O modelo de computação em nuvem possibilitou o provisionamento dinâmico de Infraestruturas Virtuais (IVs), compostas por Máquinas Virtuais (MVs), contêineres e recursos de rede virtualizados. Diante da maleabilidade no provisionamento e configuração dos recursos, é recorrente a adoção de IVs para hospedar serviços como armazenamento de dados, plataformas de desenvolvimento e aplicações distribuídas. Entretanto, as tecnologias de computação em nuvem, apesar de inovadoras, não possibilitam um modo facilitado para migração dos recursos virtualizados entre provedores distintos, induzindo a ocorrência de vendor lock-in. Dessa forma este trabalho propõe um mecanismo para realizar a migração de IV entre provedores.

Orientador: Guilherme Piêgas Koslovski

Coorientador: Charles Christian Miers

Joinville, 2018

ANO
2018

EUCLIDES CARDOSO JÚNIOR | MIGRAVI: UMA PROPOSTA PARA MIGRAÇÃO DE INFRAESTRUTURAS VIRTUAIS ENTRE PROVEDORES DE NUVES IAAS

UDESC

UNIVERSIDADE DO ESTADO DE SANTA CATARINA – UDESC
CENTRO DE CIÊNCIAS TECNOLÓGICAS – CCT
PROGRAMA DE PÓS GRADUAÇÃO EM COMPUTAÇÃO APLICADA

DISSERTAÇÃO DE MESTRADO

MIGRAVI: UMA PROPOSTA PARA MIGRAÇÃO DE INFRAESTRUTURAS VIRTUAIS ENTRE PROVEDORES DE NUVENS IAAS

EUCLIDES CARDOSO JÚNIOR

JOINVILLE, 2018

EUCLIDES CARDOSO JÚNIOR

MIGRAVI: UMA PROPOSTA PARA MIGRAÇÃO DE INFRAESTRUTURAS VIRTUAIS ENTRE PROVEDORES DE NUVENS IAAS

Dissertação submetida ao Programa de Pós-Graduação em Computação Aplicada do Centro de Ciências Tecnológicas da Universidade do Estado de Santa Catarina, para a obtenção do grau de Mestre em Computação Aplicada.

Orientador: Dr. Guilherme Piêgas Koslovski
Coorientador: Dr. Charles Christian Miers

JOINVILLE

2018

Cardoso Júnior, Euclides
MIGRAVI: UMA PROPOSTA PARA MIGRAÇÃO DE INFRAESTRUTURAS VIRTUAIS ENTRE PROVEDORES DE NUVENS IAAS / Euclides Cardoso Júnior. - Joinville , 2018.
80 p.

Orientador: Guilherme Piêgas Koslovski
Co-orientador: Charles Christian Miers
Dissertação (Mestrado) - Universidade do Estado de Santa Catarina, Centro de Ciências Tecnológicas, Programa de Pós-Graduação em Computação Aplicada, Joinville, 2018.

1. Computação em Nuvem. 2. IV. 3. IaaS. 4. Contêineres. 5. Máquina Virtual. I. Piêgas Koslovski, Guilherme. II. Christian Miers, Charles. , .III. Universidade do Estado de Santa Catarina, Centro de Ciências Tecnológicas, Programa de Pós-Graduação em Computação Aplicada. IV. Título.

MigraVI: Uma Proposta para Migração de Infraestruturas Virtuais entre Provedores de Nuvens IaaS

por

Euclides Cardoso Júnior

Esta dissertação foi julgada adequada para obtenção do título de

Mestre em Computação Aplicada

Área de concentração em "Ciência da Computação",
e aprovada em sua forma final pelo

CURSO DE MESTRADO ACADÊMICO EM COMPUTAÇÃO APLICADA
DO CENTRO DE CIÊNCIAS TECNOLÓGICAS DA
UNIVERSIDADE DO ESTADO DE SANTA CATARINA.

Prof. Dr. Guilherme Piêgas Koslovski
CCT/UDESC (Orientador/Presidente)
Membro da Banca Examinadora

Prof. Dr. Maurício Aronne Pillon
CCT/UDESC
Membro da Banca Examinadora

Prof. Dr. Fernando Frota Redigolo
USP
Membro da Banca Examinadora

Prof. Dr. Charles Christian Miers
CCT/UDESC (Coorientador)

Joinville,SC, 30 de julho de 2018.

Dedico este trabalho aos meus familiares, amigos, colegas e professores que me acompanharam e me deram forças nessa magnífica trajetória.

AGRADECIMENTOS

Desejo agradecer primeiramente a Deus, por ter me dado força e saúde nos momentos mais difíceis e ter me ajudado a superá-los. À minha esposa, por sempre estar ao meu lado e pela paciência e amor a mim oferecidos em todos os momentos. Aos meus orientadores professor Dr. Guilherme P. Koslovski e professor Dr. Charles C. Miers que foram tão importantes na minha vida acadêmica e no desenvolvimento desta dissertação. À todos que direta ou indiretamente fizeram parte da minha formação: o meu muito obrigado.

“A melhor maneira encontrada pelo homem para se aperfeiçoar é aproximando-se de Deus.”

Pitágoras

RESUMO

O modelo de computação em nuvem possibilita aos usuários a visão de que os recursos computacionais são ilimitados, podendo estes serem reservados dinamicamente. Ademais, os usuários são tarifados de acordo com a configuração contratada e tempo efetivo de reserva. Desse modo, tornou-se possível o provisionamento dinâmico de Infraestruturas Virtuais (IVs), compostas por Máquinas Virtuais (MVs), contêineres e recursos de rede virtualizados. Diante da maleabilidade no provisionamento e configuração dos recursos, é recorrente a adoção de IVs para hospedar serviços como armazenamento de dados, plataformas de desenvolvimento e aplicações distribuídas. Porém, ao migrarem aplicações para a nuvem, os usuários se deparam com uma quantidade expressiva de provedores, corretores e serviços oferecidos. Sobretudo, as tecnologias de computação em nuvem, apesar de inovadoras, não possibilitam um modo facilitado para migração dos recursos virtualizados entre provedores distintos, induzindo a ocorrência de *vendor lock-in*. A grande diversidade de provedores de nuvem, *Application Programming Interfaces* (APIs), modelos e ferramentas de gerenciamento limitam o desenvolvimento das soluções para migração de IV entre provedores, caracterizando um desafio em aberto. Este trabalho propõe um mecanismo para realizar a migração de IV entre provedores. A arquitetura resultante é agnóstica aos provedores de origem e destino, bem como às aplicações hospedadas na IV. Adicionalmente, um protótipo baseado em contêineres Docker e nuvens OpenStack é apresentado indicando como uma implementação pode ser realizada. Testes realizados com o protótipo indicam que a rede pode impactar no tempo de migração. Além disso, o tamanho dos arquivos que precisam ser sincronizados entre as nuvens de origem e de destino também degradam o desempenho da migração.

Palavras-chaves: Computação em Nuvem. IV. IaaS. Contêineres. Máquina Virtual e Migração.

ABSTRACT

Cloud computing model enables users to see computing resources as unlimited and possible of being dynamically reserved. In addition, users are charged according to a contracted configuration, and effective reservation time. Thus, it became possible to provision VI dynamically, composed of VMs, containers, and virtualized network resources. Faced with malleability in provision and configuration of resources, it is recurent the adoption of VI to host data storage services, development platforms, and distributed applications. However, when migrating applications to a cloud, users are faced with a significant amount of providers, brokers, and services offered. Above all, cloud computing technologies, although innovative do not facilitate the migration of virtualized resources between distinct providers, inducing on vendor lock-in. A wide range of cloud providers, APIs, models, and management tools limit the development of VI migration solutions between providers, characterizing an open challenge. This work proposes a mechanism to perform VI migration among providers. The resulting architecture is agnostic to source and destination providers, as well as VI-hosted applications. In addition, the prototype based on Docker containers and OpenStack clouds is an indication how an implementation can be performed. Tests performed with the prototype shows the network can impact migration time. In addition, the size of the files needed to be synchronized between source and destination clouds also impacts on migration performance.

Key-words: Cloud Computing. Virtual Infrastructure. IaaS. Containers. Virtual Machine. Migration.

