UNIVERSIDADE FEDERAL DE SANTA MARIA
COLÉGIO TÉCNICO INDUSTRIAL DE SANTA MARIA
CURSO SUPERIOR DE TECNOLOGIA EM REDES DE COMPUTADORES

OTIMIZAÇÃO DO SISTEMA DE MÁQUINAS VIRTUAIS DO CTISM UTILIZANDO LXC/LXD

TRABALHO DE CONCLUSÃO DE CURSO

Bruno Augusto Rizzetti

Santa Maria, RS, Brasil
2015

OTIMIZAÇÃO DO SISTEMA DE MÁQUINAS VIRTUAIS DO CTISM UTILIZANDO LXC/LXD

Bruno Augusto Rizzetti

Trabalho apresentado ao Curso de Graduação em Tecnologia em
Redes de Computadores, Área de concentração em Segurança da Informação, da
Universidade Federal de Santa Maria (UFSM, RS),
como requisito parcial para obtenção do grau de
Tecnólogo em Redes de Computadores.

Orientador: Prof. Me. Tiago Antonio Rizzetti
Coorientador: Prof. Me. Renato Preigschadt de Azevedo

Santa Maria, RS, Brasil
2015

Universidade Federal de Santa Maria
Colégio Técnico Industrial de Santa Maria
Curso Superior de Tecnologia em Redes de Computadores

A Comissão Examinadora, abaixo assinada,
aprova a Monografia

OTIMIZAÇÃO DO SISTEMA DE MÁQUINAS VIRTUAIS DO CTISM UTILIZANDO LXC/LXD

elaborada por
Bruno Augusto Rizzetti

como requisito parcial para obtenção do grau de
Tecnólogo em Redes de Computadores

COMISSÃO EXAMINADORA

Tiago Antonio Rizzetti, Me.
(Presidente/Orientador)

Bolívar Menezes da Silva, Tecng. (UFSM)

Rodrigo Castro Gil, Bel. (UFSM)

Santa Maria, 11 de dezembro de 2015.

DEDICATÓRIA

Dedico essa conquista, primeiramente a minha família que sempre me apoiou, seja com palavras de conforto e motivação, ou até com provocações que sempre nos motivam a ir mais longe chegando a alcançar o que tanto desejamos.

Dedico também, a todos amigos que me apoiaram e de uma forma ou de outra deram forças para continuar até chegar onde estou hoje. Essa não é apenas uma conquista minha, essa conquista é de todos vocês também.

AGRADECIMENTOS

Agradeço a minha família por todo apoio e dedicação a mim fornecido, a todos amigos que fiz no curso, em especial aos colegas de bolsa os quais me acompanharam por grande parte dessa jornada. Faz parte também, desse grupo de pessoas que me motivaram e ajudaram chegar onde cheguei, os professores do curso. Que na maioria não se tratam mais de apenas professores e sim amigos, que de uma forma ou outra sempre nos ajudavam no que estivesse ao seu alcance.

Seja entre um café ou outro na famosa "sala dos bolsistas - 302", agradeço a todos vocês pela ajuda e companheirismo de sempre nessa jornada que aqui se encerra hoje.

Agradeço ao meu irmão e orientador Tiago Antonio Rizzetti, por todos os ensinamentos, dicas, debates e projetos que contribuíram intensamente para minha formação acadêmica.

Agradeço ao meu coorientador Renato Preigschadt de Azevedo, que sempre entre uma piada e outra surgia com uma possível alternativa para problemas que surgiam no decorrer deste trabalho e me ajudou em muito chegar onde cheguei.

Muito obrigado a todos.

“A tarefa não é tanto ver aquilo que ninguém viu, mas pensar o que ninguém ainda pensou sobre aquilo que todo mundo vê. ”

Arthur Schopenhauer

RESUMO

Monografia

Curso Superior de Tecnologia em Redes de Computadores
Universidade Federal de Santa Maria

OTIMIZAÇÃO DO SISTEMA DE MÁQUINAS VIRTUAIS DO CTISM UTILIZANDO LXC/LXD

AUTOR: BRUNO AUGUSTO RIZZETTI
ORIENTADOR: TIAGO ANTONIO RIZZETTI
COORIENTADOR: RENATO PREIGSCHADT DE AZEVEDO

Data e Local da defesa: Santa Maria, 11 de dezembro de 2015.

As técnicas de uso de virtualização, não são novidades considerando os dias atuais, e são um campo vastamente explorado para os profissionais de TI (tecnologia da informação). Em ambientes de estudo que visam TI, para execução tarefas rotineiras como instalação de softwares, configurações de serviço entre outras, é necessária permissão de usuário administrador do sistema. Nesse contexto, a utilização de laboratórios de informática de uso comum se tornam inviáveis, pois, no momento em que qualquer usuário tenha acesso administrador às máquinas, os demais usuários destas máquinas são postos em risco. Visto que, assim se torna possível a instalação de softwares maliciosos, que ameacem a segurança dos demais usuários deste mesmo laboratório. Por este motivo foi desenvolvido o sistema de máquinas virtuais do CTISM, onde através de um modelo de virtualização conhecido como virtualização por *containers* aparentemente resolve este problema. Porém, com a grande número de alunos que ingressam nos cursos técnicos e tecnológicos pertencentes ao CTISM, o sistema atual não possui capacidade de comportar tal número de usuários. Aqui surge uma proposta de otimização do sistema implementado, que forneça a capacidade suficiente para atender a demanda que hoje não consegue ser atendida. Faz parte também dessa proposta prover novas funcionalidades e permitir expansão futura, agregando escalabilidade ao sistema.

Palavras-chave: Sistemas Computacionais, Máquinas Virtuais, Virtualização por C*ontainers*, Escalabilidade, Segurança.

ABSTRACT

Monograph

Technology in Computer Networks Degree
Federal University of Santa Maria

OPTIMIZATION OF THE VIRTUAL MACHINE CTISM SYSTEM USING LXC / LXD

AUTHOR: BRUNO AUGUSTO RIZZETTI
ADVISOR: TIAGO ANTONIO RIZZETTI
CO ADVISER: RENATO PREIGSCHADT DE AZEVEDO

Defense Place and Date: Santa Maria, December 11, 2015.

Use virtualization techniques are nothing new considering the present day, and are a widely explored field for IT professionals (information technology). In a study aimed at IT environments, to perform routine tasks such as software installation, service settings among others, permission is required from the system administrator user. In this context, the use of computer labs in common use become unviable because at the time that any user has administrator access to the machine, other users of these machines are put at risk. Since thus it becomes possible to install malicious software, which threaten the safety of other members of this same laboratory. For this reason, the system was developed CTISM the virtual machine, where through a virtualization model known as virtualization containers apparently solves this problem. However, with the high number of students in the technical and technological courses belonging to CTISM, the current system lacks the capacity to conduct such number of users. Here comes a system optimization proposal implemented, which provides sufficient capacity to meet the demand that today cannot be met. It is also part of this proposal to provide new functionality and enable future expansion, adding scalability to the system.

Keywords: Computer Systems, Virtual Machines, Virtualization Containers, Scalability, Security.

LISTA DE ILUSTRAÇÕES

LISTA DE ABREVIATURAS E SIGLAS

API	*Application Programming Interface*
CLI	*Command Line Interface*
CSS	*Cascading Style Sheets*
CTISM	Colégio Técnico Industrial de Santa Maria
DHCP	*Dynamic Host Configuration Protocol*
HTML	*HyperText Markup Language*
HTTP	*Hypertext Transfer Protocol*
IP	*Internet Protocol*
LDAP	*Lightweight Directory Access Protocol*
LXC	*Linux Containers*
LXD	*Linux Daemon*
PHP	*Hypertext Preprocessor*
SSH	*Secure Shell*
TI	Tecnologia da Informação
VMM	*Virtual Machine Monitor*
VMS	*Virtual Machines System*
VPN	*Virtual Private Network*
WEB	*World Wide Web*

SUMÁRIO

1 INTRODUÇÃO

O Colégio Técnico Industrial de Santa Maria (CTISM), constituí uma gama de cursos técnicos e tecnológicos que implicam na utilização de sistemas computacionais, onde estes auxiliam no aprendizado de seus discentes. Deste modo foi criado o sistema de máquinas virtuais do CTISM, que torna possível que cada discente, docente ou até mesmo funcionários desde que estes possuam cadastro na instituição, criem ambientes virtuais que auxiliarão em suas atividades. Atualmente é vastamente utilizado pelo curso de Tecnologia em Redes de Computadores – CTISM. O sistema de autenticação utilizado para acesso ao sistema é realizado através dos registros presentes em um servidor LDAP (*Lightweight Directory Access Protocol*).

Para a realização de tarefas rotineiras como instalação de softwares, configuração de serviços, alteração de interfaces de rede entre outros, essa implementação contribui de forma proativa com manutenção dos laboratórios de informática. Cada usuário deve possuir acesso total a suas máquinas, de forma a conseguir realizar tais tarefas que demandam permissões de administrador. Nesses termos, permitir que o usuário tenha esse tipo de acesso às máquinas físicas dos laboratórios de informática de uso comum é inviável, visto que, no momento em que qualquer usuário tenha acesso administrador, é colocado em risco os demais usuários destas máquinas. Pois, dessa forma é concedido a possibilidade de instalar *softwares* maliciosos, que venham a ameaçam a segurança dos demais usuários deste mesmo laboratório, como por exemplo *keyloggers* (SAGIROGLU, 2009).

O sistema de máquinas virtuais existente no CTISM conhecido como VMS (*Virtual Machine System*) aparentemente resolve esta questão, pois assim, cada máquina criada é de uso restrito de seu dono e destacada das demais. Nesse contexto, é implementado um modelo de virtualização que faz uso de *containers* isolados, tanto entres as demais máquinas virtuais criadas quanto ao computador hospedeiro. No momento presente o sistema funciona a base de um *software* denominado LXC (*LinuX Containers*), que fornece uma virtualização de forma nativa para sistemas operacionais Linux tal como o próprio nome indica, através deste é possível instanciar *containers* que fazem uso também de sistemas Linux, criando assim as máquinas virtuais de cada utilizador. No Capítulo 2 será tratado sobre os tipos de virtualizações existentes, assim como o motivo da escolha da virtualização por *containers*.

1.1 **Sistema Atual (VMS v1)**

Atualmente, o VMS versão 1 se encontra executando em um servidor do próprio CTISM, onde seus recursos de armazenamento, memória, capacidade de processamento assim como parte do *kernel* do sistema operacional Linux utilizado, são compartilhados entre todas máquinas virtuais ativas nele instanciadas. Faz parte também deste sistema uma interface WEB que permite o gerenciamento das máquinas virtuais de cada usuário de forma remota. Esta interface é composta de uma página WEB que permite aos usuários já cadastrados no sistema LDAP já existente na rede do CTISM, fazer *login* e fazer a gerência de suas máquinas de forma remota. Neste contexto, é permitido aos usuários a gerência de seus *containers* até mesmo de fora da rede do CTISM, em virtude desta interface WEB possuir um acesso via protocolo HTTP (*HyperText Transfer Protocol*), sendo acessível de qualquer localidade desde que disponha de acesso à internet. Esta interface foi desenvolvida em HTML (*HyperText Markup Language*), CSS (*Cascading Style Sheets*) e faz uso da linguagem de programação PHP (*Hypertext Preprocessor*). Neste o PHP é o responsável por realizar as chamadas de sistema, que por sua vez executam *scripts* desenvolvidos na linguagem Shell Script, que são os responsáveis pela gerência permitida aos usuários. Estas chamadas de sistema ou *system calls* atuam na busca de informações referentes aos *containers* de cada usuário, como nomes das máquinas criadas, os endereços IP (Internet Protocol) vinculados a cada máquina, criar ou remover máquinas, ligar ou desligar máquinas já existentes, entre outras funcionalidades.

