicação original, da empresa *Citrix* (*XenCenter*), feita pela comunidade, apresenta muita instabilidade.

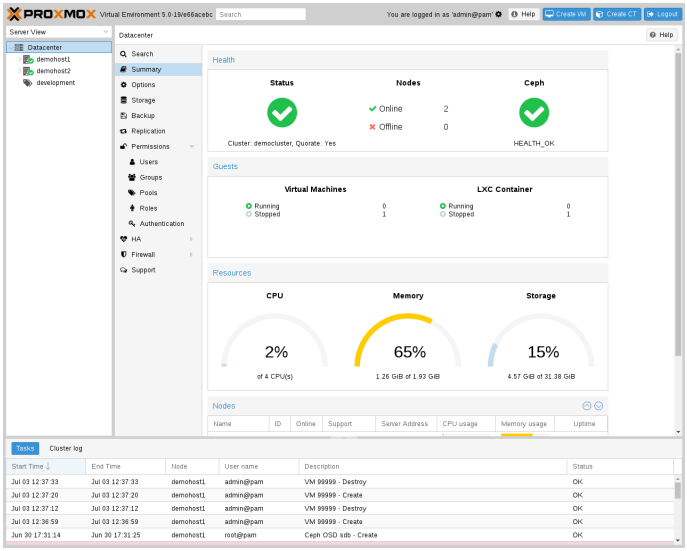


Figura 4.32: *Dashboard* do *Proxmox*

4.9.4 Clonagem e *Template* de Máquinas Virtuais

Quando se clona uma máquina virtual, cria-se uma cópia completa daquela máquina, incluindo todas as configurações de *hardware* e arquivos do disco da máquina. A clonagem de máquinas virtuais é muito útil quando se necessita de um ambiente com diversas máquinas iguais, sem a necessidade de reinstalação de todo o sistema operacional. Ao criar uma máquina que se deseja clonar, muitas vezes é mais interessante transformar essa máquina em um *template*. O *template* é uma espécie de máquina modelo com configurações pré-determinadas que podem ser rapidamente transformadas em máquinas virtuais. O *template* oferece uma forma mais segura de preservar as configurações originais da máquina virtual que se deseja criar cópias, pois, não se permite edições. As máquinas criadas a partir de um *template* são cópias independente deste. As mudanças nas máquinas criadas não afetam o *template*. Nos hipervisores *XCP-ng* e *Proxmox* a clonagem é feita facilmente ao acessar as propriedades da máquina virtual, conforme a imagem 4.33.

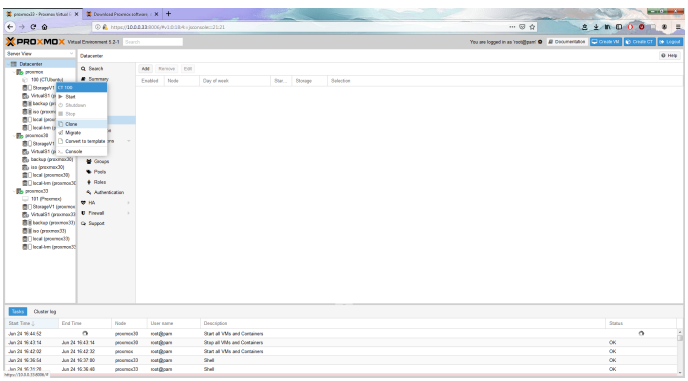


Figura 4.33: Clonagem no Proxmox

4.9.5 Exportação e importação de Máquinas Virtuais

Na virtualização, a exportação e importação de máquinas virtuais é importante, pois, em um cenário de migração, entre soluções de fabricantes diferentes, esta característica é necessária.

4.9.5.1 *XCP-ng*

Com *XCP-ng* pode-se importar arquivos nos formatos:

- ***Virtual Hard Disk* (VHD)**
- ***Virtual Machine Disk* (VMDK)**
- ***Open Virtualization Format* (OVF) (OVA)**

A exportação no *XCP-ng* pode ser feita nos formatos OVF e OVA, como mostra figura 4.34.