LISTA DE ILUSTRAÇÕES

LISTA DE TABELAS

LISTA DE SIGLAS E ABREVIATURAS

API *Application Programming Interface*

BGP *Border Gateway Protocol*

CIB *Cloud Interoperability Broker*

CLEI Conferência Latino-americana de Informática

COS *Cloud Operating System*

CPU *Central Processing Unit*

CRIU *Checkpoint/Restore In Userspace*

CSAR *Cloud Service Archive*

DHCP *Dynamic Host Configuration Protocol*

ERAD Escola Regional de Alto Desempenho

GENI *Global Environment for Network Innovations*

HOT *Heat Orchestration Template*

HTTP *Hypertext Transfer Protocol*

IaaS *Infrastructure as a Service*

InPs *Infrastructure Providers*

ISP *Internet Service Providers*

IP *Internet Protocol*

IV Infraestrutura Virtual

LabP2D Laboratório de Processamento Paralelo e Distribuído

LAN *Local Area Network*

MMV Monitor de Máquina Virtual

MV Máquina Virtual

NFS *Network File System*

NIST National Instutite of Standards and Technology

PaaS *Platform as a Service*

QoS *Quality of Service*

RBCA Revista Brasileira de Computação Aplicada

RV Rede Virtual

SaaS *Software as a Service*

SDN *Software Defined Networks*

SLA *Service Level Agreement*

SO Sistema Operacional

SP *Service Providers*

SSH *Secure Shell*

TOSCA *Topology and Orchestration Specification for Cloud Applications*

UDESC Universidade do Estado de Santa Catarina

URL *Uniform Resource Locator*

VLAN *Virtual Local Area Network*

VPC *Virtual Private Cloud*

VPN *Virtual Private Network*

WAN *Wide Area Network*

SUMÁRIO

1 INTRODUÇÃO

O paradigma de computação em nuvem provocou diversas mudanças na maneira como os recursos computacionais são provisionados e gerenciados. Esse paradigma possibilita que recursos, *e.g.,* armazenamento, processamento e comunicação, sejam alocados de maneira dinâmica pelo usuário da nuvem. Neste sentido, provedores de nuvens *Infrastructure as a Service* (IaaS) podem fornecer serviços elásticos de Infraestrutura Virtual (IV). Uma IV é composta por máquinas, contêineres e *switches* virtualizados, interconectados por uma rede virtual privada (LIU et al., 2012). Cada recurso pode ser individualmente redimensionado para atender a carga de trabalho requerida pelas aplicações dos usuários. Exemplificando, provedores públicos renomados (*e.g.,* Amazon EC2 e Google) oferecem IVs com a terminologia de Nuvens Virtuais Privadas ou *Virtual Private Cloud* (VPC).

Com a popularização da computação em nuvem ocorreu uma proliferação de provedores de nuvens IaaS (ZHANG; WU; CHEUNG, 2013; RAUGUST et al., 2018). Internamente, cada provedor realiza a orquestração de sua infraestrutura com ferramentas específicas, geralmente otimizadas para gerenciar os recursos de acordo com os serviços oferecidos pelo provedor. Consequentemente, o ambiente de computação em nuvem tornou-se heterogêneo e complexo. Os usuários acreditam que criam IVs de maneira independente dos provedores, entretanto as otimizações realizadas pelos provedores podem criar âncoras nos recursos virtualizados, fixando as IVs aos provedores (KAUR; SHARMA; KAHLON, 2017).

Eventualmente, clientes de provedores IaaS podem desejar a migração e realocação de IVs em outros provedores ou regiões geográficas. De acordo com Clark et al. (2005), o processo de migração consiste em transferir de maneira consistente uma aplicação que está em hospedeiro para um outro. Independentemente das questões burocráticas dos provedores envolvidos, uma migração possui diversos aspectos técnicos desafiadores que devem ser tratados, cada um de modo diferenciado. Fatores desafiadores para realizar a migração de IVs são observados quanto as políticas de cobrança, que podem variar entre provedores e entre regiões, e quanto a qualidade do serviço recebida, essencial para o desempenho das aplicações hospedadas nas IVs. Entretanto, diversos provedores de nuvem não conseguem atender os requisitos de aplicações sensíveis à latência (CHOY et al., 2012; LIN; SHEN, 2017).

Conforme apresentado em (ZHANG; WU; CHEUNG, 2013), os usuários enfrentam diversos problemas para realizar a migração de IVs entre provedores distintos. As dificuldades elencadas são em função da falta de padronização e até mesmo por

problemas legais. Geralmente, abordagens padronizadas são centradas nos provedores e mesmo que padrões e abordagens centradas nos usuários surjam, e tenham o objetivo de melhorar as questões e interoperabilidade e de portabilidade, existe a relutância dos grandes provedores em adotar estes padrões. Essa relutância deve-se ao fato de que essa incompatibilidade entre provedores de nuvem pode proteger seus interesses individuais (KAUR; SHARMA; KAHLON, 2017; HOFMANN; WOODS, 2010).

Não obstante, soluções e tecnologias que auxiliem na realização desta migração encontram-se em um estado inicial de desenvolvimento (GROZEV; BUYYA, 2014; TOOSI; CALHEIROS; BUYYA, 2014; GARCÍA; CASTILLO; FERNáNDEZ, 2016; TANG et al., 2018). Realizar a migração de IVs ainda é um assunto em aberto (ZHAO; ZHOU, 2014; JAMSHIDI; AHMAD; PAHL, 2013; MEDINA; GARCÍA, 2014) e esse tipo de migração requer uma atenção especial, pois além de migrar as aplicações dos usuários, toda a rede subjacente que faz parte da IV deverá ser migrada concomitantemente. Nesse cenário, dois aspectos importantes da rede devem ser considerados:

1. Manutenção das conexões na rede privada interna. A preocupação é referente a manutenção da comunicação entre as aplicações quando realizada sobre a rede *Internet Protocol* (IP) interna. Após a migração, as aplicações devem retomar a execução sem a necessidade de estabelecimento de novas conexões, ou seja, devem se reencontrar internamente na nuvem.

2. Manutenção das comunicações externas com as aplicações hospedadas na IV e outros serviços de nuvem. Nesse caso, deve ser levado em consideração que estas conexões são realizadas através de endereços IPs públicos e, consequentemente, ao migrar para um novo provedor de nuvem, não é possível manter os mesmos endereços IP, pois cada provedor tem a sua própria faixa de IPs. Neste cenário, soluções para o redirecionamento do tráfego da rede devem ser utilizadas, com o objetivo de manter as conexões abertas e funcionando.

Para a realização deste trabalho foi realizada uma revisão sistemática sobre migração de IVs em um ambiente de computação em nuvem. Em resumo, de acordo com Gelain et al. (2014), este trabalho pode ser classificado como o paradigma tecnocrático. Quanto ao problema, o cenário descrito apresenta uma inconformidade com os requisitos de portabilidade e interoperabilidade propostos pelo National Instutite of Standards and Technology (NIST) (LIU et al., 2011). De acordo com Liu et al. (2011), da perspectiva do usuário a portabilidade trata da capacidade em mover dados e aplicações através de múltiplos ambientes de nuvem, sempre com o menor custo e com o mínimo de interrupção dos seus serviços. Ainda de acordo com Liu et al. (2011) a interoperabilidade, refere-se a capacidade do usuário utilizar seus serviços e da-

dos entre os diversos ambientes computacionais e através de uma interface unificada. Além disso, o NIST define que os provedores de nuvem devem fornecer mecanismos para suportar a interoperabilidade de serviço.

Nesse cenário, esse trabalho apresenta uma proposta para a realização da migração de IV entre provedores de nuvens distintos, privados ou públicos. Sendo, que dessa forma, a proposta pode ser caracterizada como um corretor de nuvem, o qual disponibiliza para o usuário de nuvem uma interface na qual ele pode realizar a migração de seus dados e serviços através de diferentes provedores de nuvem.

1.1 OBJETIVOS

O objetivo deste trabalho é propor e desenvolver um corretor de nuvem o qual possibilita ao usuário realizar a migração de IV, e dessa forma utilizar seus dados e serviços através de provedores de nuvem distintos.

Os objetivos específicos são:

- Identificação das particularidades e desafios na migração de Máquinas Virtuais (MVs) e contêineres entre provedores de nuvem.
- Estudo sobre os desafios existentes para a migração de redes virtualizadas entre provedores de nuvem.
- Definição de uma arquitetura de um corretor para a migração de IVs.
- Implementação de um protótipo baseado em OpenStack, o qual atenda os requisitos levantados no objetivo geral.
- Análise experimental do protótipo em ambiente controlado interno e posteriormente externo (outro domínio administrativo).