O modelo implementado também fornece aos usuários um acesso remoto via SSH (*Secure Shell*), através de um emulador de terminal chamado Gate One[1], onde este é acessível por meio da internet, sendo necessário para seu acesso apenas conectar na página WEB específica designada a este. O software se encontra na mesma faixa de endereço IP das máquinas virtuais, deste modo é possível o acesso SSH, e por meio deste é permitido executar tarefas rotineiras como instalar/remover programas, realizar o monitoramento de serviços que estão em execução na máquina e demais tarefas que sejam possíveis de serem realizadas através de interface CLI (*Command Line Interface*) utilizado pelo SSH.

[1] Maiores informações em http://liftoffsoftware.com/Products/GateOne.

1.2 **Problema**

O VMS na sua primeira versão deixa a desejar em alguns aspectos possuindo alguns pontos negativos, onde estes causam indisponibilidade do sistema e perda significativa de desempenho. Isso se deve à falta de escalabilidade do VMS v1 e da crescente demanda de usuários que utilizam o sistema. Em função disso, faz-se necessário métodos de expansão futura do sistema, de modo a causar o menor impacto possível aos usuários que já possuem máquinas virtuais criadas neste. Nesses termos, é de significativa importância um meio de tornar este sistema escalável, realizando o melhor aproveitamento possível das tecnologias já utilizadas.

1.3 **Objetivos**

Este trabalho tem como objetivo uma proposta de melhoria fazendo uso de ferramentas livres, atualizadas e complementares, sem gerar impactos aos usuários. E que forneça ao VMS a escalabilidade necessária para atender ao seu público crescente de usuários.

1.3.1 **Objetivos específicos**

- Através das ferramentas escolhidas proporcionar um maior gerenciamento, de modo mais intuitivo e simplificado para o administrador de todo o sistema, além de prover funcionalidades não existentes no VMS v1.
- Não perder nenhuma das funcionalidades presentes na versão pregressa do sistema, a melhoria manterá todo sistema funcionando de forma equivalente ao VMS v1, com pequenos ajustes para serem implementados, porém após a otimização aqui proposta teremos algumas funcionalidades que serão adicionadas a este sistema.
- Adicionar critérios como: escalabilidade, uma API-REST (*Application Program Interface – Representational State Transfer*) para utilização via interface WEB,

um menu completo e amigável para o administrador, organização de perfis de usuários para um melhor aproveitamento do sistema como um todo, entre outros.

- Para a escolha dos softwares para elaboração deste trabalho, foi visado principalmente reduzir ao máximo os impactos no sistema existente, a possibilidade de incluir funcionalidades sem prejudicar as incluídas até o momento.
- Para elaboração deste trabalho foi utilizado novamente a ferramenta LXC, porém trabalhando em conjunto com uma ferramenta LXD que é um *daemon* que fica executando no sistema em background. Esta por sua vez, diferentemente do que pode ser pensado, não têm como objetivo substituir o LXC, mas sim complementá-lo. Dessa forma, permitindo ir muito além do que o LXC convencional permite. No capítulo 3 será discutido mais detalhes sobre as ferramentas utilizadas.

1.4 **Estrutura do trabalho**

No Capítulo 2, é apresentado conceitos e técnicas utilizadas de virtualização. No Capítulo 3, é apresentado as funcionalidades e conceitos sobre as ferramentas escolhidas para implementação deste trabalho. No capítulo 4, é tratado as características principais do modelo proposto, tal como funcionalidades providas através das ferramentas. No capítulo 5, é apresentado a estruturação dos modelos versão atual e versão proposta. No capítulo 6, é mostrado como o sistema foi implementado e como ficou organizado o novo sistema. No capítulo 7, exibido uma análise dos resultados obtidos através de testes do novo sistema. No capítulo 8, é apresentado as considerações finais sobre os objetivos deste trabalho.

2 VIRTUALIZAÇÃO

As técnicas de uso de virtualização, não são nenhum tipo novidade considerando os dias atuais, e é um campo vastamente explorado para os profissionais de TI (tecnologia da informação). Uma máquina virtual é uma camada de *software*, que fornece ao usuário um ambiente completo, muito semelhante ao que uma máquina física proporciona. Cada máquina virtual, assim como uma máquina física, possui seu próprio sistema operacional, aplicativos e serviços de rede (CARISSIMI, 2008).

Surge o conceito de virtualização, como uma alternativa de baixo custo para prover confiabilidade, escalabilidade e isolamento a alguns sistemas. Contudo, na atualidade seus interesses vão além disso, englobando características como segurança, confiabilidade, adaptabilidade, balanceamento de carga e suporte a aplicações legadas (MATTOS, 2008).

Nos dias atuais, a evolução nos sistemas computacionais desde a última década, principalmente no que se refere ao *hardware* utilizado nos computadores é evidente. A virtualização pode ser utilizada como um meio de aproveitar de forma mais eficaz os recursos disponíveis, sem que haja subutilização e desperdício de recursos computacionais.

Existem diferentes formas de virtualizar sistemas, onde dentre esses se destacam a virtualização completa, a paravirtualização e a virtualização por *containers*. As seções a seguir, destina-se a descrição mais específica de cada um dos tipos citados anteriormente.

2.1 **Virtualização Total**

A técnica utilizada para realizar uma virtualização completa do sistema é fazer uso de uma simulação completa da subcamada de *hardware* para os sistemas convidados. Isto é, todos os sistemas operacionais e/ou ferramentas que são capazes de executar diretamente de um *hardware* serão capazes também de ser executados na máquina virtual que faz uso desde modelo de virtualização (DA SILVA, 2007).

Um ponto que pode ser destacado como vantagem dessa técnica, é o fato do sistema convidado ou *guest*, ser capaz de ser executado como se estivesse rodando diretamente em um

computador, ou seja, já que por meio deste, cada máquina virtual possui uma simulação completa de um *hardware* físico do sistema hospedeiro.

Na Figura 1, é possível perceber uma camada denominada Monitor de máquinas virtuais ou também conhecido como *hypervisor*.

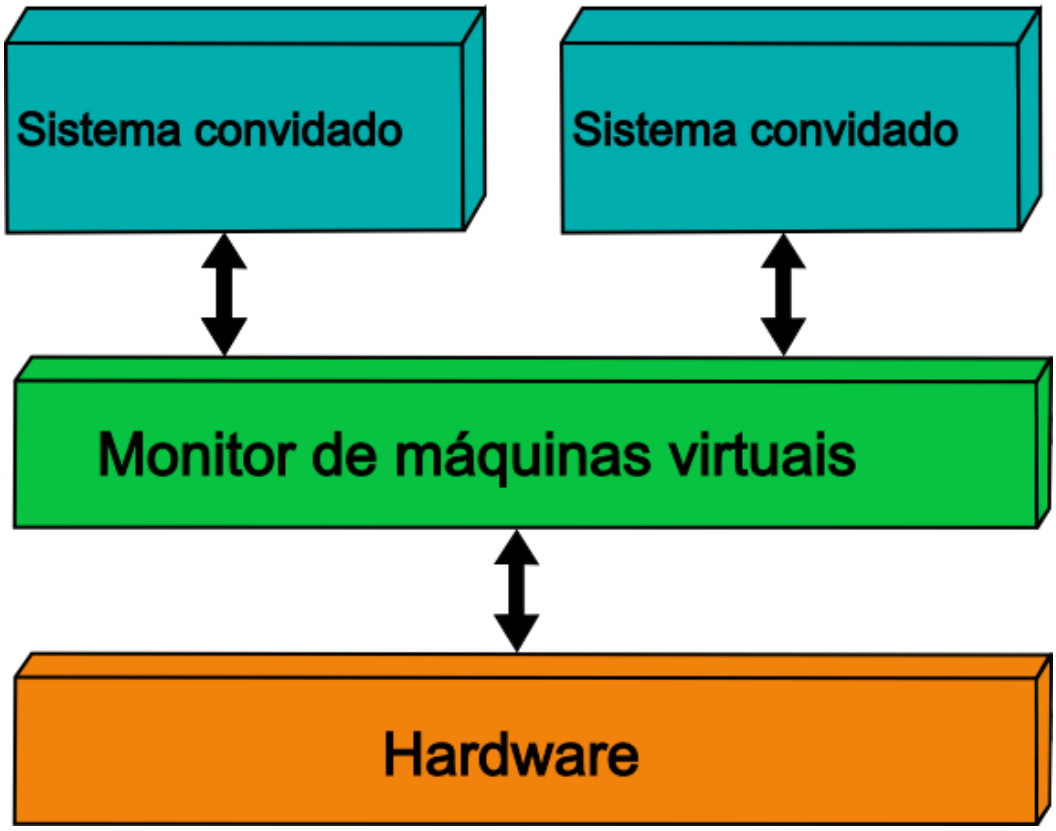

Figura 1: Representação de um sistema de virtualização total.
Fonte: Acervo Pessoal.

Hypervisor conhecido como monitor de máquinas virtuais ou até mesmo VMM (*Virtual Machine Monitor*), é uma camada de *software* que faz a interação entre o *hardware* e o sistema operacional da máquina virtual. Nesse contexto, os recursos que são acessados pela máquina virtual são controlados pelo *hypervisor*, possuindo uma instância deste para cada máquina virtual criada. Também é de responsabilidade deste fazer a gerência dos sistemas emulados da máquina hospedeira, tais como, processador, memória RAM, disco rígido, interfaces de entrada e saída, *mouse*, teclado, memória de vídeo, interfaces de rede entre outros (ROCHA, 2013).

2.2 **Paravirtualização**

É um modelo de virtualização alternativo em relação a virtualização total. Nesse modelo, é modificado o sistema operacional do convidado, de modo a realizar a chamada do *hypervisor* sempre que houver a execução de uma instrução que venha a alterar o estado do sistema. Nesse contexto, não existe mais a necessidade do *hypervisor* testar instrução por instrução, fazendo todo o sistema possuir uma interação mais eficiente diminuindo a carga do *hypervisor*, assim representando um ganho expressivo em desempenho.

Na Figura 2, pode ser observado a interação do dado sistema. Dessa forma, os sistemas convidados só se comunicam com o *hypervisor* para instruções mais complexas. Enquanto as de nível mais baixo são enviadas diretamente ao processador do *host* hospedeiro.

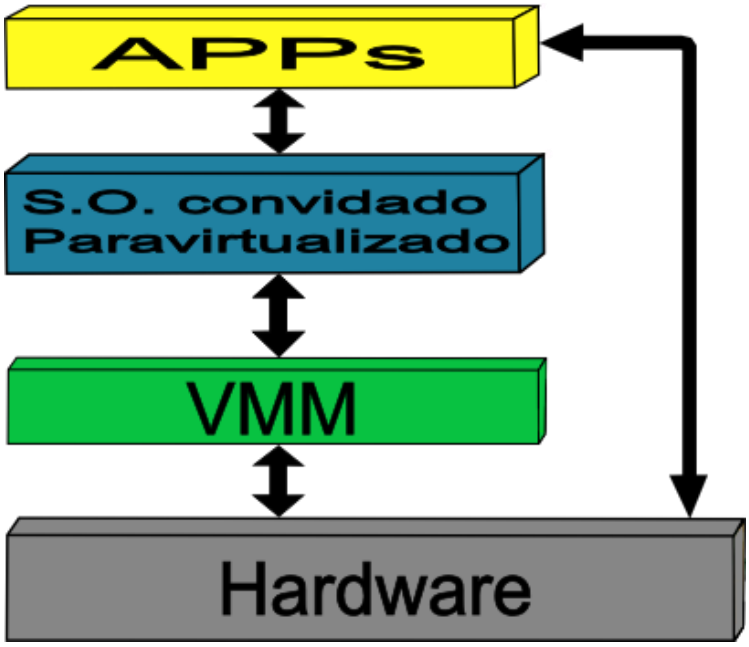

Figura 2: Esquema de um sistema de paravirtualização em arquitetura x86.
Fonte: Acervo Pessoal.