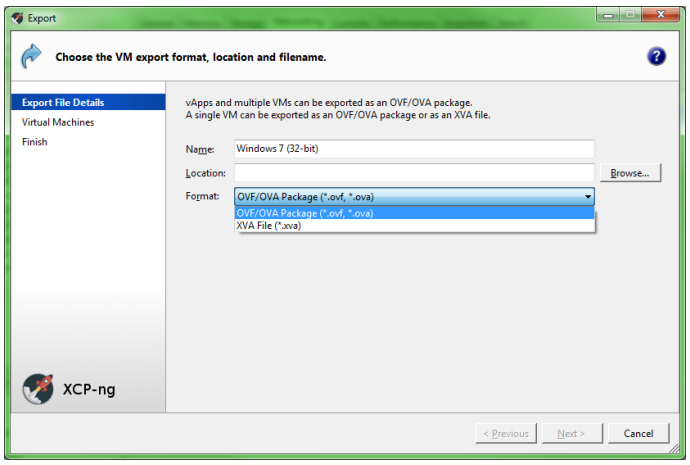


Figura 4.34: Formatos de exportação

4.9.5.2 *Proxmox*

O *Proxmox* exporta no formato *lxc* para *containers* e um formato *qemu* padrão do virtualizador KVM. A exportação no *Proxmox* é integrada com a funcionalidade de *backup*, capaz de optar por formatos *qcow2*, *raw*, e *vmdk*. O formato *qcow2* é um formato com alto consumo periférico e velocidade de processamento baixa. Não é recomendando utilizar esse formato para máquinas virtuais com muito espaço em disco, como por exemplo, um banco de dados. Entretanto, o formato *qcow2* é útil caso haja limitações financeiras e de espaço de armazenamento. O formato *raw* possui melhor desempenho e o tamanho em disco pode ser fixo ou *thin-provisioned*, então, é o formato preferível para exportações

no *Proxmox*. O formato *vmdk* é um formato mais usado caso seja migrado para uso em outros ambientes que não sejam *Proxmox* (Ally, 2018).

A tabela 4.5 mostra um resumo das informações de importação e exportação dos hipervisores envolvidos no estudo *XCP-ng*, *Proxmox* e do hipervisor *VMWare* presente no ambiente real.

	Formato	XCP		Proxmox		VMWare	
		imp	exp	imp	exp	imp	exp
1	VHD/VMDK	✓	**X**	✓	**X**	✓	✓
2	OVF/OVA	✓	✓	✓	**X**	✓	✓
3	lxc	**X**	**X**	✓	✓	**X**	**X**
4	qcow2 / raw	**X**	**X**	✓	✓	**X**	**X**

Tabela 4.5: Tabela de formatos de exportação e importação

4.10 Funcionalidades avançadas dos hipervisores

As funcionalidades abaixo são encontradas apenas em ambientes com mais de um hipervisor. Este tipo cenário oferece mais segurança, desempenho e economia, pois nele, podem ser configurados alta disponibilidade, balanceamentos de carga entre outras configurações avançadas.

4.10.1 Alta Disponibilidade

A função alta disponibilidade é ativada a partir de dois servidores, de forma que eles fiquem conectados por um monitoramento à prova de falhas de *hardware* e *software*. Em um sistema com alta disponibilidade, em caso de um servidor do grupo sofrer algum desastre físico ou digital, as máquinas virtuais e serviços rodando neste servidor serão migradas para o outro disponível do grupo. Este processo ocorre sem perda de dados e com menor tempo possível de parada nos serviços.