1.2 ORGANIZAÇÃO DO TRABALHO

O restante do trabalho esta organizado da seguinte forma: O Capítulo 2 apresenta uma revisão dos temas relacionados ao trabalho. Neste capítulo são revisados os conceitos de nuvens IaaS, virtualização de recursos em ambientes de computação em nuvem, Infraestrutura Virtual (IV), assim como também é discutido o tema de migração de recursos entre provedores de nuvem e também são revisados os atores de nuvem. No Capítulo 3 são apresentados os trabalhos relacionados, assim como é discutido o problema de migração de IV entre provedores distintos. Já no Capítulo 4 é apresentada uma proposta para o problema de migração de IV. Na sequência, o

Capítulo 5 apresenta um protótipo implementado em OpenStack assim como as ferramentas utilizadas no desenvolvimento do protótipo. O Capítulo 6 apresenta uma análise experimental do protótipo, ambientes de testes e resultados obtidos. Por fim, estão as considerações finais do trabalho e são discutidos trabalhos futuros.

2 REVISÃO BIBLIOGRÁFICA

Neste capítulo são revisados alguns conceitos de computação em nuvem relacionados com a proposta e desenvolvimento deste trabalho. O capítulo é organizado da seguinte maneira: a Seção 2.1 introduz a definição de nuvens computacionais com foco em serviços de infraestrutura. Em seguida, na Seção 2.2, são apresentadas as técnicas de virtualização de recursos, são elas, Máquinas Virtuais (MVs) e contêineres. Na sequência, a Seção 2.3 apresenta o conceito de Infraestruturas Virtuais (IVs). Já na Seção 2.4 é abordado sobre a migração em nuvens IaaS. A Seção 2.5 revisa os atores envolvidos no gerenciamento e utilização de nuvens computacionais, posicionando o presente trabalho. Por fim, na Seção 2.6, as considerações parciais do capítulo são apresentadas.

2.1 NUVENS IAAS

Diversas definições para o termo IaaS foram propostas durante os últimos anos, entretanto a definição mais aceita e utilizada é a fornecida pelo National Instutite of Standards and Technology (NIST), o qual define *Infrastructure as a Service* (IaaS) como uma abstração, na qual o usuário possui a capacidade de alocar recursos computacionais como serviços, de forma que estes são fornecidos por algum provedor remoto (MELL; GRANCE, 2011). Dessa forma, nuvens IaaS são nuvens que oferecem recursos computacionais como serviços. Esses recursos computacionais podem ser Máquinas Virtuais (MVs) e serviços de armazenamento, todos agregados por uma rede (comutadores, roteadores e enlaces) com elementos reais e virtualizados.

Em nuvens IaaS, os serviços virtualizados podem ser alocados e escalados com rapidez e eficiência pelo próprio usuário (MELL; GRANCE, 2011; BHARDWAJ; JAIN; JAIN, 2010). Outra característica é que cada um destes recursos computacionais possuem suas próprias configurações como, por exemplo, cada uma das MVs, ofertadas por um provedor de nuvem possui capacidades pré-definidas como *Central Processing Unit* (CPU), memória, armazenamento e largura de banda (MELL; GRANCE, 2011; RODRIGUEZ; BUYYA, 2017). Esses recursos são oferecidos de maneira agrupada e em diferentes pacotes para os usuários como, por exemplo, pacotes por tipos de MVs.

A variedade de pacotes oferecida pelos provedores de nuvem tem como principal objetivo se adequar às necessidades das aplicações dos usuários, que podem ser as mais diversas possíveis. Além disso, cada provedor de nuvem pode oferecer diferentes modelos de cobrança pela utilização destes pacotes de recursos. Tomando

como exemplo o serviço AWS-EC2 da Amazon (EC2, 2017), este provedor de serviço em computação em nuvem pratica um modelo de cobrança por hora de utilização dos recursos. Ao observar a nuvem Microsoft Azure (MICROSOFT, 2017) e a nuvem da RackSpace (RACKSPACE, 2017), o modelo de cobrança dos recursos utilizados difere dos demais. Nestes dois últimos casos, a utilização é cobrada do usuário por minuto gasto, ou seja, por minuto em que o recurso fica ativo na nuvem. Nota-se que cada provedor pode praticar diversos modelos de utilização e de cobrança dos seus recursos ofertados. Essa característica evidencia a complexidade e a heterogeneidade do mercado. Este fato leva ao aumento na dificuldade de escolha de um provedor de nuvem por parte do usuário, seja para a configuração e provisionamento inicial, ou até mesmo para uma migração para um novo provedor que pode atender melhor às suas necessidades.

No modelo de referência proposto pelo NIST, o usuário não gerencia diretamente os recursos da infraestrutura da nuvem física subjacente, somente os recursos virtualizados dessa infraestrutura. Além disso, também possui controle para instalar os seus próprios sistemas operacionais e aplicativos (MELL; GRANCE, 2011). Tal situação é diferente do que ocorre nas nuvens *Platform as a Service* (PaaS) ou *Software as a Service* (SaaS), nas quais o provedor de nuvem, além de realizar o gerenciamento para manter o *data center* funcional, é responsável pelos recursos hospedeiros das plataformas e aplicações (BHARDWAJ; JAIN; JAIN, 2010).

Comumente, uma nuvem IaaS pode ser caracterizada pela entrega de *hardware* e *software* virtualizados como serviços dinamicamente provisionados (BHARDWAJ; JAIN; JAIN, 2010). Para definir os recursos e técnicas aplicadas no oferecimento de IaaS, a Seção 2.2 revisa as tecnologias para virtualização de recursos computacionais e de comunicação.

2.2 VIRTUALIZAÇÃO DE RECURSOS

A virtualização de recursos é fundamental na computação em nuvem, principalmente na entrega de serviços em nuvens IaaS. As técnicas de virtualização tornam possível a criação de um ambiente abstrato, isolado e separado do hospedeiro (BUYYA; VECCHIOLA; SELVI, 2013). Em suma, a virtualização faz uso de uma camada de *software* adicional sobre um Sistema Operacional (SO) ou diretamente sobre o *hardware* adjacente, obtendo uma abstração de múltiplos recursos virtuais, os quais podem ser disponibilizados para os usuários como se fossem recursos físicos do hospedeiro. Dentre as práticas mais usuais de virtualização de recursos computacionais estão as MVs e contêineres, descritos nas Subseções 2.2.1 e 2.2.2, respectivamente. Contudo, recursos de rede (tipicamente RV) também podem ser virtualizados para conectar os recursos computacionais (Subseção 2.2.3).

2.2.1 Máquinas Virtuais (MVs)

As MVs surgiram entre o fim da década de 60 e no início da década de 70. Entretanto, com a redução do custo do *hardware* e com a proliferação dos computadores pessoais, as MVs ficaram restritas a *mainframes* até meados da década de 90 (ROSENBLUM; GARFINKEL, 2005). Na final da década de 90, em função do avanço da capacidade de memória e processamento dos computadores (especialmente a existência de vários núcleos e processadores), a virtualização ganhou um novo impulso (KOSLOVSKI, 2011). A técnica de virtualização de MVs, consiste basicamente em adicionar uma camada de *software* no hospedeiro, e dessa forma obter o suporte para a arquitetura desejada (SMITH; NAIR, 2005). Essa camada de *software* é conhecida como Monitor de Máquina Virtual (MMV) ou hipervisor. Entretanto, havia uma fragmentação das soluções de virtualização que implicava em uma dificuldade na criação de grandes infraestruturas para fornecer serviços de virtualização.

A computação em nuvem, em especial os serviços de IaaS, forneceram uma meio facilitado de provisionamento recursos virtualizados. Atualmente, com o advento da computação em nuvem, MVs são amplamente difundidas e utilizadas (MAHMUD; KOTAGIRI; BUYYA, 2018).