Os dispositivos de *hardware*, acessados pelas máquinas virtuais que fazem uso deste modelo, diferentemente das que utilizam virtualização completa, são acessados através dos *drivers* das próprias máquinas virtuais. Nesses termos, assim não se tornando mais necessário

a utilização de *drivers* genéricos, que acabam impedindo a utilização dos recursos de *hardware* disponível de uma forma totalmente plena.

Ainda que, esse modelo apresente aparentemente um melhor desempenho em relação a virtualização total, as tecnologias nos processadores avançaram, sendo assim criadas instruções de virtualização nos processadores Intel (IVT) e AMD (AMD-V). Com isso, favorecendo a virtualização total e superando a aparente vantagem deste modelo presente (MATTOS, 2008).

2.3 **Virtualização por Containers**

Modelo de virtualização alternativo às demais virtualizações clássicas, também é conhecido como virtualização em nível de sistema operacional. Em determinados cenários que demandam sistemas virtuais de maior desempenho, e não necessitam de sistemas operacionais diferentes. Nesse contexto, a virtualização por *containers* é a mais adequada (FERNANDES, 2010).

Este modelo emprega diferentes técnicas distintas, incluindo a emulação completa do *hardware* até a virtualização a nível de sistema operacional (*Container-based Operating System Virtualization*) (REIS, 2015). Nesse contexto, é também utilizado o compartilhamento do *kernel* do sistema operacional entre todas máquinas virtuais hospedadas ao sistema. Neste modelo, devido ao compartilhamento do *kernel*, diferente das demais técnicas de virtualização, é evitado que seja adicionada a camada de *software* (*hypervisor*) ao sistema. Dessa forma, tornando este um modelo de virtualização mais eficiente (FERNANDES, 2010).

A virtualização por *containers* fornece múltipla segurança para servidores virtuais e físicos, já que faz o compartilhamento do mesmo sistema operacional, entretanto de forma isolada, sem que um *container* possa interferir na execução de outro, ou até mesmo no sistema operacional hospedeiro (SILVA, 2012).

Na Figura 3, é representado de forma simplificada a estrutura de um sistema de virtualização que utiliza a virtualização por *containers*. Dessa forma, é demonstrado como os *containers* são totalmente isolados uns dos outros, mantendo assim a segurança contra interferências de processos entre os mesmos.

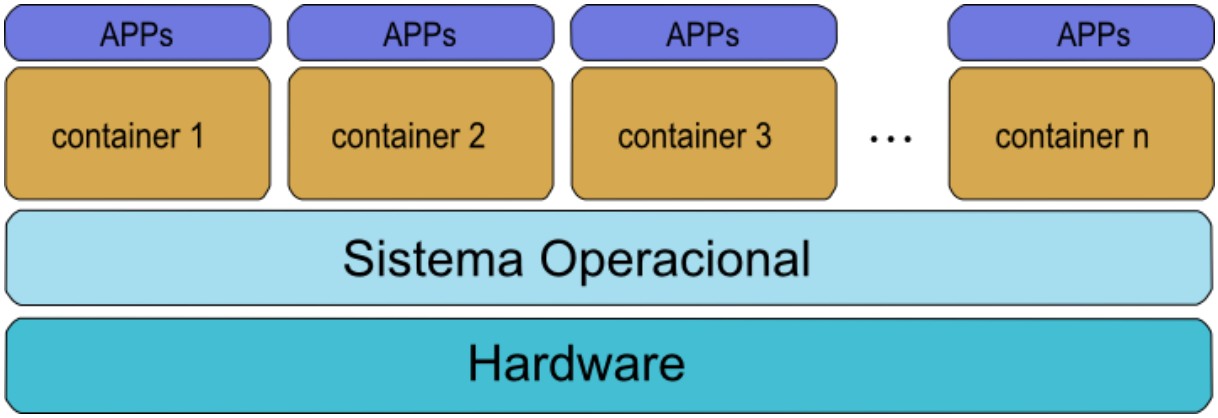

Figura 3: Esquema simplificado da estrutura de um sistema de virtualização por *containers*.
Fonte: Acervo Pessoal.

Comparando o presente modelo à virtualização tradicional, fica claro que uma aplicação fazendo o uso de *containers* demanda menos recursos, consumindo menor espaço em disco, e possuindo também um nível de portabilidade que dificilmente pode ser alcançado através de outras plataformas de virtualização (IBM.COM, 2014).

Na Figura 4, é apresentado um comparativo entre os modelos de virtualização convencionais em relação ao funcionamento do dado modelo, onde neste pode ser verificado que todas máquinas virtuais (*containers*) utilizam o mesmo sistema operacional (*kernel*). Dessa forma, também fazendo uso dos mesmos recursos de *hardware* da máquina hospedeira. Nesse contexto, é nítida a remoção da camada de *software* em relação ao modelo comparado.

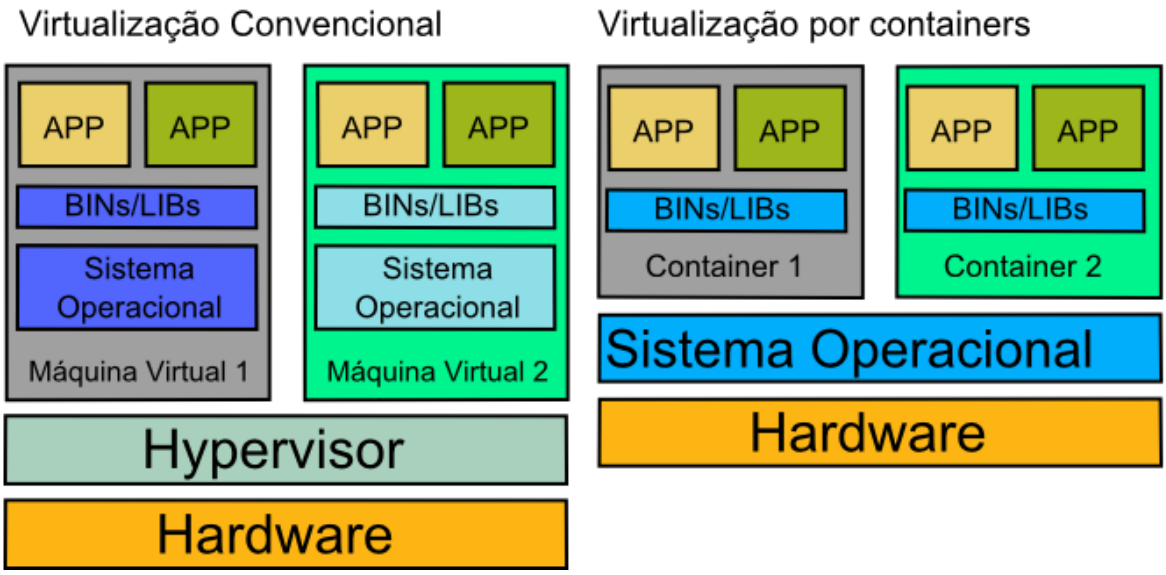

Figura 4: Comparativo de sistemas de virtualização tradicionais em relação a virtualização por *containers*.
Fonte: Acervo Pessoal.

Os processos pertencentes as máquinas virtuais, são contidos em *containers* distintos. Sendo assim, do ponto de vista do *kernel* todas as máquinas virtuais são um conjunto de processos. Já fazendo uma análise do ponto de vista da máquina virtual, os processos que estão em execução dentro desta, assemelham-se como uma execução de forma exclusiva no *kernel* (FERNANDES, 2010).

No Capítulo seguinte, serão mostradas as características sobre as ferramentas escolhidas para implementação deste trabalho, no qual estas fazem uso deste tipo de virtualização aqui discutido.

3 FERRAMENTAS

As ferramentas que serão apresentadas neste capítulo possuem como foco a virtualização por *containers*, dando continuidade ao que foi apresentado na seção 2.3 do capítulo que trata sobre Virtualização.

Os softwares escolhidos foram LXC e LXD, principalmente por sua compatibilidade com o sistema já existente e sua integração entre si. Será visto nas próximas seções o funcionamento de cada ferramenta, tal como o objetivo de cada uma destas.

3.1 **LXC**

LXC (*LinuX Containers*) é um *software* livre que fornece ao usuário as características de uma virtualização por *containers*. Nesses termos, isso é fornecido através de uma API integrada a um conjunto de ferramentas simples, que permite aos usuários que utilizam sistemas Linux, facilmente criar e gerenciar contêineres de sistema ou aplicativo (LINUXCONTAINERS.ORG).

O objetivo deste *software*, é criar um ambiente virtual o mais próximo possível de um sistema Linux padrão, porém sem a necessidade de um *kernel* separado.

LXC é composto de diversos componentes, entre eles uma biblioteca liblxc, diversas linguagens de programação (python3, lua, Go, ruby, python2 e Haskell) para sua API, um conjunto de ferramentas padrão para gerenciamento dos *containers*, entre outras.

Como pode ser visto na Figura 5, os utilizadores possuem por padrão, após a instalação do LXC, diversos *scripts* que fornecem um gerenciamento de modo simplificado aos seus *containers*. Dessa forma, a criação, exclusão, informações, parada, inicialização e demais tarefas de gerência dos *containers* são simples e intuitivas para qualquer usuário.

```
root@odin:~# lxc
lxc                 lxc-freeze          lxc-setuid
lxc-attach          lxc-info            lxc-shutdown
lxc-backup          lxc-kill            lxc-start
lxc-cgroup          lxc-list            lxc-start-ephemeral
lxc-checkconfig     lxc-ls              lxc-stop
lxc-checkpoint      lxc-monitor         lxc-unfreeze
lxc-clone           lxc-netstat         lxc-unshare
lxc-console         lxc-ps              lxc-version
lxc-create          lxc-restart         lxc-wait
lxc-destroy         lxc-restore
lxc-execute         lxc-setcap
```

Figura 5: *Scripts* providos através da instalação padrão do LXC.
Fonte: Acervo Pessoal.

3.2 **LXD**

LXD (*Linux Containers Daemon*) como o próprio nome indica é um *daemon*, ou seja, um programa ou processo que roda continuamente no sistema, com a finalidade de manipular e executar serviços recorrentes do sistema operacional. Nesse contexto, sua definição pode ser abstraída como um processo que está rodando em *background* ao sistema operacional, e é utilizado sempre que necessário.

O projeto LXD é um dos projetos pertencentes ao projeto LXC, que foi fundado e atualmente liderado pela Canonical Ltd. e Ubuntu com contribuições de várias outras empresas e contribuintes individuais. O LXD pode ser entendido como uma ferramenta complementar ao LXC, desde sua elaboração não foi intenção este ser lançado como uma substituição ao LXC. Tanto que, este foi construído em cima da ferramenta LXC. Dessa forma, este *daemon* surgiu como uma nova experiência para os usuários do LXC. É composto especificamente de três componentes: um *daemon* de todo sistema (LXD), um cliente via linha de comando (LXC) e um *plugin* chamado nova-compute-LXD (LINUXCONTAINERS.ORG).

O *plugin* chamado nova-compute-LXD é utilizado para realizar a integração do sistema LXD com OpenStack. O *software* OpenStack é utilizado para criação de nuvens privadas e/ou públicas. Nesse contexto, a utilização deste *plugin* permite que os *hosts* LXD sejam tratados como nós computacionais executando cargas de trabalho em *containers* ao invés de máquinas virtuais (INSIGHTS.UBUNTU.COM, 2015) (OPENSTACK.ORG).