Com o aumento da utilização de serviços digitais, é praticamente inconcebível qualquer tipo de interrupção nestes, pois, além de desagradável, pode ocasionar prejuízos financeiros e na imagem do prestador do serviço. Para garantir que não ocorram interrupções nos serviços prestados, além de um ambiente com alta disponibilidade, é necessário, muitas vezes, dispôr de *hardwares* redundantes, que entrem em funcionamento automaticamente, quando a falha de um dos componentes em utilização for detectada. Também é recomendado que o conjunto de servidores que irão ser configurados com alta disponibi-

lidade possuam as mesmas características para fornecer os mesmos recursos em todos os nós participantes do grupo.

Quanto mais redundância existir no sistema com alta disponibilidade, menores serão os SPOF (*Single Points Of Failure*), estes são os pontos de falha de um sistema. Quanto menores esses pontos, menor será a probabilidade de interrupções no serviço. Com servidores ligados em alta disponibilidade, diminuem os SPOF, além de facilitar no gerenciamento nas aplicações de atualizações e manutenções, sem a necessidade de interrupções nos serviços prestados por aquele *cluster*.

Atualmente os *Clusters* são soluções construídas a partir de *hardwares* acessíveis, que entregam sistemas altamente escaláveis e de custos razoáveis. Os *Clusters* são implementações que unem servidores em grupos, compartilhando recursos de *hardwares* e criando ambientes redundantes. Este conceito está diretamente relacionado aos sistemas de alta disponibilidade. Pode-se imaginar a união de vários computadores ou servidores resultando em uma única máquina de porte maior. Com a alta disponibilidade, diversas funcionalidades são adicionadas ao ambiente virtual. A migração pode ser realizada com máquinas ligadas e com baixo *downtime*. Com os servidores unidos em um agrupamento, pode-se realizar balanceamento de carga para otimizar a utilização de processamento. Isso otimiza o desempenho do ambiente, além da segurança ao manter os serviços sempre ligados.

Cada um dos softwares possui uma seção específica para configurar a alta disponibilidade. No *Proxmox*, como dito anteriormente, é feito separando os servidores em grupos que fornecerem a funcionalidade as máquinas atribuídas. A figura 4.35 mostra como é a adição da máquina. Já no *XCP-ng*, a configuração é feita automaticamente nos servidores do *cluster*, ao se escolher ativar alta disponibilidade pela aba alta disponibilidade. A figura 4.36 mostra a funcionalidade ativada.