Com a utilização do hipervisor, é possível executar diferentes máquinas em um único computador, isso ocorre pelo fato de que o hipervisor oferece uma visão uniforme do *hardware* subjacente às MVs (ROSENBLUM; GARFINKEL, 2005). O hipervisor possibilita aos administradores suspender, realizar *checkpoint* e resumir a execução das MVs a um estado anterior de execução. Sobretudo, o hipervisor oferece a possibilidade de migração de MVs entre diferentes hospedeiros (ROSENBLUM; GARFINKEL, 2005; GARFINKEL; ROSENBLUM et al., 2003; CLARK et al., 2005). Dessa forma, devido ao fato de que a migração de MV necessitar acesso ao hipervisor para ser realizada, os administradores são os maiores beneficiados com a migração, pois dessa forma é possível realizar a consolidação em único domínio de gerenciamento coerente (CLARK et al., 2005). Entretanto, para fornecer todos esses serviços, um hipervisor deve ser capaz de exportar uma interface de *hardware* para o *software* da MV, de maneira que essa interface seja equivalente ao *hardware* físico subjacente.

Como o tema de virtualização de MVs e a arquitetura da virtualização já foi amplamente abordado pela literatura especializada, por exemplo em (ROSENBLUM; GARFINKEL, 2005; GARFINKEL; ROSENBLUM et al., 2003; SMITH; NAIR, 2005; BARHAM et al., 2003; XU et al., 2014) entre outros, a discussão das técnicas de virtualização dos recursos computacionais (CPU e memória), assim como a arquitetura e questões de desempenho dos hipervisores não são abordadas no escopo deste trabalho.

2.2.2 Contêineres

Os recursos computacionais são comumente virtualizados através de duas tecnologias, sendo Máquinas Virtuais (MVs) a primeira e mais antiga, e recentemente por contêineres. A virtualização de MVs realiza a emulação de todas as funcionalidades da máquina hospedeira. Em contrapartida, um contêiner pode ser classificado como uma versão mais leve de um sistema operacional que executa dentro de um sistema hospedeiro (CELESTI et al., 2016). Dessa forma, os contêineres tem como objetivo reduzir a utilização dos recursos em comparação as MVs, tendo em vista que os contêineres executam instruções nativas de uma CPU, eliminado as eventuais simulações das instruções de uma CPU virtualizada (DUA; RAJA; KAKADIA, 2014; SCHEEPERS, 2014).

O isolamento do processo em questão do restante do sistema operacional é aplicado para todos os processos filhos do primeiro. O isolamento pode ser realizado por *namespaces* (BIEDERMAN; NETWORX, 2006), que é uma funcionalidade do núcleo Linux a qual possibilita que processos diferentes tenham uma visão diferente do sistema operacional. O gerenciamento dos recursos neste tipo de virtualização usualmente pode ser realizada através de grupos de controle (*cgroup*) (XAVIER et al., 2013), através do qual é possível limitar/priorizar o uso de CPUs, memória e uso de I/O por cada grupo, ou seja, por contêiner. Assim, cada contêiner é capaz de executar seu próprio SO, além disso é possível isolar e compartilhar sistema de arquivos de cada contêiner através do *chroot* (DUA; RAJA; KAKADIA, 2014). Essa funcionalidade altera o diretório raiz de um processo e seus filhos para um novo diretório. Porém, o *kernel* do sistema hospedeiro é compartilhado para todos os contêineres por ele hospedado. Dessa forma não é possível executar em um sistema MS-Windows um contêiner GNU/Linux ou Unix e vice-versa (DUA; RAJA; KAKADIA, 2014). De um modo geral, a técnica de virtualização por contêineres realiza a partição dos recursos físicos da máquina hospedeira realizando a criação de diversas instâncias isoladas em espaço de usuário (XAVIER et al., 2013; ZHANG; LU; PANDA, 2016; MERKEL, 2014).

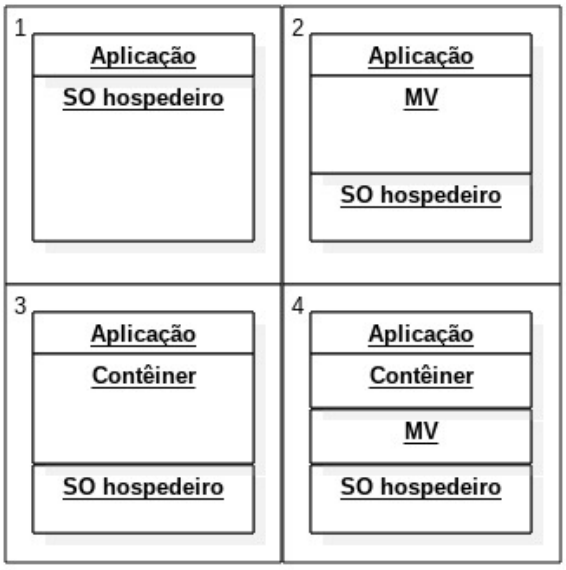
Tendo em vista que um hospedeiro pode abrigar uma quantidade considerável de contêineres (ZHANG; LU; PANDA, 2016), essa técnica de virtualização pode proporcionar um ganho às nuvens IaaS pois possibilita que as aplicações implantadas consumam menor volume de memória principal. Em suma os benefícios dos contêineres em relação as MVs são:

- Tempo de inicialização: por não necessitar emular todo o sistema hospedeiro, o tempo de inicialização de um contêiner é menor quando comparado ao tempo de inicialização de uma MV (SEO et al., 2014).
- Consumo de memória: devido ao compartilhamento do *kernel* entre SO hospe-

deiro e contêineres, estes consomem uma quantidade menor de memória do que as MVs (CELESTI et al., 2016).

A virtualização por contêineres pode ser opcionalmente utilizada em conjunto com MVs. A Figura 1 apresenta como as duas tecnologias podem ser posicionadas, indicando os cenários de utilização de contêineres.

Figura 1 – Utilização de contêineres.



Fonte: O próprio autor.

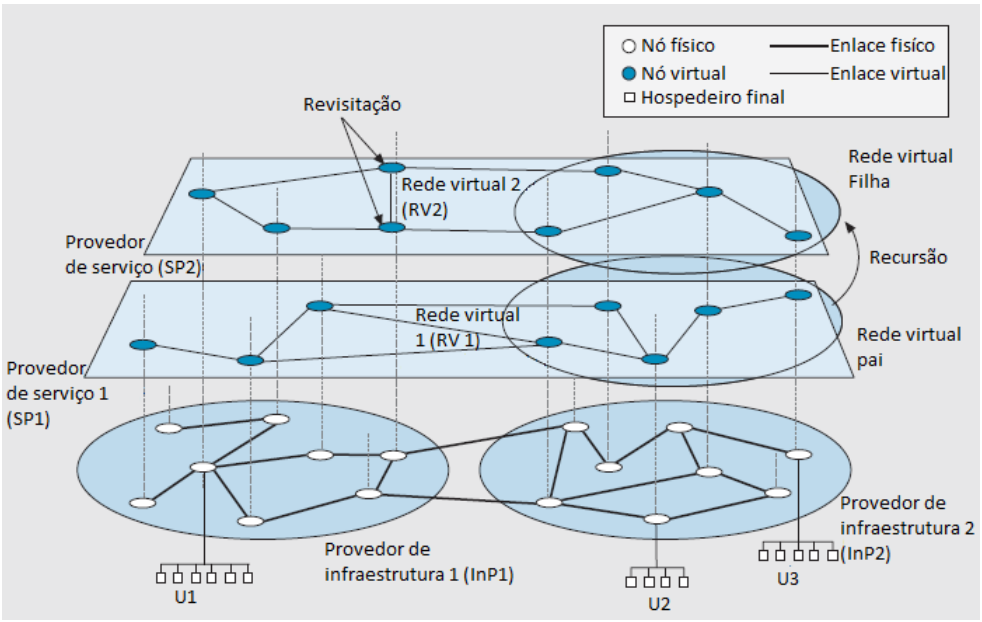
Partindo de um cenário no qual não existe nenhuma camada de virtualização, na primeira composição da Figura 1 (iniciando a leitura pela esquerda), é exemplificado o caso de uma aplicação executando diretamente no SO. Pode-se associar essa composição a um ambiente no qual, geralmente, o usuário é dono do hospedeiro e dessa forma possui acesso direto ao SO. Já na segunda composição (SO, MV e aplicação), observa-se uma camada de virtualização por MV, cenário comum em nuvens IaaS (BHARDWAJ; JAIN; JAIN, 2010). No terceiro exemplo (SO, contêiner e aplicação), é introduzido um contêiner, executando diretamente no SO. Dessa forma, de maneira similar ao apresentado no primeiro exemplo, o usuário também necessita de acesso ao SO do hospedeiro para pode manipular o seu contêiner e consequentemente a sua aplicação. Por fim, o quarto exemplo (SO, MV, contêiner e aplicação) introduz um cenário no qual o contêiner está executando dentro de uma MV. Assim como no segundo exemplo, a última composição apresentada é comumente utilizada em nuvens computacionais.