O *daemon* exporta uma API-REST, que provê a comunicação tanto localmente, como através da rede, para busca de informações de *containers* e/ou na execução de comandos remotos em *containers* ou servidores LXC (LINUXCONTAINERS.ORG).

O cliente via linha de comando (LXC), é uma ferramenta projetada para ser muito simples e muito poderosa ao mesmo tempo, para a gerência dos *containers*. Por meio desta, é possível realizar diversas tarefas com poucas linhas de comandos. Além da simplicidade fornecida, esta também dispõe de um menu de ajuda intuitivo que pode ser sempre exibido, apenas através da passagem do parâmetro "-h" após o comando "lxc".

Na Figura 6, pode ser visto o menu de ajuda fornecido pela interface CLI, onde tudo é muito intuitivo até mesmo para os usuários mais inexperientes em relação ao LXC convencional.

```
zeus@Olimpo ~ $ lxc -h
Usage: lxc [subcommand] [options]
Available commands:
        config      - Manage configuration.
        copy        - Copy containers within or in between lxd instances.
        delete      - Delete containers or container snapshots.
        exec        - Execute the specified command in a container.
        file        - Manage files on a container.
        help        - Presents details on how to use LXD.
        image       - Manipulate container images.
        info        - List information on containers.
        launch      - Launch a container from a particular image.
        list        - Lists the available resources.
        move        - Move containers within or in between lxd instances.
        profile     - Manage configuration profiles.
        publish     - Publish containers as images.
        remote      - Manage remote LXD servers.
        restart     - Changes one or more containers state to restart.
        restore     - Set the current state of a resource back to what it was when it was snapshotted.
        snapshot    - Create a read-only snapshot of a container.
        start       - Changes one or more containers state to start.
        stop        - Changes one or more containers state to stop.
        version     - Prints the version number of LXD.

Options:
  --all             Print less common commands.
  --debug           Print debug information.
  --verbose         Print verbose information.

Environment:
  LXD_CONF          Path to an alternate client configuration directory.
  LXD_DIR           Path to an alternate server directory.
```

Figura 6: Menu de ajuda obtido através do cliente via linha de comando (LXC).
Fonte: Acervo Pessoal.

A utilização do LXD fornece algumas características específicas, dentre essas as maiores são:

- Segurança: Permite uma série de restrições tanto em nível de *software*, quanto a nível de *hardware* que podem ser aplicadas aos containers. Restrições como número de interfaces de rede disponíveis, memória RAM, número de CPUs entre outras.
- Escalabilidade: Permite a migração de *containers* entre servidores LXD. Dessa forma, pode ser usado um conjunto de servidores LXD onde os *containers* podem ser migrados entre estes. Possui migração tanto para *containers* desligados quanto em tempo real de execução, sendo possível migrar um *container* em execução para outro servidor LXD, sem a necessidade de ter que congelar o mesmo.
- Menu com uma interface amigável: Simples e limpa, fornece para o utilizador uma nítida facilidade de trabalho utilizando a linha de comando.
- Imagens baseadas: Não utiliza mais os *templates* como eram utilizados pelo LXC nativo, agora faz uso de imagens de sistemas operacionais que podem ser baixadas através do LXD, e então após o *download*, estas podem ser disponibilizadas para os novos containers que serão criados (LINUXCONTAINERS.ORG).

3.2.1 Imagens de disco para *containers*

Os sistemas operacionais utilizados nos *containers* são disponibilizados por meio da ferramenta LXD, através de um comando de importação que passa por como parâmetro o nome da imagem desejada. Assim, é feito o *download* da imagem de um sistema operacional específico para essa finalidade. Após a conclusão deste é possível criar *containers* vinculados a imagem descarregada.

Como discutido anteriormente, a virtualização por *containers* utiliza o compartilhamento do *kernel* do sistema operacional hospedeiro com os *containers*, porém isso não limita a utilização apenas do sistema operacional idêntico ao do hospedeiro. A imagem de sistema operacional que será disponibilizada para os *containers*, pode ser de diferentes distribuições Linux, como por exemplo, Ubuntu, Debian, CentOS entre outras. Além disso, as imagens podem ser criadas ou personalizadas de acordo com a necessidade, com *softwares* específicos já instalados ou arquivos já disponíveis por padrão nos *containers* vinculados a esta.

Basicamente o que compõe essas imagens, é um diretório raiz chamado *rootfs* que contém todo diretório principal dos sistemas operacionais baseados em Linux. Vale ressaltar

que, todas imagens devem possuir compatibilidade com os recursos do *kernel* do sistema operacional hospedeiro, já que este será compartilhado com os *containers*. Após a importação da imagem escolhida, pode ser atrelado a esta um *alias*. Desse modo, a chamada para criação de novos *containers* será simplificada, sendo necessário passar como parâmetro para o comando de criação de *containers* apenas o seu alias definido.

4 DESENVOLVIMENTO

Este Capítulo, será trata de assuntos referentes ao VMS versão 2 em relação ao VMS versão 1. Dentre os assuntos aqui tratados, serão descritas as principais funcionalidades providas à um sistema em relação ao outro.

4.1 **Modelo Conceitual**

Considerando a adoção das ferramentas aqui propostas, o VMS v1 ganharia novas funcionalidades sem perder suas características principais. Dessa forma, novos recursos estariam disponíveis para o administrador do sistema de uma forma mais intuitiva, tornando assim uma implementação mais produtiva e rápida.

Com o sistema proposto (VMS v2), seria reduzido também em grande parte os problemas de configuração dos *containers*, onde até então estes são criados e configurados por *scripts* criados pelo administrador do sistema e porventura acabavam corrompendo os arquivos de configurações de alguns *containers* criados. Nesse contexto, fazendo uso do novo sistema esses problemas seriam minimizados, uma vez que, praticamente toda configuração de *containers* fica a cargo das ferramentas e não do administrador.

Os seguintes tópicos, demonstram o modo com que o VMS v1 trabalha e algumas das características/funcionalidades de maior destaque que serão providas com adoção dessas ferramentas.

4.1.1 Endereçamento IP para novos *containers*

No modelo atual, o sistema de distribuição de endereços IP (*Internet Protocol*) é realizado através de *scripts*, desenvolvidos de forma manual pelo administrador do sistema.

Já com a adoção do novo sistema, o mesmo sistema de distribuição de endereços é realizado automaticamente pelas ferramentas. No qual, é necessário somente ao administrador

do sistema, realizar a configuração do arquivo "/etc/default/lxc-net" de acordo com a necessidade que o sistema visa atender. Permitindo através desse arquivo realizar a distribuição dos endereços até mesmo por DHCP (*Dynamic Host Configuration Protocol*) retirando possíveis falhas se este fosse criado manualmente.

Na Figura 7, é exibido o arquivo com configuração de rede para novos *containers*, este permite configurações como o DHCP mencionado anteriormente. Nestes termos, é possível também vincular as interfaces de rede do *container* em modo *brigde* com alguma das interfaces do servidor.

```
root@server-lxc:~# cat /etc/default/lxc-net
# This file is auto-generated by lxc.postinst if it does not
# exist.  Customizations will not be overridden.
# Leave USE_LXC_BRIDGE as "true" if you want to use lxcbr0 for your
# containers.  Set to "false" if you'll use virbr0 or another existing
# bridge, or mavlan to your host's NIC.
USE_LXC_BRIDGE="true"
#USE_LXC_BRIDGE="false"

# If you change the LXC_BRIDGE to something other than lxcbr0, then
# you will also need to update your /etc/lxc/default.conf as well as the
# configuration (/var/lib/lxc/<container>/config) for any containers
# already created using the default config to reflect the new bridge
# name.
# If you have the dnsmasq daemon installed, you'll also have to update
# /etc/dnsmasq.d/lxc and restart the system wide dnsmasq daemon.

LXC_BRIDGE="lxcbr0"
LXC_ADDR="10.99.0.1"
LXC_NETMASK="255.255.192.0"
LXC_NETWORK="10.99.0.0/18"
LXC_DHCP_RANGE="10.99.0.2,10.99.63.254"
LXC_DHCP_MAX="16381"
```

Figura 7: Arquivo lxc-net que contém as configurações de rede para novos *containers*.
Fonte: Acervo Pessoal.

4.1.2 Criação de *profiles* de utilizadores

A possibilidade de criação de perfis para *containers* é outra funcionalidade que até então, no VMS v1 não está presente. Através deste é possível a criação de perfis personalizados para cada grupo de utilizadores, tanto como recursos de *hardware* quanto em dispositivos disponíveis para estes.

A utilização de perfis de usuários permite designar quais *containers* serão vinculados a qual perfil. Em algumas ocasiões, em ambientes acadêmicos se faz necessário, por exemplo, diversas interfaces de rede, uma quantidade de memória RAM específica, uma quantidade específica de armazenamento, entre outros. Utilizando-se desses perfis isso se torna uma tarefa fácil, pois através destes o sistema possuirá uma maior facilidade na criação dos *containers*. Dessa forma, pode-se criar perfis específicos a cada necessidade e vincular a este perfil quantos *containers* forem necessários, assim como criar *containers* associados a estes perfis.

Os perfis são basicamente arquivos-texto, onde nestes é possível definir quais configurações de *hardware* ou dispositivos farão parte deste perfil. Possuem basicamente três campos, são eles o nome do perfil, as configurações de *hardware* do perfil e as configurações de dispositivos do perfil como interfaces de rede, por exemplo.

Considerando hipoteticamente, que um grupo de alunos necessitem de máquinas virtuais que possuam cinco interfaces de rede e uma memória RAM de 256 megabytes para experimentos em aula. Através da criação de um perfil com as características de *hardware* citadas, é possível que diversos *containers*, já criados ou não, sejam vinculados a este perfil. Dessa forma, seguindo o exemplo citado todo grupo de alunos terá os mesmos recursos de hardware necessário disponível para seus *containers*.

Na Figura 8, é observado como é representado as informações do perfil exemplo aqui exposto, pode ser observado o nome do perfil alunos, como primeira linha do arquivo, logo após o nome é identificado um limite de memória RAM de 256 megabytes e, por fim, é mostrado as interfaces de rede disponíveis para *containers* vinculados a este perfil.

```
root@server-lxc:~# lxc profile show alunos
name: alunos
config:
  limits.memory: 256M
devices:
  eth0:
    nictype: bridged
    parent: lxcbr0
    type: nic
  eth1:
    nictype: bridged
    parent: lxcbr0
    type: nic
  eth2:
    nictype: bridged
    parent: lxcbr0
    type: nic
```

Figura 8: Exibindo o perfil alunos com suas respectivas configurações.
Fonte: Acervo Pessoal.

4.1.3 API-REST para obtenção de informações sobre containers

No VMS v1, existe uma interface web no qual os usuários podem fazer *login* e verificar o *status* de seus *containers*, além de ligar/desligar, remover entre outros. Todas estas operações são realizadas através de chamadas de sistema, ou seja, se conectam ao servidor, dentro do servidor executam um script para obtenção dos dados desejados, após a obtenção da resposta esta é encaminhada para o servidor WEB que exibe para o usuário as informações requisitadas.

Na atualidade todo esse processo de obtenção de dados é realizado de forma manual, ou seja, foram desenvolvidos *scripts* para que todas estas operações sejam possíveis. Outro diferencial da adoção da proposta aqui exposta, é que as ferramentas já fornecem uma API-REST que facilita estas operações. Nesse contexto, é necessário ao servidor WEB, caso rode em outro servidor que não o próprio servidor de máquinas virtuais, possuir certificados para realizar a comunicação com o servidor LXC. Dessa forma, a comunicação é possível entre estes, e pensando nisso foi desenvolvida uma API na linguagem PHP que faz uso da API-REST fornecida pelo LXD, onde esta conta com um menu dos comandos passíveis de serem executados no servidor LXC e traz resultados no formato JSON que podem ser facilmente tratados para a exibição aos usuários do sistema.