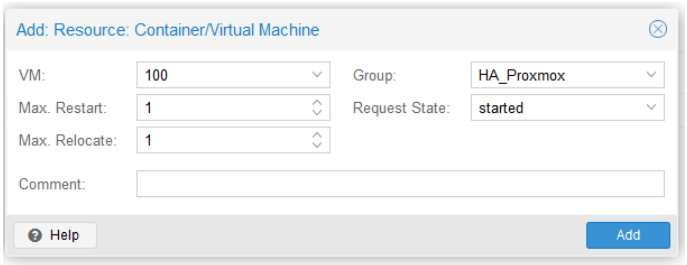


Figura 4.35: Gerenciamento de Alta Disponibilidade no *Proxmox*

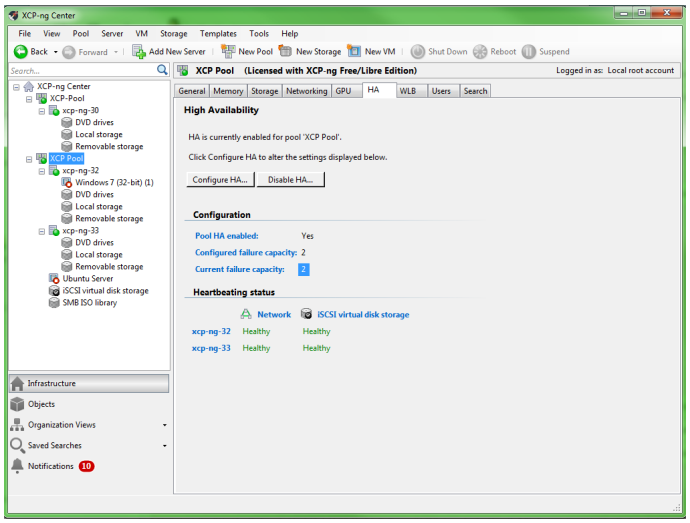


Figura 4.36: Alta Disponibilidade no *XCP-ng*

4.10.2 *Live Migration*

Live migration refere-se ao processo de mover uma máquina virtual ligada, entre servidores físicos conectados em *cluster*, sem desligar a mesma. As conexões de memória, *storage* e conexão com a rede são transferidas entre os servidores. Com *live migration* é possível transferir as máquinas virtuais no estado ligada entre os servidores no *cluster* com o mínimo de *downtime*, para que os serviços fornecidos pelas máquinas não desliguem, como exemplifica a imagem 4.37. Para que essa funcionalidade seja ativada, é necessário habilitar alta disponibilidade no *cluster*. Para testar essa função nos dois hipervisores preparamos uma estação física para fazer *pings* constantes a uma máquina virtual ligada. Em seguida migramos essa máquina para outro servidor e observamos pequenas perdas de pacotes ICMP, exatamente no momento que a máquina virtual migra para o outro servidor do *cluster*. O teste em ambos os hipervisores funcionaram muito bem.

4.11 Outras funcionalidades

Uma funcionalidade importante em ambientes virtuais com alta demanda é o balanceamento de carga. Esta função permite distribuir a carga igualmente entre os servidores, evitando sobrecarga e falhas no serviço. Pode ser feito o controle dos recursos físicos como processamento, memória e energia assim como controle de tráfego de rede.

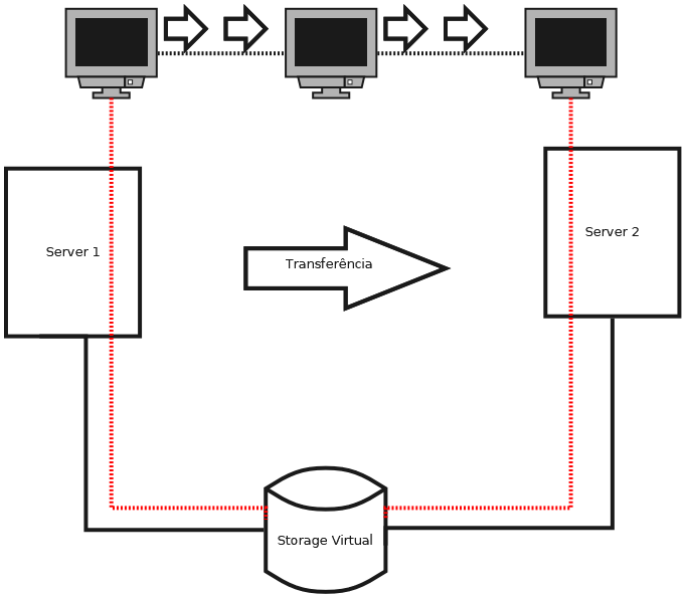


Figura 4.37: *Live migration*

4.11.1 Balanceamento de carga

Balanceamento de carga permite alocar os recursos físicos de forma mais eficaz. O ambiente é monitorado para avaliar o consumo de energia e recursos dos servidores de forma que estes sejam equilibrados entre todos os hipervisores disponíveis. Entretanto no *XCP-ng*, como mostra a figura 4.38, seria necessário um servidor adicional para controle do gerenciamento de carga. No *Proxmox* essa funcionalidade é configurada pela escolha de prioridades, ao colocarmos servidores num grupo de alta disponibilidade.