2.2.3 Redes Virtuais

A virtualização de redes é uma tecnologia fundamental para evolução das arquiteturas de comunicação, auxiliando no avanço contra a ossificação das redes (HANDLEY, 2006). Esse conceito de RVs possibilita que uma rede física seja particionada em várias redes lógicas, de acordo com as necessidades do usuário (CHOWDHURY; BOUTABA, 2009). Existem dois principais atores ou participantes: provedores de infraestruturas (*Infrastructure Providers* (InPs)) e provedores de serviços (*Service Providers* (SP)) (CHOWDHURY; BOUTABA, 2009). Desse modo, em um ambiente de RV, as funções dos tradicionais provedores de serviço (*Internet Service Providerss* (ISPs)) são divididas para os dois atores, os InPs e os SP.

A Figura 2 exemplifica a composição de um ambiente de redes virtuais. De um modo geral, uma RV pode ser definida como um conjunto de múltiplas e heterogêneas arquiteturas de rede, de maneira que essas arquiteturas são provenientes de diferentes SP (CHOWDHURY; BOUTABA, 2009). De acordo com Chowdhury e Boutaba (2009), os InPs são responsáveis por implantar e gerenciar os recursos físicos da rede física subjacente. Por sua vez, os SPs alugam recursos de múltiplos InPs para criar e gerenciar uma ou mais RVs.

Figura 2 – Rede Virtual (RV).



Fonte: Traduzido de (CHOWDHURY; BOUTABA, 2009).

Opcionalmente, um SP pode desempenhar o papel de um InPs virtual e dessa maneira prover os seus recursos para outro SP, o qual irá criar uma RV filha, ou aninhada. Sendo assim, os usuários finais podem se conectar a diferentes RVs por diferentes SP para a utilização dos diferentes serviços.

Quanto as tecnologias utilizadas para a implementação de redes virtuais, inici-

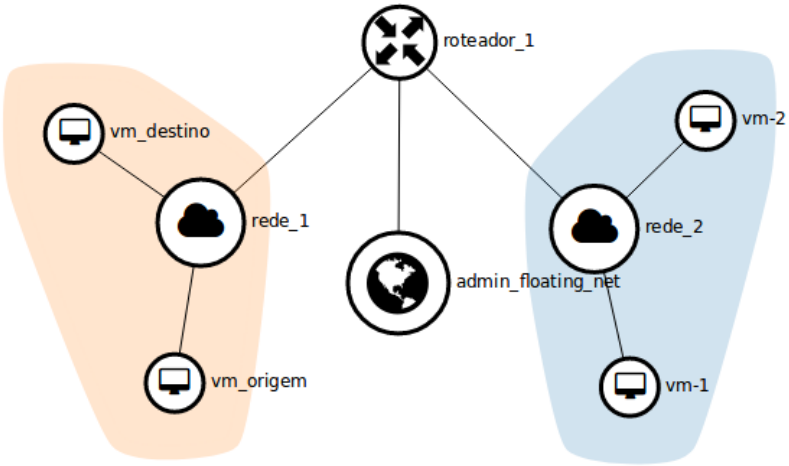
almente, o particionamento de uma rede física em diversas redes virtualizadas ou lógicas era unicamente realizado através da composição de *Virtual Local Area Network*s (VLANs) (TANEMBAUM; WETHERRAL, 2010). Para tal, os *switches* intermediários devem suportar a interpretação das etiquetas de VLAN para realizar a comutação dos pacotes. Essa comutação de pacotes é realizada de acordo com o identificador, previamente configurado para quais portas deve ser encaminhado.

Além das VLANs, a entrega de redes virtualizadas pode ser realizada por VPN. A VPN possibilita aos provedores modelem sua topologia de rede e configurações de segurança como, *e.g.,* regras de *firewall* (ZHANG; CHENG; BOUTABA, 2010). Outra tecnologia que possibilita a entrega de redes virtualizadas é *Software Defined Networks* (SDN). Através de SDN é possível separar o plano de controle do plano de dados de forma programável (JAIN; PAUL, 2013). O provisionamento combinado de RVs com as tecnologias de virtualização de recursos computacionais previamente discutidas (MVs e contêineres) originou as IVs, discutidas na Seção 2.3.

2.3 INFRAESTRUTURA VIRTUAL

Formalmente, uma Infraestrutura Virtual (IV) pode ser definida como um conjunto de recursos virtuais conectados por serviços de comunicação. Tais serviços utilizados para interconectar MVs e contêineres, oferecem roteadores, comutadores e canais de comunicação, todos virtualizados (ANHALT; KOSLOVSKI; PRIMET, 2010; MELL; GRANCE, 2011; BHARDWAJ; JAIN; JAIN, 2010). Ou seja, os elementos de encaminhamento e roteamento também podem ser virtualizados e gerenciados pelo provedor IaaS. Com a virtualização destes componentes de rede, surge a possibilidade deles serem implantados e configurados de acordo com as aplicações utilizadas pelo usuário da nuvem. Na Figura 3 é possível visualizar uma representação gráfica de uma IV.

Figura 3 – Exemplo de uma Infraestrutura Virtual - Representação do OpenStack.



Fonte: O próprio autor.

A IV (Figura 3) foi criada utilizando o gerenciador de nuvens OpenStack (detalhado na Subseção 5.1.1, durante a explicação do protótipo desenvolvido), o qual oferece uma ferramenta para visualização dos recursos alocados de maneira gráfica, conforme ilustrado. É possível observar (Figura 3) que diversas MVs foram alocadas pelo usuário, sendo possível identificar que as MVs são interconectadas pelos canais de comunicação, que formam as sub-redes internas, as quais possuem suas próprias características como, por exemplo, faixas de endereços IPs internos e capacidade de comunicação. As sub-redes, por sua vez, são interconectadas por roteadores virtuais, os quais possuem interfaces de rede que podem ser configuradas pelo usuário contratante do serviço, de acordo com a sua necessidade e também de acordo com pacotes de recursos adquiridos do provedor de serviço. Um serviço de comunicação que pode ser contratado é a disponibilização de IPs flutuantes, ou seja, endereços IP válidos, roteados na Internet, que podem ser movidos dinamicamente entre as MVs (RDO, 2017). Em suma, a composição de todos esses recursos virtualizados e alocados pelo usuário, compõe uma IV.

Dessa forma, tem-se que uma IV é composta por recursos virtualizados alocados pelo usuário, que podem ser MVs, contêineres, rede virtualizada e também serviços gerenciáveis. Cada um dos recursos elencados podem ser individualmente migrados (ou em conjunto) entre *data centers*, regiões ou provedores (JUNIOR; MIERS; KOSLOVSKI, 2017). O escopo do presente trabalho é a migração completa da IV. Entretanto, realizar a migração desses recursos não é uma tarefa trivial.

2.4 MIGRAÇÃO DE RECURSOS ENTRE PROVEDORES DE NUVEM

Realizar a migração de recursos computacionais virtualizados entre provedores IaaS é um desafio em aberto e pouco abordado na literatura especializada (os trabalhos relacionados são revisados na Seção 3.1). Cada elemento que compõe uma IV pode ser migrado separadamente, entretanto cada um possui características particulares as quais devem ser consideradas no momento da migração. Existem diversas unidades de migração em nuvens IaaS, sendo elas: processos / contêineres, MVs e IVs. Entretanto, além dessas unidades, ainda pode-se considerar a realização da migração de aplicações entre diferentes provedores de nuvem. Dessa forma, nessa seção são discutidos os pontos positivos e negativos da migração de cada uma dessas unidades, iniciando pela camada mais alta ou seja a de aplicação.

2.4.1 Migração de aplicações

Em computação em nuvem usualmente as aplicações executam sobre componentes virtualizados, os quais podem ser MVs e contêineres, conforme apresentado na Figura 1. Dessa forma, realizar a migração de uma aplicação consiste em mover os

dados e os processos entre diferentes provedores de nuvem, sem realizar nenhuma alteração no modelo da aplicação ou nos recursos de hospedagem.