Para obter informações referentes ao servidor e seus *containers* via esta API, existe uma lista[2] de parâmetros de consulta nesta busca de informações. Estes podem ser relativos a *containers*, perfis, certificados, imagens de sistema operacionais utilizados para os containers, entre outros.

[2] Esta lista de parâmetros pode ser consultada no GitHub da própria API, disponível em: https://github.com/lxc/lxd/blob/master/specs/rest-api.md

4.1.4 Possibilidade de migração entre servidores LXC/LXD

O sistema existente ainda não possui a possibilidade de migração entre servidores LXC, até porque é utilizado apenas um servidor para essa finalidade no momento. Adotando a otimização aqui proposta isso se torna possível, através de *sockets* UNIX as máquinas que desempenham o papel de servidores LXC podem se comunicar. Dessa forma, começa a se tornar viável a migração de *containers* criados a partir de um servidor "LXC-1" para "LXC-2".

A migração de *containers* entre servidores LXC se torna possível através de duas maneiras. Através da ferramenta CRIU ou através de perfis *migratable* que já estão presentes por padrão na instalação do LXD (INSIGHTS.UBUNTU.COM).

4.1.4.1 Migração utilizando a ferramenta CRIU

CRIU é uma ferramenta CR (*checkpoint/restart*), ou seja, consiste em criar um *snapshot* suspendendo o *container* que está em execução e salvando seu estado de memória para ser resumido mais tarde. Após a criação deste realizar a migração para outro servidor e resumir o estado do *container* migrado (CRIU.ORG). A forma de migração citada é uma outra alternativa para realizar a migração de *containers*, porém o foco deste trabalho visa utilizar o LXD para realizar a migração de *containers* entre servidores LXC.

4.1.4.2 Migração utilizando perfil *migratable*

A instalação padrão do LXD fornece perfis com configurações padrões já por *default*, um destes perfis é o perfil denominado por padrão de *migratable*. Nesse contexto, através da vinculação deste perfil ao *container* "alvo" é possível realizar uma migração entre os servidores LXC, lembrando que o perfil *migratable* deve ser existente em ambos servidores LXC e com as mesmas configurações, tanto de sistema operacional quanto de configuração do próprio perfil.

Vale ressaltar, que atualmente essa forma de migração não funciona em sistemas que fazem uso de sistemas de inicialização systemd (INSIGHTS.UBUNTU.COM, 2015). Deste modo, ambos os servidores LXC, não devem utilizar sistemas operacionais que façam uso de sistemas de inicialização systemd. Alguns exemplos de distros Linux que fazem uso deste sistema de inicialização são, Fedora, Gentoo, Debian entre outras.

Outro ponto que deve ser analisado para uso desta técnica de migração, é que deve ser removido o sistema de arquivos LXCFS que é instalado automaticamente na instalação do LXC. Isso devido a biblioteca do LXC chamada liblxc não suportar migração da configuração de montagem o qual o LXCFS utiliza. A empresa responsável ainda está trabalhando sobre essa questão para criar um *patch* que resolva este problema (INSIGHTS.UBUNTU.COM, 2015).

Partindo do pressuposto que ambos servidores estão utilizando sistemas operacionais não systemd, as mesmas versões de LXC/LXD com perfis de migração idênticos e com o LXCFS removido. Os servidores estão aptos a realizarem a migração de *containers*.

5 PROPOSTA DE UM MODELO VMS

5.1 **Modelo pregresso (VMS v1)**

O VMS v1 é composto de uma interface WEB que faz a interação com os usuários, onde é por meio desta que a execução de comandos no servidor LXC se torna possível. Nesse contexto, é através desta que as operações relativas aos *containers* são enviadas ao servidor totalmente transparente aos usuários.

Na Figura 9, é demonstrado de forma simplificada a interação entre a interface WEB, chamada de *bash scripts* dentro do PHP e execução destes no servidor LXC.

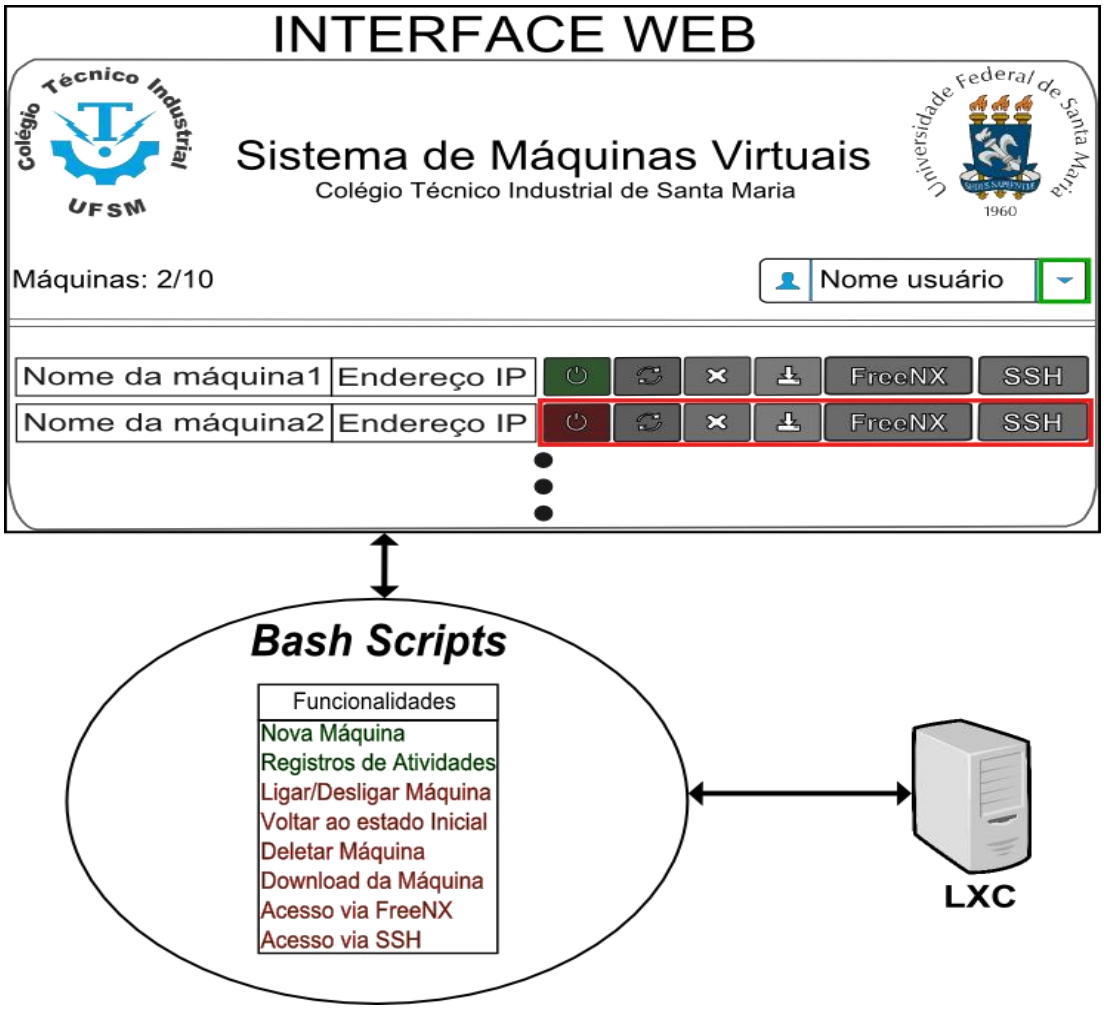

Figura 9: Estrutura de interação da interface WEB com servidor LXC no VMS v1.
Fonte: Acervo Pessoal.

Conforme a Figura 9, pode ser observado grifos no objeto interface WEB onde estes representam as funcionalidades que estão à disposição para o usuário. Dentre estas funcionalidades, algumas são providas por *softwares* terceiros, como por exemplo acesso via FreeNX e SSH. Para o usuário utilizar interface gráfica nos *containers* a funcionalidade FreeNX fornece um arquivo de configuração de extensão "nxs" que por sua vez provém a *softwares* como NoMachine um acesso via interface gráfica. Dessa forma, deixando os *containers* com uma interface mais amigável aos usuários.

O mecanismo de comunicação entre a interface WEB e o servidor LXC na versão 1 do VMS, é dado totalmente através de chamadas de sistema, agindo de forma bidirecional tanto para busca de informações sobre *containers*, quanto na execução de comandos no servidor LXC. Nesse contexto, a busca de qualquer informação sobre determinado *container* é feita por meio de *scripts* que serão executados no servidor LXC.

Em conformidade com a Figura 10, pode ser observado o funcionamento da troca de mensagens entre interface WEB e servidor LXC. Desse modo, cada funcionalidade provida pela interface desencadeia o envio de *scripts* ao servidor onde neste é executado o *script* e enviado o retorno novamente para interface WEB.

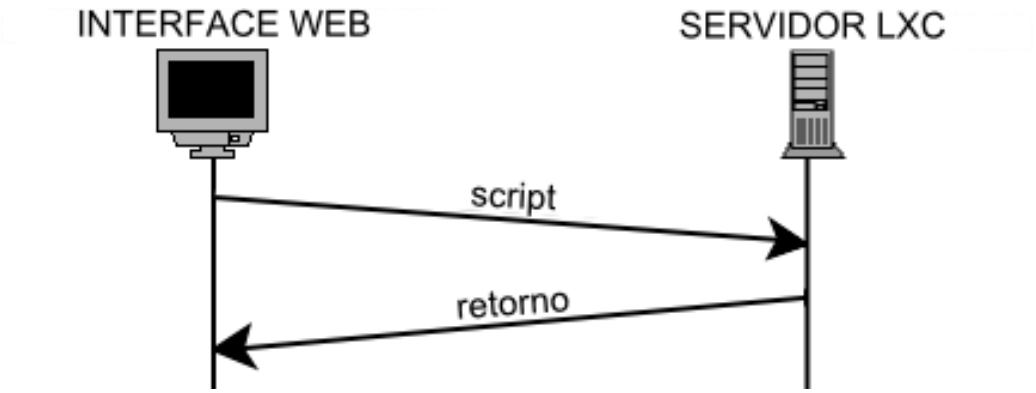

Figura 10: Diagrama de comunicação entre interface WEB e servidor LXC no VMS v1.
Fonte: Acervo Pessoal.

Pode ser citado como exemplo desta comunicação a ação de ligar uma máquina virtual, onde clicamos no botão ligar na máquina virtual, neste momento um comando de requisição para ligar aquele determinado *container* é enviado ao servidor LXC e fica aguardando um retorno deste. No servidor o *script* é executado e o *container* será iniciado e rodará. Após isso

o servidor retorna que o *container* está rodando para interface, que após o recebimento da resposta ajusta a cor do botão de ligar e desligar conforme o *status* da máquina virtual.

5.2 **Modelo Proposto (VMS v2)**

Na Figura 11, é demonstrado como se fará a interação da interface WEB com o(s) servidor(es) LXC. Pode ser observado também, que na própria interface WEB é possível incluir funcionalidades que até então não estavam presentes como criação e restauração de *snapshots* dos *containers*, tal como a adição de mais servidores LXC.