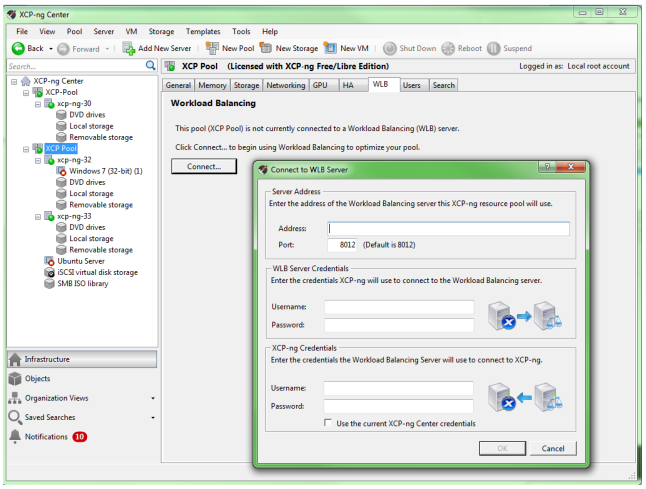


Figura 4.38: Opção de habilitar *Workload Balance* necessita de *server* adicional

4.11.2 Balanceamento de carga para redes

Balanceamento de carga para redes é uma funcionalidade que permite distribuir o tráfego de pacotes nas interfaces de redes agregadas, aumentando o desempenho e segurança. Como o ambiente disponibilizava um *switch cisco* compatível com tecnologia de balanceamento de rede, *Link Aggregation Control Protocol*(LACP) [4], foi possível testar a configuração de *Network Interface Card* (NIC) *bonding* disponível no *XCP-ng* e no *Proxmox* como exemplifica a figura 4.39.

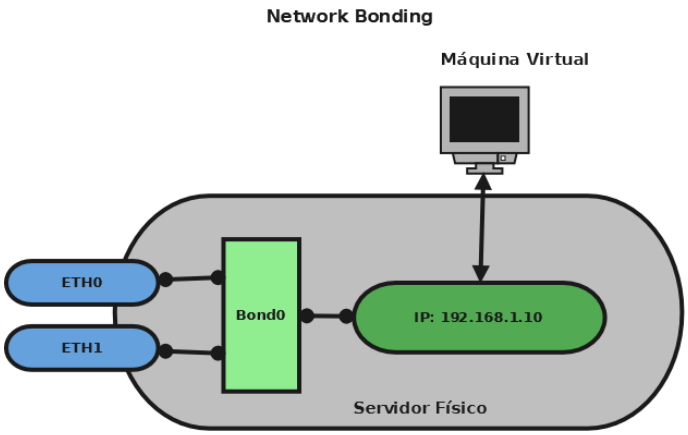


Figura 4.39: Representação do NIC *Bonding*

Primeiro entretanto seria necessário preparar as portas do *switch* cisco para essa tarefa seguindo o guia do fabricante (Systems, 2019). Entretanto após a configuração das portas e da criação da rede conjunta, os tempos de transferência foram comparados com LACP e sem LACP, e as médias eram muito próximas, e não foi possível concluir se houve vantagem nesse procedimento ou se não funcionou de modo geral. Apesar dos tempos de transferência não demonstrarem vantagem aparente, ao simular um desastre de rede removendo um dos dois cabos de rede configurado com LACP, a conexão do servidor não era perdida, assim o teste de *failover* a nível de placa de rede foi, pelo menos, bem sucedido nos dois *softwares*.

4.12 Resultado das análises

Com o desfecho das comparações e identificação das vantagens de cada *software*, podemos estabelecer uma recomendação de escolha de qual *software* utilizar com base no levantamento apresentado na tabela 4.6, que resume essas comparações. Para optarmos pela utilização de um dos dois *softwares* avaliados para o ambiente do CCET, levamos

[4]Tecnologia que agrega um conjunto de saídas de rede em uma saída lógica única unindo as velocidades de transmissão (Systems, 2019)

em consideração o que é necessário para o ambiente. Como ambos os *softwares* são capazes de fornecer as principais funcionalidades para o ambiente focado, entendemos que o *Proxmox* deve ser recomendado por possuir maior facilidade de gerenciamento e, principalmente, pela possibilidade de utilizar *containers*. O uso de *containers* reduz e otimiza o espaço em disco em função do reaproveitamento do sistema operacional do *host*, sendo que a escassez de espaço em disco vem se apresentando como um dos principais problemas do ambiente de *datacenter* do CCET, refletindo-se também no espaço necessário para *backup*. Essa propriedade, alinhada ao fato de que a maioria das máquinas do ambiente real *datacenter* do CCET são do ambiente operacional *Linux*, aponta o *Proxmox* como sendo a solução mais vantajosa.