Um dos pontos positivos da migração de aplicações é a abstração do gerenciamento da infraestrutura subjacente à aplicação. Dessa forma, todo esse gerenciamento fica sob responsabilidade do provedor de nuvem, enquanto o usuário preocupa-se somente com o funcionamento e gerenciamento de sua aplicação. Sendo assim, as técnicas de migração de aplicações podem ser aplicadas para serviços fornecidos em nuvens PaaS e SaaS.

Por outro lado, ao delegar a configuração da rede para o provedor de nuvem, o usuário fica restringido ao nível de abstração oferecido pelo provedor. Outro ponto que deve ser considerado na migração de aplicações é que a mesma deve ser independente de bibliotecas específicas do sistema operacional (TRAN et al., 2011), pois estas dependências podem ocasionar erros no momento de restaurar a aplicação no servidor de destino, o qual pode não possuir as mesmas bibliotecas e configurações do servidor de origem. Uma maneira de contornar essa problemática é implantar a aplicação em MVs e quando necessário realizar a migração das MVs.

2.4.2 Migração de Máquinas Virtuais (MVs)

Através da técnica de virtualização de MVs, os provedores de nuvens conseguem fornecer para os usuários a ideia de um hospedeiro completamente privado. Neste cenário, os provedores se beneficiam com migração de MVs. Um dos principais benefícios que a migração pode trazer para o provedor é o balanceamento de carga em *data centers* (RUCK et al., 2017). Entretanto, a migração de MVs pode ser utilizada para a composição de nuvens híbridas, ou seja, a migração de dados não críticos para nuvens públicas.

Em contrapartida, o desempenho das aplicações hospedadas é afetado durante a migração de MVs, caracterizando um problema reconhecido (AKOUSH et al., 2010; DESHPANDE; KEAHEY, 2017). Um fator importante que deve ser considerado é o armazenamento virtualizado, necessário para realizar o processo de *live migration*, sem interromper a execução da aplicação hospedada (CLARK et al., 2005) (MASHTIZADEH et al., 2014). Para realizar a migração entre nuvens de pequeno porte, um armazenamento compartilhado pode ser utilizado como uma solução. Entretanto, quando se trata de nuvens de larga escala essa situação pode não ser suficiente para resolver o problema, devido ao elevado volume de dados trafegados.

Existem algumas propostas para melhorar o desempenho da migração de MVs (CLARK et al., 2005; HINES; DESHPANDE; GOPALAN, 2009; ZHANG; FU; YAHYAPOUR, 2017). Essas propostas buscam a melhora de desempenho através da adição

de mecanismos como a pré-cópia (CLARK et al., 2005) e pós-cópia (HINES; DESHPANDE; GOPALAN, 2009). Todavia, esses mecanismos são focados na migração de MVs em *data centers* privados (ZHANG; FU; YAHYAPOUR, 2017) e não na migração de MVs através de provedores diferentes, *i.e,* através da Internet. Além disso, para realizar a migração de MVs, é necessário ter acesso ao hipervisor, restringindo assim o processo de migração somente para o administrador da nuvem. Dessa forma o usuário ainda está suscetível ao *vendor lock-in*. Uma alternativa à virtualização de MVs podem ser os contêineres que, por serem executados em espaço de usuário, removendo a barreira da migração imposta pelas MV.

2.4.3 Migração de contêineres

A técnica de virtualização de contêineres difere da técnica de virtualização por MV pois os contêineres compartilham o mesmo núcleo do SO. Assim, migrar contêineres consiste em mover processos em execução de um hospedeiro para outro (semelhante ao discutido na Seção 2.4.2). Ou seja, o processo deve ser desacoplado do SO e disponibilizado de forma integral em um novo hospedeiro. De forma que o seu estado interno é transferido e reinicializado no SO de destino (JUNIOR; MIERS; KOSLOVSKI, 2017).

Um dos pontos positivos da migração de contêineres é que estes possuem todos os componentes necessários como, *e.g.,* arquivos e bibliotecas. Além disso, na migração de contêineres não é necessário ter acesso ao hipervisor, possibilitando assim que o próprio usuário realize a migração sem a necessidade de contato com o provedor de serviço, evitando assim o chamado *vendor lock-in*.

Em contrapartida, a virtualização baseada em contêineres ainda é um processo novo e em desenvolvimento nos provedores de nuvem (ZHAO; ZHOU, 2014; JAMSHIDI; AHMAD; PAHL, 2013; MEDINA; GARCÍA, 2014). Ou seja, nem todas as ferramentas de gerenciamento de contêineres possuem suporte para a migração.

2.4.4 Migração de Infraestruturas Virtuais (IVs)

O processo de migração de IVs consiste em mover uma aplicação juntamente com a camada de virtualização que a hospeda. Adicionalmente aos recursos computacionais, a topologia da rede (enlaces, roteadores e *switches*) virtualizados são conjuntamente migrados. É importante ressaltar que no cenário de computação em nuvem, a camada de virtualização pode ser MVs, contêineres ou contêineres executando dentro de MVs.

Ao mover uma IV, as configurações de rede e da aplicação hospedada são mantidas inalteradas. Para tanto, a RV deve ser pré-configurada no provedor de ser-

viço de destino de maneira que a configuração da mesma não seja alterada, por exemplo, endereços de IP privados. Além dos desafios de migrar contêineres e MVs, a migração da rede ainda é um desafio em aberto (ZHAO; ZHOU, 2014; JAMSHIDI; AHMAD; PAHL, 2013; MEDINA; GARCÍA, 2014). Dessa forma esse trabalho se propõe atacar o problema de migração de IVs através de uma proposta para esse problema e uma implementação desta proposta (protótipo) focada em nuvens OpenStack.

2.4.5 Discussão sobre migração de recursos virtualizados

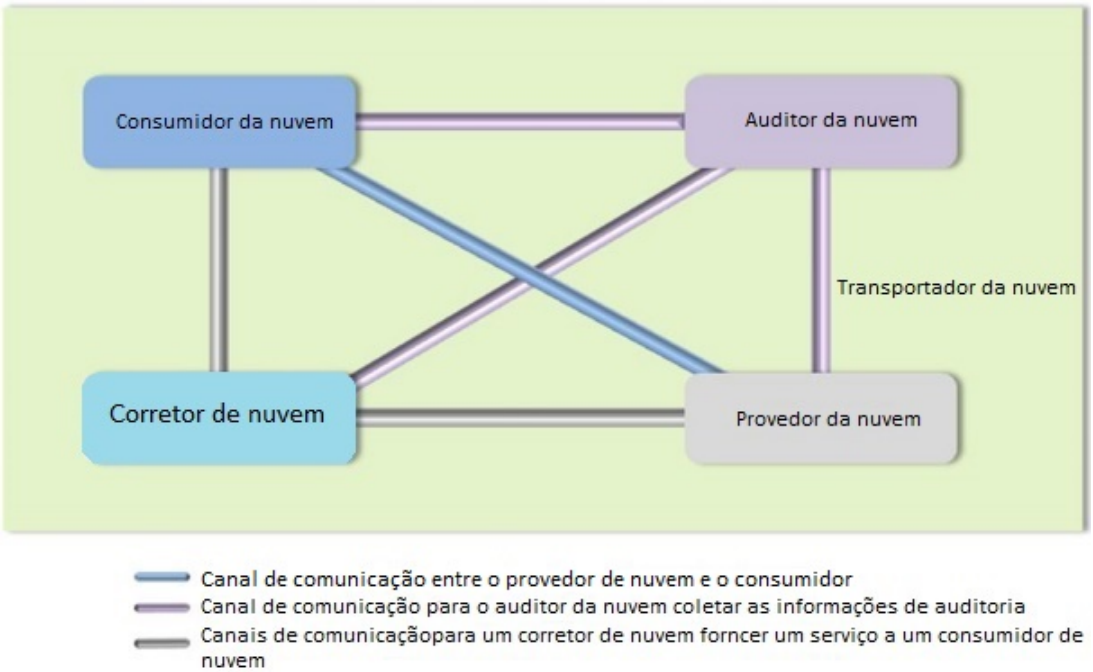
Realizando uma análise da migração de cada uma dessas unidades assim como seus desafios, pontos positivos e negativos, esse trabalho foca na migração de IVs compostas por contêineres. Ainda, o foco desse trabalho está na migração dos cenários nos quais o contêiner está executando dentro de uma máquina virtual ou diretamente sobre um sistema operacional hospedeiro. Os dois cenários são utilizados para compor os serviços oferecidos atualmente pelos provedores de nuvens IaaS. O cenário com execução do contêiner diretamente sobre o *hardware* foi descartado, uma vez que exige privilégios de gerenciamento que podem não ser oferecidos por nuvens IaaS públicas.