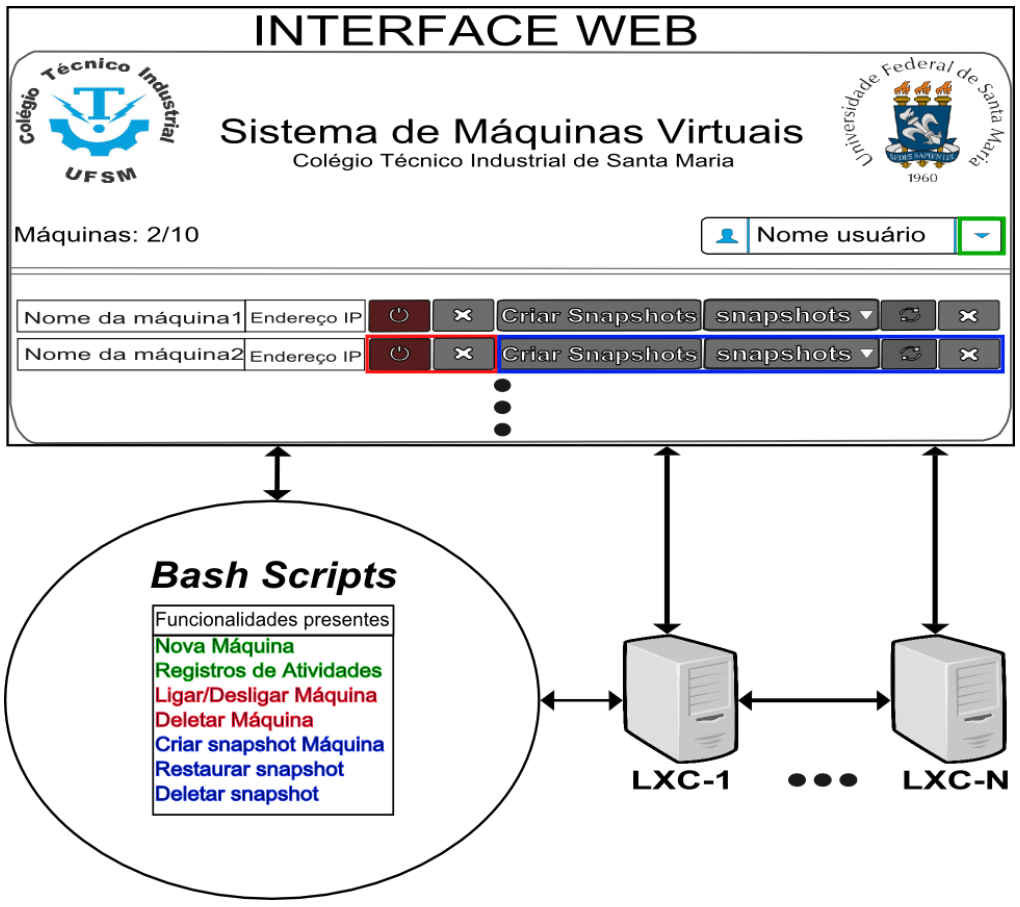

Figura 11: Estrutura de interação da interface WEB com servidor LXC no VMS v2.
Fonte: Acervo Pessoal.

No VMS v2 em relação ao VMS v1, não se faz mais necessário a todo *script* executado no servidor, esperar um retorno da ação para enviar este à interface WEB. Isso se torna possível em função da API-REST ter capacidade de realizar buscas de informações relativas a *containers* diretamente no servidor LXC.

Neste também se tem a possibilidade de adição de mais servidores LXC, onde através da migração de *containers*, com alguma ferramenta que forneça um serviço de balanceamento de carga, permite-se não sobrecarregar servidores e possuir um *hardware* adicional para alocar um maior número de *containers*.

Na Figura 12, pode ser analisado troca de mensagens que ocorre entre a interface WEB e o servidor LXC, no qual consiste basicamente de um único envio de *script* e fecha a conexão. Dessa forma, o retorno dessa operação é buscado através da interface WEB utilizando-se da API-REST.

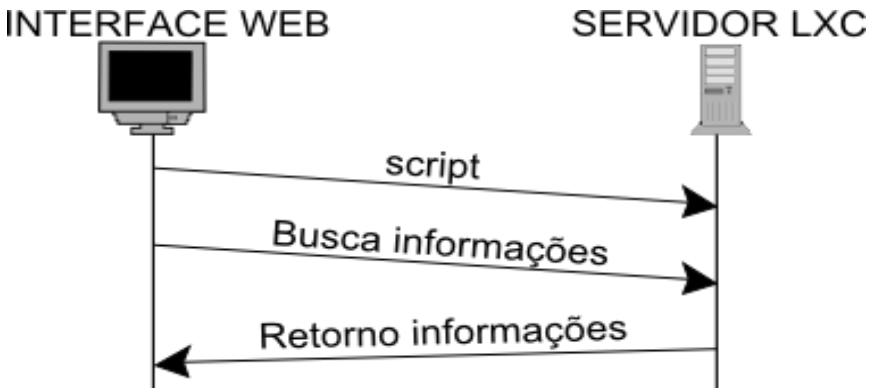

Figura 12: Diagrama de comunicação entre interface WEB e servidor LXC no VMS v2.
Fonte: Acervo Pessoal.

Seguindo o mesmo exemplo, supondo que é clicado na interface no botão de ligar a máquina virtual, o *script* para ligar a máquina é enviado para o servidor e executado, assim ligando o *container*. Contudo diferentemente do VMS v1, esse *script* não espera nenhum retorno do servidor, deixando a busca de informações dos *containers* a cargo da interface WEB. Então, após a execução do *script* e do *container* rodando, a interface WEB envia um comando via API e assim recebe um retorno com informações sobre o *status* do *container*. Com base neste retorno modifica ou não a cor do botão que indica seu *status* como no exemplo citado.

6 IMPLANTAÇÃO DO MODELO PROPOSTO

Para elaboração deste projeto, foi utilizada uma máquina virtual fazendo uso do modelo de virtualização total, pois assim a simulação fica o mais próximo possível de um servidor físico dedicado para esse fim. O *software* utilizado para criação desta máquina foi o VirtualBox, onde neste foram utilizadas as seguintes configurações de *hardware*:

- Memória RAM: 32GB;
- Número de CPUs (virtuais): 10;
- Tamanho de Disco: 50GB;

O sistema operacional instalado nesta máquina foi Ubuntu Server 14.04, nesta mesma máquina foi implementada toda interface gráfica e o servidor LXC/LXD. Nesse contexto, essa máquina foi denominada “server-lxc”.

Uma outra máquina virtual, também utilizando o VirtualBox, foi criada para desempenhar o papel de servidor de banco de dados, onde esta armazena as informações referentes aos *containers* como estabelecido na interface gráfica. As configurações de *hardware* utilizado para essa máquina foram:

- Memória RAM: 800M;
- Número de CPUs (virtuais): 1;
- Tamanho de Disco: 8GB;

O sistema operacional utilizado nesta máquina também foi o Ubuntu Server 14.04. Esta máquina foi utilizada para atuar exclusivamente como um servidor de banco de dados, para simular um cenário de servidores o mais próximo possível do sistema que se encontra implementado na atualidade.

As máquinas virtuais servidor de banco de dados e servidor LXC/LXD foram criadas dentro de um servidor físico, com sistema operacional Ubuntu Server 14.04 onde este possui as seguintes configurações de *hardware*:

- Memória RAM: 48GB;
- Número de CPUs: 4;
- Número de Cores por CPU: 8;
- Tamanho de Disco: 4TB;

6.1 **Servidor LXC/LXD**

Com as ferramentas LXC/LXD devidamente instaladas no servidor, sua configuração é relativamente fácil, pois possui um conjunto reduzido de arquivos de configurações, onde estes são bem documentados e de fácil compreensão. Após realizar todas configurações desejadas como por exemplo: definir a range de DHCP, máscara de rede utilizada, IP da interface que fará a *bridge* etc. Antes do servidor estar pronto para criação de *containers* é necessário baixar uma imagem de sistema operacional para ser espelhado e distribuído para todos *containers* que serão criados. A imagem escolhida foi Ubuntu 14.04, após a conclusão de importação desta imagem o servidor está pronto para criar *containers*.

Para utilização da API-REST, qualquer comando remoto ou migração de *containers*, que possa ser utilizada mais tarde, é necessário configurar uma senha que fará autenticação com o *socket* do LXC. Além da configuração da senha, no primeiro acesso remoto ao servidor é necessário o servidor aceitar o certificado para validar a conexão.

Na Figura 13, pode ser observado como o comando *list* exibe todos containers criados, e além disso informações de nome do *container*, *status*, rede e número de *snapshots* criados referentes a qual *container*.

```
root@server-lxc:~# lxc list
+-----------------+---------+--------------+------+-----------+-----------+
|      NAME       |  STATE  |     IPV4     | IPV6 | EPHEMERAL | SNAPSHOTS |
+-----------------+---------+--------------+------+-----------+-----------+
| vmbrunomaquinaa | RUNNING | 10.99.13.228 |      | NO        | 2         |
| vmbrunomaquinab | RUNNING | 10.99.9.113  |      | NO        | 0         |
| vmbrunomaquinac | STOPPED |              |      | NO        | 0         |
| vmteste1        | RUNNING | 10.99.39.168 |      | NO        | 0         |
| vmteste2        | RUNNING | 10.99.7.207  |      | NO        | 0         |
| vmteste3        | RUNNING | 10.99.46.181 |      | NO        | 0         |
| vmteste4        | RUNNING | 10.99.5.199  |      | NO        | 0         |
| vmteste5        | RUNNING | 10.99.50.232 |      | NO        | 0         |
+-----------------+---------+--------------+------+-----------+-----------+
```

Figura 13: Lista de containers criados sendo exibidos através do script de listagem do LXD.
Fonte: Acervo Pessoal.

6.2 **Interface WEB modificada**

A interface WEB se encontra hospedada na própria máquina servidor LXC, assim os comandos enviados através da API são executados diretamente no servidor. No desenvolvimento da interface foi necessário a instalação de mais *softwares* nesta máquina para esta cumprir a função também de servidor WEB, os *softwares* adicionais que foram instados para atender a este propósito foram basicamente, php5, apache2 e php5-curl.

Para desenvolvimento da interface WEB, foram utilizados os códigos do VMS v1 como base e reajustado na sua maior parte para assim poder atender a este VMS v2. Para o usuário final o que pode ser percebido é uma mudança na interface, com funcionalidade adicionais, porém o mais próximo possível do *layout* com que estes já estavam familiarizados.

Na Figura 14, é exposto com a primeira tela da interface é exibida para o usuário, onde nesta tela de *login* o usuário informa suas credenciais de acesso conforme o cadastro no LDAP da instituição.

Figura 14: Tela de login da nova interface gráfica.
Fonte: Acervo Pessoal.

Até então tudo exatamente igual ao VMS v1 para o usuário, após efetuar o *login* surge as únicas modificações que podem ser visíveis aos usuários finais. Devido a implantação de uma VPN (*Virtual Private Network*) para acesso externo às máquinas virtuais é depreciado o SSH via *gateone* que até então está presente no VMS v1, por este motivo foi removido desta nova versão. Foi removido também, por falta de uso dos usuários e por ser acessível apenas de dentro da rede interna do CTISM, o FreeNX que era utilizado para acesso à interface gráfica das máquinas virtuais. Além destes, foi removida também a opção de restaurar a máquina virtual para o estado *default*, ou seja, o estado inicial da máquina, esta foi considerada depreciada pois, a possibilidade de criar *snapshots* e restaurá-los já engloba essa funcionalidade se o usuário assim desejar.

Sobre a parte lógica, diferente do VMS v1 ocorreu mudanças no que tange as chamadas de sistema e informações enviadas para interface WEB, onde no VMS v1 os scripts depois de executados no servidor aguardavam o retorno de sua execução para assim a interface WEB receber as informações. Já VMS v2, os scripts são executados no servidor LXC e a conexão entre interface WEB e servidor LXC é encerrada. Para busca de informações referentes aos *containers* ou relativas ao servidor LXC, é utilizado a API fornecida através do LXD e por meio desta, é aberta uma comunicação com o servidor LXC passando como parâmetro via URL a consulta que deseja receber informações. Após o recebimento do retorno da consulta a conexão é encerrada, até a próxima consulta.