Funcionalidades para comparação	**Proxmox VE**	**XCP-ng**
Complexidade da instalação	Simples, configuração de IP manual	Simples, configuração de IP automática
Atribuição do *storage* compartilhado	Complexo	Simples
Configuração de *Cluster*	Simples	Simples
Migração de máquina virtual	Simples	Simples
Métodos de Virtualização	Virtualização Completa, Paravirtualização, Containers	Virtualização Completa, Paravirtualização
Arquiteturas do servidor	KVM	Citrix Hipervisor
Sistema Operacional Hóspede/Convidado	Compatível com os principais *softwares* do mercado	Compatível com os principais *softwares* do mercado
Gerenciamento de Rede	Complexo	Simples
Criação de *Snapshot*	Simples	Simples
Complexidade da rotina *Backup*/*Restoration*	Simples	Complexo
Monitoramento de Desempenho	Disponível na interface	Não funciona bem na aplicação
Clonagem e *Templates*	Simples	Simples
Formatos de importação/exportação	Importa os principais formatos. Exporta diferentes formatos	Importa os principais formatos. Só exporta pra Xenserver
Alta disponibilidade	Simples	Simples
Live Migration	Simples	Simples
Balanceamento de carga	Simples	Necessário servidor adicional
Network Bonding	Não conclusivo	Não conclusivo

Tabela 4.6: Comparação dos *softwares*

5. Conclusão

A elaboração deste trabalho busca identificar a viabilidade de substituir o *software* com licença paga no ambiente da universidade por um *software* com licença livre. Esse estudo permitiu analisar como dois *softwares* de virtualização livres funcionariam, em um cenário baseado no ambiente real da universidade. Além disso, com o que foi aprendido, poderemos escolher qual o mais indicado para utilização no ambiente da universidade. Os dois *softwares* são excelentes opções do mercado para se utilizar. Ambos estão aptos a resolver as necessidades da universidade, além de oferecerem as funcionalidades avançadas, antes, presentes apenas em ferramentas proprietárias. A escolha por qualquer uma das soluções estará baseada na metodologia utilizada neste trabalho, onde foi montado um ambiente semelhante ao de produção, para depois serem feitos os testes e as comparações.

Com a instalação e configuração das soluções de forma a replicar o ambiente real, podemos observar com clareza, como cada *software* se comportaria com as principais tarefas do ambiente real. Com isso poderíamos identificar qual seria a melhor opção. Entretanto, foi observado que as duas soluções são ótimas escolhas. Cada *software* tem suas vantagens e desvantagens. O *XCP-ng* na fase de instalação é simples e intuitivo, como por exemplo na configuração de rede. A possibilidade de configurar a rede automaticamente com protocolo DHCP facilita muito. Ao fim da instalação, as configurações do ambiente são guiadas pela aplicação, tornando a preparação do ambiente simples e intuitiva.

No *Proxmox* foi bem mais difícil configurar a rede, pois, ao fazer modificações pela interface, as máquinas não se conectavam. Ao se fazer modificações na rede em geral, perdia-se conexão com servidor, obrigando a configuração por linha de comando. Encontrar a configuração ideal por linha de comando foi uma tarefa difícil e demorada. Outra dificuldade foi a atribuição do *storage* virtual. Ao contrário do *XCP-Center* que faz uma configuração guiada simplificando muito o processo de conexão com *storage*, no *Proxmox*, primeiro é necessário criar uma conexão com o *storage* e depois configurar essa conexão para disponibilizar o *storage*. Esse processo não é intuitivo, não fica claro que

são necessários esses dois passos para configuração. Assim, cada um dos *softwares* teve uma vantagem em relação a desvantagem do outro.