Caso os contêineres sejam disponibilizados sobre o SO, acesso administrativo é necessário para realizar a migração, caracterizando uma execução em nuvens privadas. Em nuvens públicas, o acesso ao *hardware* e SO hospedeiros não é permitido. Assim, o único cenário aplicável compreende a utilização de MVs intermediárias para hospedar os contêineres. De maneira geral, a solução proposta por este trabalho (Capítulo 4) pode ser utilizada desde que o usuário possua acesso ao SO oferecido pelo provedor da nuvem IaaS, caso contrário a migração não poderá ser realizada.

2.5 ATORES DE COMPUTAÇÃO EM NUVEM

Em computação em nuvem, quatro atores principais são identificados, de acordo com os papéis desempenhados (SOKOL; HOGAN, 2013). Este atores são consumidor da nuvem, provedor da nuvem, auditor da nuvem, corretor da nuvem e transportador da nuvem (*cloud carrier*). A Figura 4 lista os relacionamentos entre os atores de computação em nuvem. Ainda de acordo com Sokol e Hogan (2013) podem existir consumidores e provedores específicos para cada uma das três modalidades de serviços (IaaS, PaaS e SaaS), entretanto este trabalho foca somente nos corretores de nuvem.

Figura 4 – Papéis dos participantes em computação em nuvem.



Fonte: Traduzido de: (SOKOL; HOGAN, 2013).

O consumidor de nuvem é o usuário final, o qual utiliza os serviços e recursos da nuvem através de um provedor de serviço. Neste cenário, o usuário pode ser desde uma pessoa até uma organização, sendo que o usuário é tarifado pelo provedor de acordo com os recursos utilizados (SOKOL; HOGAN, 2013).

Já observando os provedores, tem-se que estes podem ser organizações ou entidades as quais disponibilizam os serviços para os consumidores. Dessa maneira os provedores são responsáveis pelos aspectos técnicos da infraestrutura necessária para o provisionamento destes serviços. Um provedor de IaaS é responsável por gerenciar a infraestrutura da nuvem fornecida para o consumidor; ele também pode prover processamento, armazenamento e outros recursos computacionais (SOKOL; HOGAN, 2013).

Segundo Sokol e Hogan (2013), os auditores de nuvem tem por responsabilidade realizar a coleta de dados das nuvens e assim realizar avaliações independentes. Nestas avaliações são revisadas questões de segurança, desempenho e aderência ao *Service Level Agreement* (SLA) dos serviços fornecidos pelos provedores. Dessa forma, os auditores são os atores responsáveis por aferir a conformidade do uso, desempenho e a entrega dos serviços de computação em nuvem. Por sua vez, os transportadores de nuvem tem como papel fornecer a conectividade e o transporte de serviços de nuvem entre os provedores e os consumidores. Eles são responsáveis por fornecer aos consumidores acesso aos serviços fornecidos pelos provedores.

Por fim, os corretores de nuvem são responsáveis por gerenciar o uso e a entrega dos serviços de nuvem (JUNIOR; MIERS; KOSLOVSKI, 2017). Dessa forma, o corretor de nuvem pode ser responsável por intermediar o relacionamento entre o provedor de nuvem e o consumidor. Em outras palavras, os corretores de nuvem fornecem um único ponto de entrada para gerenciar múltiplos serviços de nuvem. Neste cenário dos atores de nuvem e tendo em vista que os provedores de nuvem são relutantes em fornecer padrões e meios de portabilidade e interoperabilidade (ZHANG; WU; CHEUNG, 2013), a proposta deste trabalho pode ser classificada como um corretor de nuvem.

2.6 CONSIDERAÇÕES PARCIAIS

Apesar de ser um conceito um pouco recente, a computação em nuvem faz uso de diversas tecnologias e aplicações já utilizadas e consolidadas, como por exemplo a virtualização de MVs. Entretanto, com a evolução e o avanço da tecnologia, novas soluções foram propostas, como a virtualização de redes e por contêineres.

A virtualização por contêineres, empregada junto com a técnica de virtualização de redes pode compor uma IV. É nesse cenário de IVs, compostas por MVs e dentro destas diversos contêineres, os quais executam as aplicações dos usuários, que este trabalho está centrado. O trabalho foca na migração dessas IVs entre diferentes provedores de nuvens IaaS e levando em consideração os desafios da migração de uma IV e propondo uma solução que avança no estado da arte através do aumento da portabilidade entre provedores de nuvens. A definição da problemática estudada neste trabalho é apresentada no Capítulo 3.

3 DEFINIÇÃO DO PROBLEMA

Este capítulo tem como objetivo apresentar os trabalhos relacionados ao objeto de pesquisa, assim como a problemática da migração em ambientes de computação em nuvem. O restante deste capítulo é organizado da seguinte maneira: na Seção 3.1 é feito um levantamento dos trabalhos relacionados à migração em computação em nuvem. Essa migração pode ser de MVs, contêineres e aplicações. Na Seção 3.2 são discutidos e apresentados os problemas encontrados na realização da migração de IVs em computação em nuvem. Por fim, a Seção 3.3 discute as considerações parciais do capítulo.

3.1 TRABALHOS RELACIONADOS

A migração de MVs em um ambiente de computação em nuvem já foi abordada na literatura especializada, entretanto a migração de IVs é um tema ainda não resolvido e amplamente estudado (ZHAO; ZHOU, 2014; JAMSHIDI; AHMAD; PAHL, 2013). Dessa forma, foram elencados os trabalhos relacionados com base em uma pesquisa bibliográfica. A pesquisa foi realizada utilizando as bases de dados ScienceDirect, IEEE e Google Scholar. Para a pesquisa, foram utilizadas as seguintes palavras chaves em português e as suas variantes em inglês: Computação em Nuvem, Infraestrutura Virtual (IV), Migração, Máquina Virtual (MV) e contêineres. Foram considerados somente os artigos publicados nos últimos quatro anos, dando preferência para trabalhos mais recentes deste ano, de maneira que os mesmos tenham relevância para o objeto de pesquisa deste trabalho. Consequentemente, os trabalhos que abordam o tema de migração em computação em nuvem foram selecionados, de maneira que foi selecionado ao menos um trabalho que aborde a migração de cada unidade de migração definida na Seção 2.4. Os trabalhos selecionados abordam o tema da migração nas três camadas de serviços de computação em nuvem proposta pelo NIST, sendo elas IaaS, PaaS e SaaS. Alguns estudos abordam somente a migração de aplicações (CARRASCO; DURÁN; PIMENTEL, 2017; RISTOV; KOSTOSKA; GUSEV, 2014; ALI; MOAWAD; HOSNI, 2017), sem considerar a infraestrutura virtualizada subjacente, tampouco a virtualização de rede.

Iniciando pela migração de aplicações em computação em nuvem, o trabalho apresentado por Carrasco, Durán e Pimentel (2017) propõe um algoritmo de orquestração. Esse algoritmo tem como finalidade realizar *live migration* de aplicações em ambientes de nuvens computacionais, usando o padrão *Topology and Orchestration Specification for Cloud Applications* (TOSCA), e é capaz de migrar cada componente

da aplicação separadamente. Apesar de ser um algoritmo agnóstico ao provedor de nuvem e utilizar um padrão reconhecido, o algoritmo proposto no artigo não leva em consideração a migração da rede subjacente. Por ser um algoritmo de migração de aplicações, o trabalho tem um foco em nuvens PaaS mas pode ser aplicado para realizar a migração de aplicações de nuvens PaaS para nuvens IaaS.

O processo de migração de aplicações também é abordado em (RISTOV; KOSTOSKA; GUSEV, 2014). Neste artigo os autores descrevem o P-TOSCA, que é um modelo utilizado especificamente em nuvens PaaS para mover aplicações e dados através de diferentes provedores de nuvem. O processo de migração ocorre com a utilização do *Cloud Service Archive* (CSAR), o qual emprega um arquivo compactado que contém todos os metadados e definições utilizadas pelo TOSCA e que são necessários para implantar uma aplicação (STANDARD, 2013). Dessa forma, todas as especificações e artefatos utilizados para a implantação da aplicação são colocados dentro de um CSAR. Após esse passo, o CSAR é enviado para o provedor de nuvem de destino e é processado pela sua plataforma. Na sequência as instâncias são criadas de acordo com o plano de execução especificado no documento de definições do TOSCA. Ao analisar a proposta, nota-se uma limitação pelo fato de que o provedor de nuvem deve ser capaz de replicar o ambiente da aplicação e permitir a implantação da aplicação proveniente de outro provedor de nuvem, ou seja, o provedor de nuvem de destino da migração deve ser capaz de processar os arquivos no formato definido pelo TOSCA.