Na Figura 15, é apresentado a primeira tela após o usuário efetuar o *login* com sucesso. Nesta página são exibidas todas as máquinas virtuais criadas relativas a aquele usuário em específico, tal como nome das respectivas máquinas, seus endereços IP, e um painel de comandos no qual este possui modificações ao modelo atual. Neste pode ser percebido a inclusão da funcionalidade *snapshots*, onde através do botão “criar snapshot” uma cópia do estado atual da máquina é gerada e o usuário nomeia este *snapshot*. Ao lado deste possui um menu *dropdown*, onde neste é exibido todos os *snapshots* criados referentes a aquele *container* em específico. Ao lado existem dois botões, onde estes são ativados, ou seja, clicáveis apenas quando um *snapshot* do menu estiver selecionado, logo, após selecionar um *snapshot* é possível com um simples clique restaurar ou deletar aquele *snapshot* selecionado. Agora também uma máquina no estado de desligada não fica ocupando um endereço IP, minimizando as chances de esgotamento de endereços a longo prazo, assim que a máquina é ligada novamente o DHCP configurado no servidor LXC envia um endereço IP para a máquina.

Figura 15: Página pessoal do usuário na nova interface gráfica.
Fonte: Acervo Pessoal.

7 RESULTADOS

Para análise de desempenho do novo sistema, foi criado um conjunto de *scripts* que cronometraram o tempo para cada operação desde seu início ao seu final. As coletas destes dados foram relativas a criação de *containers*, criação de *snapshots*, restauração de *snapshots* e remoção de *containers*, respectivamente. Estas operações realizadas por estes *scritps* foram realizadas de forma sequencial e estes foram executados diretamente no servidor LXC.

A Figura 16, é um gráfico representado através da relação tempo x quantidade de máquinas virtuais, que demonstra o tempo em segundos para criação de determinados números de *containers*. Dessa forma, pode ser observado através deste que o tempo para criação de novas máquinas mostrou-se linear com o incremento do número de máquinas a serem criadas.

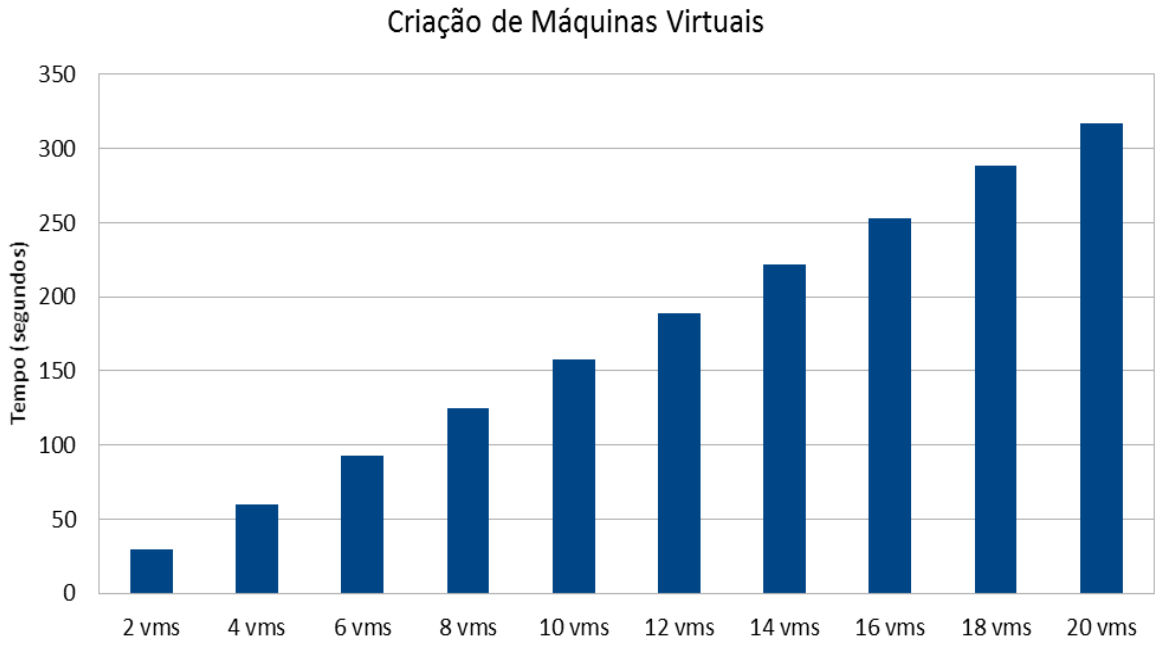

Figura 16: Gráfico da relação entre tempo x número de máquinas, para criação de máquinas.
Fonte: Acervo Pessoal.

Na Figura 17, é possível verificar o tempo de criação de *snapshots* em função do número de *snapshots* criados. Deve ser salientado também que, para esta amostragem de *snapshots* as máquinas da Figura 16 já se encontravam criadas, visto que, para criação de um *snapshot* é preciso de um *container* de referência. Através deste, pode ser observado que as

operações não se mantiveram lineares como o gráfico de criação. Isso se deve ao fato de que as operações de criação de *snapshots* dependem totalmente da velocidade de leitura e gravação do disco da máquina servidor. Neste contexto, utilizando discos rígidos de maior rotação por minutos ou discos de estado sólido (SSD), o tempo de criação de *snapshots* seria diminuído.

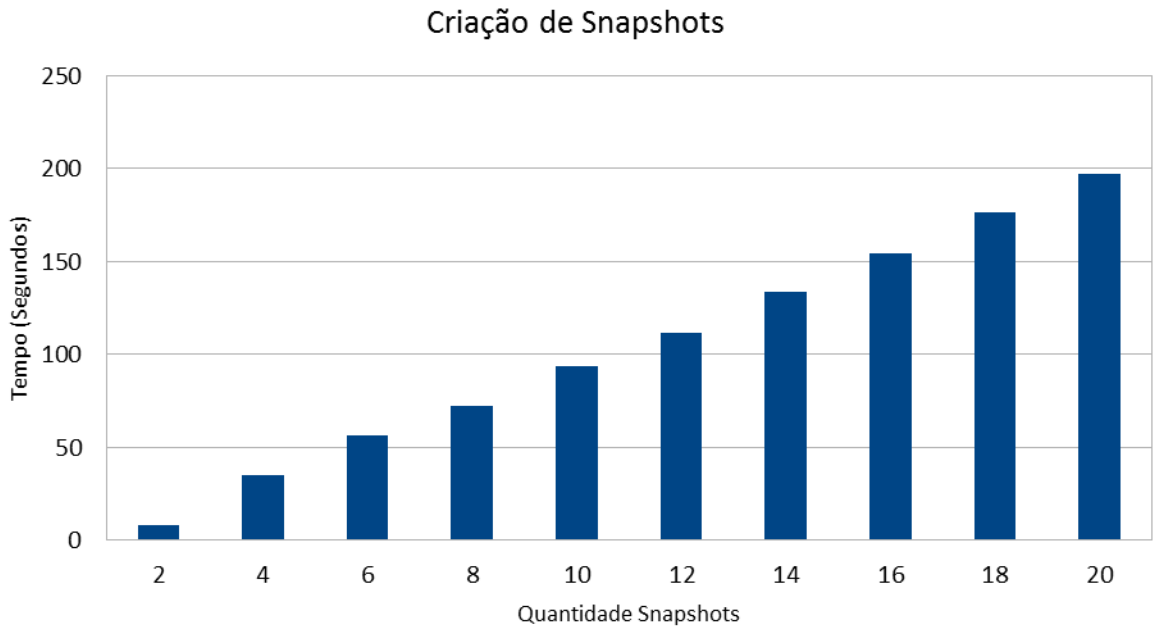

Figura 17: Gráfico da relação entre tempo x número de snapshots, para criação de snapshots.
Fonte: Acervo Pessoal.

No gráfico representado pela Figura 18, é possível analisar o tempo de restauração de determinados números de *snapshots*. Vale ressaltar que para geração deste gráfico, tanto os *containers* quanto os *snapshots* relativos a estes já haviam sido criados. A operação de restauração de *snapshots* segue o mesmo princípio da criação de *containers*, onde este também faz uso de leitura e gravação no disco. Primeiramente, é removido o *container* desejado e movido da pasta *containers*, todo diretório que contém o *rootfs* daquele determinado *container*.

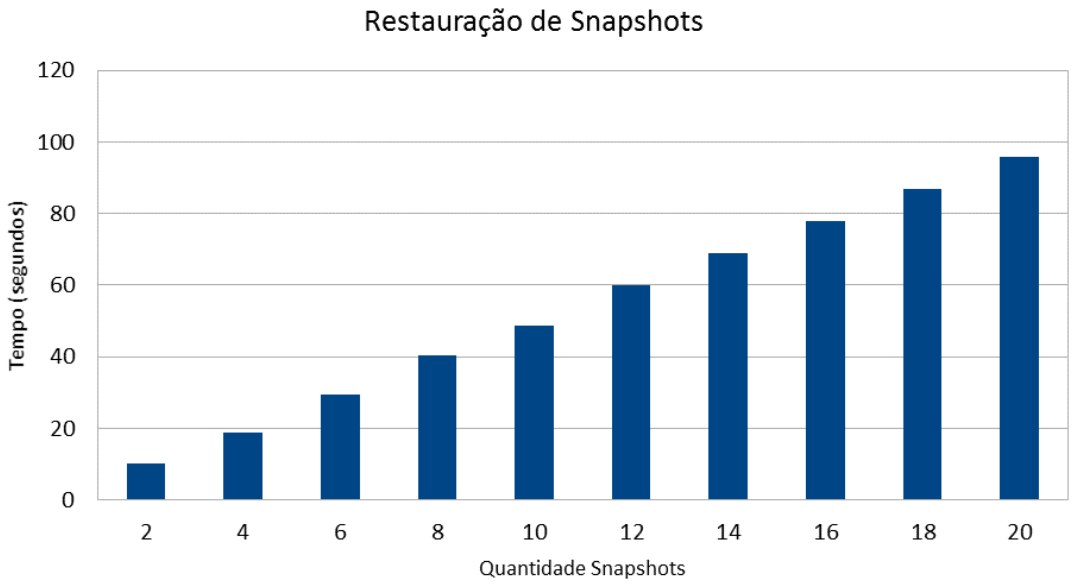

Figura 18: Gráfico da relação entre tempo x número de snapshots, para restauração de snapshots.
Fonte: Acervo Pessoal.

Na Figura 19, é expresso a relação tempo em função do número de máquinas a ser removido. Ressalta-se que as operações de exclusão de máquinas, não removem apenas o número de *containers* criados estabelecido. Esta operação além da remoção dos *containers* criados, removem também todos *snapshots* vinculados a estes.