Algumas funcionalidades são configuradas de forma diferente entre os *softwares*. A alta disponibilidade no *XCP-ng* é aplicada automaticamente no *cluster* desejado. Já no *Proxmox* é bem mais elaborado, dando a opção de criar grupos de máquinas e escolher quais grupos possuem a funcionalidade, assim como dar prioridades aos membros do grupo. Configurar *backup* no *XCP-Center* é complicado, se faz necessário utilizar *scripts* dentro do servidor. E os formatos de exportação do *XCP-ng* são limitados oferecendo apenas OVF e OVA. Com o *Proxmox* a funcionalidade de *backup* é facilmente configurada pela interface, sem precisar de *scripts* e possui mais formas de exportação.

Por utilizarmos os recursos disponíveis na universidade, algumas tarefas não foram possíveis como realizar balanceamento de carga no *XCP-ng*. Era necessário um servidor extra de controle não disponível no ambiente. Já no *Proxmox*, é possível gerenciar carga com controle de prioridades na configuração de alta disponibilidade. Em função de algumas limitações com relação ao gerenciamento do *switch*, os testes de agregação de placas de rede foi limitado. Testes de *benchmark* não foram feitos por utilizarmos equipamentos reutilizados, alguns falhos e que não representariam o funcionamento adequado para um ambiente real.

Ao fazermos a implementação e analisarmos o funcionamento de cada *software*, podemos identificar *softwares* de uso livre capazes de substituir soluções proprietárias. Nesta análise, era esperado obtermos uma clara diferença entre os funcionamentos dos *softwares* para que a escolha do mais adequado fosse feita. Como ambos os *softwares* analisados se provaram capazes para o ambiente da universidade, optamos por escolher aquele com mais facilidade de gerenciamento nas funcionalidades mais importantes, como alta disponibilidade e *backup*, assim como a minimização do principal problema do ambiente de *datacenter* do CCET, que é o pouco espaço de armazenamento em disco. Assim sendo, para esse ambiente recomendamos a utilização do *Proxmox* devido à facilidade de gerenciamento pela interface gráfica, por garantir mais controle de alta disponibilidade, por configurar *backup* sem *scripts* e, principalmente, por fornecer *containers*. Com o uso de *containers*, pode-se reduzir e otimizar o espaço em disco em função do reaproveitamento do sistema operacional do *host*, alinhado ao fato de que a maioria das máquinas do ambiente real são do ambiente operacional *Linux*. Apoiado pelo estudo feito pela IBM, que mede o desempenho entre máquinas virtuais e *containers Linux* através de testes de *benchmark*, concluiu-se que os *containers* possuem desempenho igual ou até melhor na maioria dos testes, além de serem mais seguros e práticos (Felter et al., 2015). Portanto, a utilização de *containers* também torna o ambiente de *datacenter* melhor e mais seguro.

Como trabalhos futuros, esse estudo comparativo abre espaço para um maior aprofundamento, envolvendo a aplicação dos *softwares* no ambiente real com testes de desempenho, ou uma nova comparação com outros *softwares* no mercado.

Referências

Ally, S. (2018, March). Comparative Analysis Of Proxmox VE And Xenserver As Type 1 Open Source Based Hypervisors. *International Journal of Scientific & Technology Research*, *7*(3), 72–77. Retrieved 2018-11-12, from http://www.ijstr.org/paper-references.php?ref=IJSTR-0318-18753

Aust, S., Kim, J., Davis, P., Yamaguchi, A., & Obana, S. (2006, November). Evaluation of Linux Bonding Features. In *2006 International Conference on Communication Technology* (pp. 1–6). doi: 10.1109/ICCT.2006.341935

Barnoschi, A. (2019). BACKUP AND DISASTER RECOVERY. , 6.