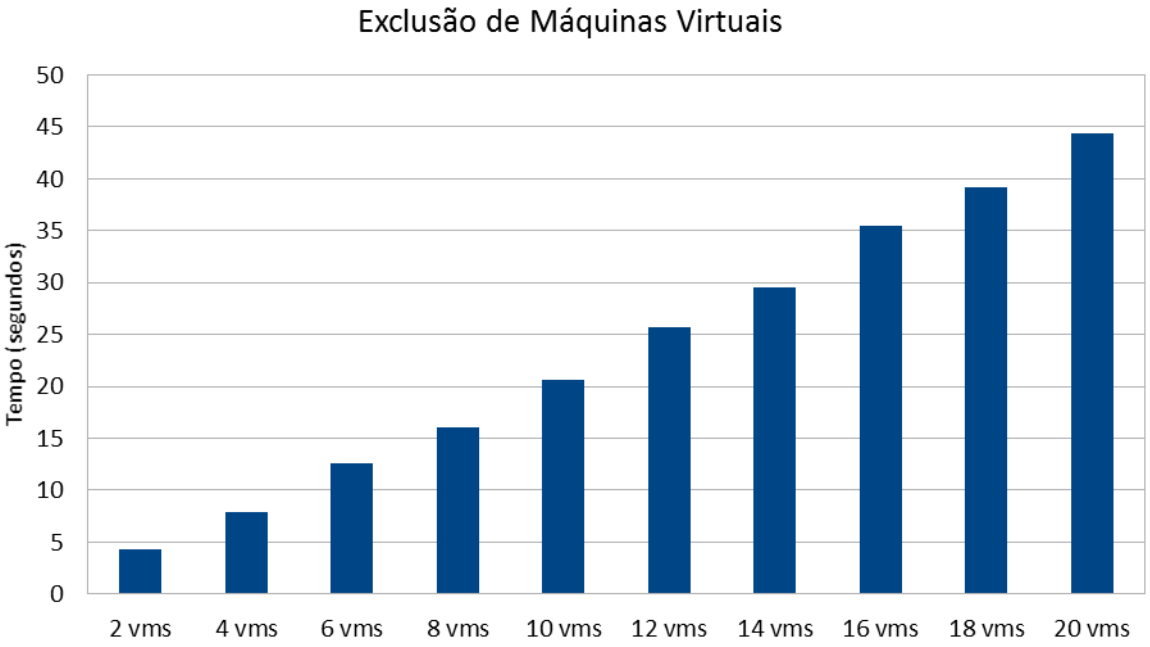

Figura 19: Gráfico da relação entre tempo x número de máquinas, para exclusão de máquinas.
Fonte: Acervo Pessoal.

Deve ser enfatizado que, todas operações citadas para elaboração dos gráficos aqui mostrados, por se tratar de operações que englobam leitura e gravação em disco, não dependem somente do poder de processamento e/ou memória RAM do servidor. Estas dependem igualmente da velocidade de leitura e gravação de arquivos no disco rígido do servidor LXC.

8 CONSIDERAÇÕES FINAIS

A proposta apresentada neste trabalho evidenciou resultados satisfatórios, considerando que atendeu ao objetivo proposto. Ficou comprovado a adição de funcionalidades que até então não estavam presentes ao VMS v1. Uma versão beta deste sistema, foi divulgado a alguns alunos dos cursos de Redes de Computadores e Técnico em Informática integrado ao Ensino Médio, ambos do CTISM, para testes de aceitação e/ou críticas. Dessa forma, o novo sistema obteve sucesso na aceitação e melhoria significativa em relação ao VMS v1.

Portanto, a contribuição deste trabalho é trazer um novo sistema de máquinas virtuais, que não se ausente da essência com o qual foi desenvolvido o sistema antigo. Com a vantagem da resolução de problemas presentes até hoje no VMS v1, e com adição de dois pontos cruciais para um sistema de virtualização que atende um público crescente, uma maior flexibilidade e escalabilidade do sistema como um todo.

A comunidade de desenvolvedores da ferramenta LXD mostrou-se participativa, tanto na correção de problemas que eventualmente são descobertos em suas versões, quanto em resolução de problemas dos usuários que fazem uso dessa ferramenta, com ajuda em fóruns ou diretamente no canal IRC da ferramenta. Onde através deste canal as eventuais dúvidas técnicas sobre a ferramenta podem ser sanadas pelos próprios desenvolvedores. Outro ponto positivo da utilização do LXD é a frequência de atualização desta, com inclusão de novas funcionalidades e correção de *bugs*, a versão utilizada para elaboração deste trabalho foi a versão 0.17 em setembro, em novembro já havia disponível a versão 0.23 para download. Dessa forma, exemplificando o alto índice de participação dos desenvolvedores para melhoria desta ferramenta.

PROJETOS FUTUROS

A Como proposta futura, para um melhor desempenho do novo sistema aqui proposto, faz-se de grande importância este sistema ser implementado diretamente em um servidor físico, que por sua vez proverá um maior recuso de *hardware* disponível para o sistema. Nesse contexto, será necessário a implementação de *quotas* de disco para cada *container* de cada usuário, assim impedindo a possibilidade de sobrecarga dos discos rígidos dos servidores.

Seria interessante também, complementar ainda mais a ideia aqui exposta com uso de outras ferramentas que agregariam maior escalabilidade e segurança ao sistema. A ideia base para complementar todo trabalho aqui proposto, consiste na interação de três novas ferramentas, sendo elas Openstack, MAAS e por último uma ferramenta de monitoramento de carga para os servidores.

Com o passar dos anos, a computação na nuvem (*cloud computing*) vem crescendo gradativamente, então parte de um projeto futuro é inserir todo sistema na nuvem. Nesse contexto, a ferramenta Openstack nos permite criar uma nuvem pública ou privada. Nos oferecendo a liberdade de inserção deste sistema em nuvem, devido a sua compatibilidade com todas ferramentas utilizadas até o momento e com as demais aqui propostas.

Diferente do VMS v1, o modelo proposto permite migração de containers entre diversos servidores LXC, então surge a ideia de utilização de um cluster LXC. Onde para criação do *cluster*, será utilizado a ferramenta MAAS, que possui uma interface amigável e principalmente por permitir ao administrador inserir e remover *nodes* do *cluster* com um simples clique através da interface web por esta fornecida. Com a utilização do MAAS, um servidor é utilizado como *node* ou *cluster master*, onde este é o responsável por todo gerenciamento dos *nodes slaves* a este vinculados.

Uma ferramenta de monitoramento dos *nodes* pertencentes ao servidor MAAS faz-se necessária, pois, deve haver um balanceamento de carga entre os *nodes*. Nesse contexto, com a utilização de uma ferramenta que realize o balanceamento de carga, é eliminado as possíveis falhas como travamento de algum dos *nodes* por sobrecarga de utilização. Assim esta ferramenta deverá detectar qual dos *nodes* pertencentes ao MAAS está mais ocioso e informa ao *cluster master*. Para essa ferramenta de monitoramento é proposto ser desenvolvida em

linguagem PHP e que esta faça buscas de informações na MIB de cada *node* pertencente ao *cluster* via protocolo SNMP. Onde este já estaria previamente instalado em cada servidor.

Na Figura 20, pode ser observado uma estrutura de forma abstrata de como seria o funcionamento básico do sistema que aqui é proposto. Pode-se perceber que neste projeto futuro do sistema a interface WEB não possui alteração em relação as demais, pois todo conceito de *cloud* será aplicado somente nos servidores LXC.

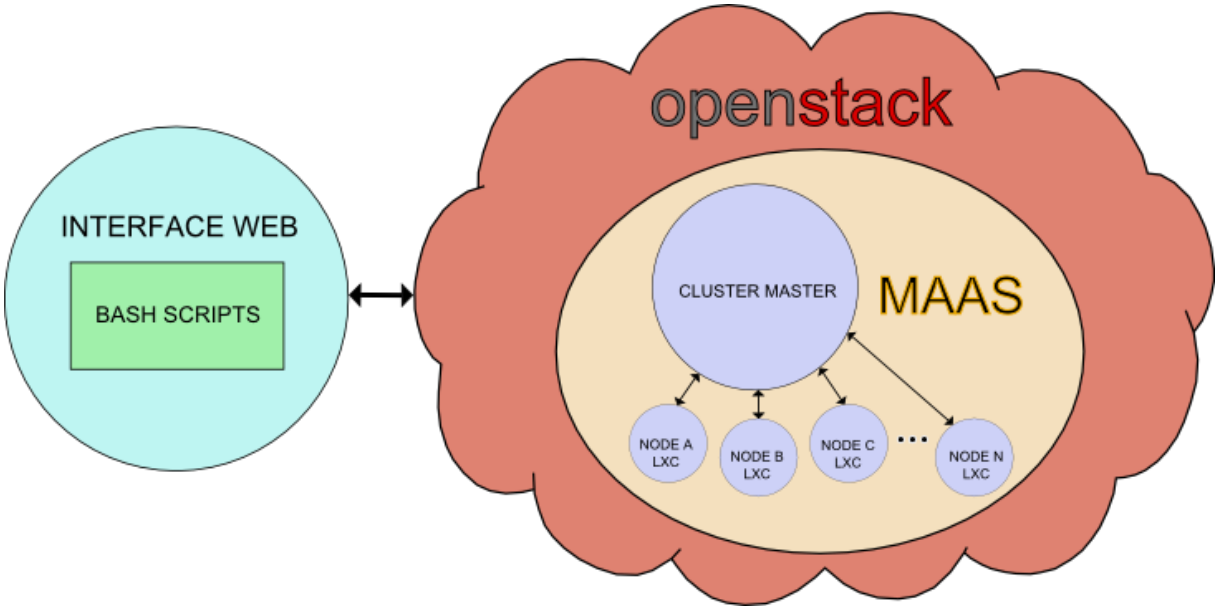

Figura 20: Esquema futuro de integração de serviços e servidores com a interface WEB.
Fonte: Acervo Pessoal.

REFERÊNCIAS

CARISSIMI, Alexandre. **Virtualização: da teoria a soluções**. Minicursos do Simpósio Brasileiro de Redes de Computadores–SBRC, v. 2008, p. 173-207, 2008.

CRIU.ORG. **Welcome to CRIU, a project to implement checkpoint/restore functionality for Linux in userspace**, 2015. Disponível em: < https://criu.org/Main_Page>. Acesso em: 30 de outubro. 2015.

DA SILVA, Rodrigo Ferreira; DE OLIVEIRA, Fábio Borges. **Virtualização de Sistemas Operacionais**. 2007.

FERNANDES, Almir Dominicini. **Avaliação experimental de técnicas de virtualização através de balanceamento de carga em clusters de computadores**. 2010. Tese de Doutorado. Universidade Federal do Rio de Janeiro.

IBM.COM. **Container, o novo passo para a virtualização**, 2014. Disponível em: <https://www.ibm.com/developerworks/community/blogs/tlcbr/entry/mp234?lang=en>. Acesso em: 19 de setembro. 2015.

INSIGHTS.UBUNTU.COM. **Introduction to nova-compute-lxd**, 2015. Disponível em: <https://insights.ubuntu.com/2015/05/06/introduction-to-nova-compute-lxd>. Acesso em: 19 de setembro. 2015.

INSIGHTS.UBUNTU.COM. **Live Migration in LXD**, 2015. Disponível em: <https://insights.ubuntu.com/2015/05/06/live-migration-in-lxd/>. Acesso em: 17 setembro. 2015.

LINUXCONTAINERS.ORG. **What's LXC?**. Disponível em: <https://linuxcontainers.org/lxc/introduction/>. Acesso em: 17 de setembro. 2015.

MATTOS, Diogo Menezes Ferrazani. **Virtualização: VMWare e Xen**. Grupo de Teleinformática e Automação da UFRJ, p. 13, 2008.

OPENSTACK.ORG. **OpenStack compute**. Disponível em: <http://os-new-software.getforge.io/software/openstack-compute/>. Acesso em: 26 de outubro. 2015.

REIS, Wanderson Santiago dos. **Virtualização de serviços baseado em contêineres: uma proposta para alta disponibilidade de serviços em redes Linux de pequeno porte**. 2015.

ROCHA, Lucio Agostinho**. Introdução à Computação em Nuvem**. 2013.

SILVA, Luis Paulo da. **Uso da tecnologia de virtualização para melhor aproveitamento de recursos de hardware**. FaSCi-Tech, v. 1, n. 2, 2012.

SAGIROGLU, Seref; CANBEK, Gurol. **Keyloggers**. Technology and Society Magazine, IEEE, v. 28, n. 3, p. 10-17, 2009.