Citrix, X. (2019, May). Citrix XenServer 7.6. , 732.

des Ligneris, B. (2016). *Comparison of Open Source Virtualization Technology* [Technology]. Retrieved 2018-11-12, from https://www.slideshare.net/bligneri/comparison-of-open-source-virtualization-technology

Felter, W., Ferreira, A., Rajamony, R., & Rubio, J. (2015, March). An updated performance comparison of virtual machines and Linux containers. In *2015 IEEE International Symposium on Performance Analysis of Systems and Software (ISPASS)* (pp. 171–172). doi: 10.1109/ISPASS.2015.7095802

Hanqian Wu, Yi Ding, Winer, C., & Li Yao. (2010, November). Network security for virtual machine in cloud computing. In *5th International Conference on Computer Sciences and Convergence Information Technology* (pp. 18–21). doi: 10.1109/ICCIT.2010.5711022

Joy, A. M. (2015, March). Performance comparison between Linux containers and virtual machines. In *2015 International Conference on Advances in Computer Engineering and Applications* (pp. 342–346). doi: 10.1109/ICACEA.2015.7164727

KVM contributors. (2016, November). *KVM*. Retrieved 2019-07-01, from http://www.linux-kvm.org/page/Main_Page

Obasuyi, G. C., & Sari, A. (2015). Security Challenges of Virtualization Hypervisors in Virtualized Hardware Environment. *International Journal of Communications, Network and System Sciences*, *08*(07), 260–273. Retrieved 2019-06-05, from http://www.scirp.org/journal/doi.aspx?DOI=10.4236/ijcns.2015.87026 doi: 10.4236/ijcns.2015.87026

Padhy, R. P., Patra, M. R., & Satapathy, S. C. (2010). VIRTUALIZATIONTECHNIQUES & TECHNOLOGIES: STATE-OF-THE-ART. , 16.

Patel, A., Daftedar, M., Shalan, M., & El-Kharashi, M. W. (2015, March). Embedded Hypervisor Xvisor: A Comparative Analysis. In *2015 23rd Euromicro International Conference on Parallel, Distributed, and Network-Based Processing* (pp. 682–691). doi: 10.1109/PDP.2015.108

Proxmox Server Solutions GmbH. (2019, May). *Compare Proxmox VE vs VMware vSphere, Hyper-V, XenServer.* Retrieved 2018-11-12, from https://www.proxmox.com/en/proxmox-ve/comparison

Proxmox Server Solutions Gmbh. (2019). Proxmox VE Administration Guide. , 389.

Qinlu He, Zhanhuai Li, & Xiao Zhang. (2010, October). Analysis of the key technology on cloud storage. In *2010 International Conference on Future Information Technology and Management Engineering* (Vol. 1, pp. 426–429). doi: 10.1109/FITME.2010.5656540

Rampazzo, L. (2005). *Metodologia científica*. Edicoes Loyola. (Google-Books-ID: rwyufjs_DhAC)

Red Hat. (2019). *O que é KVM?* Retrieved 2019-08-06, from https://www.redhat.com/pt-br/topics/virtualization/what-is-KVM

Sahoo, J., Mohapatra, S., & Lath, R. (2010). Virtualization: A Survey on Concepts, Taxonomy and Associated Security Issues. In *2010 Second International Conference on Computer and Network Technology* (pp. 222–226). Bangkok, Thailand: IEEE. Retrieved 2019-04-15, from http://ieeexplore.ieee.org/document/5474503/ doi: 10.1109/ICCNT.2010.49

Systems, C. (2019, April). *Link Aggregation Control Protocol (LACP) (802.3ad) for Gigabit Interfaces.* Retrieved 2019-04-29, from https://www.cisco.com/c/en/us/td/docs/ios/12_2sb/feature/guide/gigeth.html

Veras, M., & Carissimi, A. (2019). *Virtualização de Servidores | Virtualização | Máquina Virtual.* Retrieved 2019-07-06, from https://pt.scribd.com/doc/50570155/Virtualizacao-de-Servidores