Ainda referente a migração de aplicações em ambientes de nuvens computacionais, existe a abordagem proposta por (ALI; MOAWAD; HOSNI, 2017). Os autores propuseram o *Cloud Interoperability Broker* (CIB), um corretor de nuvem que atua como um intermediário para promover a migração de aplicações entre diferentes provedores de nuvens SaaS. A migração da aplicação para outros provedores de nuvens é realizada por meio de API padronizadas, as quais devem ser expostas por cada provedor de nuvem, de maneira que seja possível extrair os dados das aplicações no provedor de nuvem de origem da migração e transformá-los em um formato, o qual possa ser utilizado em outro provedor de nuvem. Novamente, essa abordagem é especifica para provedores de nuvens da camada SaaS e não considera a rede subjacente à aplicação e que pode implicar em *vendor lock-in*.

O tema de migração em nuvens IaaS é abordado em (DUGGAN et al., 2017) no qual é apresentado uma abordagem para realizar *live migration* de MVs em nuvens IaaS. Essa abordagem propõe um agente de aprendizado autônomo que possui a capacidade de agendar e de realizar a migração de MVs em um período apropriado. Em outras palavras, tem o objetivo de balancear a utilização dos recursos de rede e o agente possui capacidade de tomar uma decisão de migrar baseado, por exemplo,

na sazonalidade do uso da rede. A abordagem proposta realiza a migração de MV em nuvens IaaS, entretanto a mesma não se preocupa com as configurações de rede que as MVs possuem, dessa forma após realizar a migração pode ser necessário realizar toda a configuração da rede subjacente à aplicação.

Uma outra abordagem para realizar *live migration* de MVs em nuvens IaaS é apresentada em (TSAKALOZOS et al., 2017). Nesse trabalho, o objetivo é realizar a migração de uma fila de tarefas de migração definidas pelo administrador da nuvem, assim como também gerenciar todos os recursos de maneira que as migrações ocorram sem violar o SLA acordado e nem afetar os requisitos de *Quality of Service* (QoS). Sendo assim, a proposta considera a migração de todas as MVs envolvidas, sendo que o agendamento da migração deve ocorrer em um período de baixa atividade nas MVs. Esse método foca somente na migração de MVs e não aborda os contêineres nem a rede. Além disso, é uma proposta voltada para o administrador da rede, o que não auxilia na prevenção do *vendor lock-in*.

Ainda referente a migração de MVs, em (ZHANG; FU; YAHYAPOUR, 2017) é proposto o CBase, um *framework* para realizar *live migration* de MVs e melhorar o desempenho das migrações de MVs em ambientes de nuvens computacionais. Resumidamente, a proposta do artigo é a criação de um repositório de imagens de MVs centralizado entre *data centers*. Estas imagens são acessadas por diversos provedores de nuvens. Dessa forma, como um repositório de imagens, o tempo de migração é reduzido facilitando assim a realização de *live migration* entre diferentes provedores de nuvens. A melhora no desempenho ocorre devido ao fato de que os provedores de nuvens podem rapidamente entregar uma MV através da clonagem dessa imagens localizadas no CBase.

Em (ZHAO et al., 2017) é desenvolvido um mecanismo para a migração de IVs para *Global Environment for Network Innovations* (GENI). Essa migração de IV é realizada utilizando a tecnologia de SDN[1]. O processo de migração ocorre através da clonagem das tabelas de fluxo dos *switches* da nuvem de origem para os *switches* da nuvem de destino. Assim, os hospedeiros não são migrados, são somente reconectados ao novo provedor de nuvem, o qual passa a fazer o encaminhamento do tráfego. Uma restrição da migração utilizando SDN é que a mesma deverá ocorrer dentro de um mesmo domínio SDN, no qual seja possível alterar as tabelas de fluxo dos *switches*.

A Tabela 1 resume os trabalhos relacionados e apresenta para cada um dos trabalhos as características dos mesmos. As características consideradas na elabora-

[1] SDN é uma arquitetura de rede a qual faz uso de APIs para permitir que programadores de rede definam e configurem a maneira que os dados que trafegam na rede sejam tratados (KIRKPATRICK, 2013).

ção da Tabela 1 foram se os trabalhos abordam a migração da rede, de contêineres, de MVs ou da aplicação. Também foi considerado em qual tipo de serviço da nuvem os trabalhos se encaixam (IaaS, PaaS ou SaaS).

Tabela 1 – Trabalhos relacionados

Artigo	Rede	Contêineres	MVs	Aplicação	Camada de serviço
(RISTOV; KOSTOSKA; GUSEV, 2014)	Não	Não	Sim	Sim	PaaS
(CARRASCO; DURÁN; PIMENTEL, 2017)	Não	Não	Não	Sim	IaaS e PaaS
(ALI; MOAWAD; HOSNI, 2017)	Não	Não	Não	Sim	SaaS
(DUGGAN et al., 2017)	Não	Não	Sim	Sim	IaaS
(TSAKALOZOS et al., 2017)	Não	Não	Sim	Sim	IaaS
(ZHANG; FU; YAHYAPOUR, 2017)	Não	Não	Sim	Sim	IaaS
(ZHAO et al., 2017)	Sim	Não	Sim	Sim	IaaS

Fonte: O próprio autor

Observando a Tabela 1, pode-se constatar que as soluções listadas não possuem todas as características levantadas nessa revisão. Logo, há um problema para o qual não foi identificada uma solução que atenda plenamente.

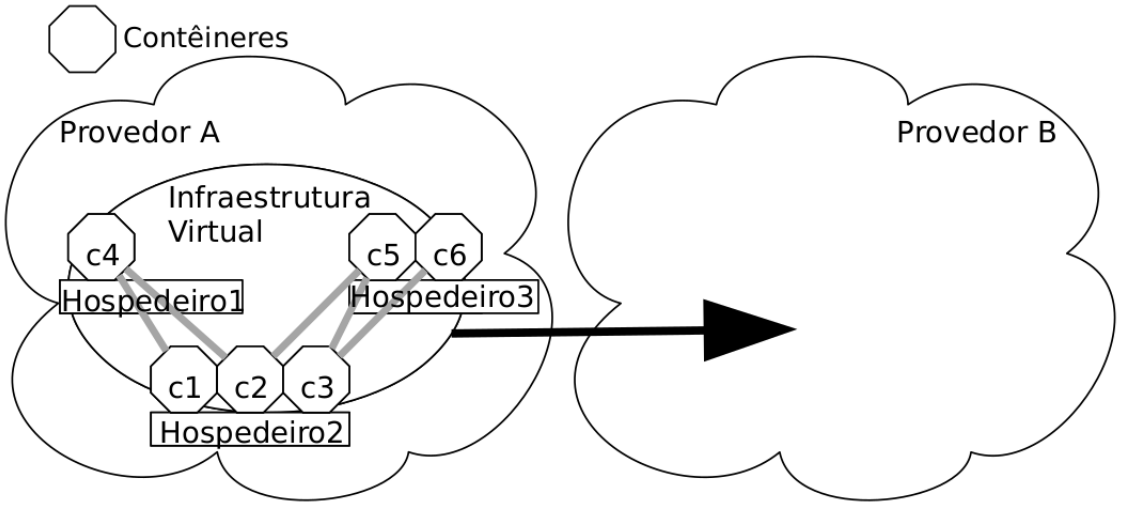
3.2 PROBLEMÁTICA DA MIGRAÇÃO DE IVs

Além dos já conhecidos problemas e desafios da migração de MVs e contêineres, um mecanismo que realize a migração de uma IV deve levar em consideração as configurações da rede virtual utilizada pela mesma. Desse modo, para realizar a migração de uma IV, diversas tarefas devem ser realizadas, de maneira que algumas requerem uma comunicação direta tanto com o provedor de nuvem de origem e com o provedor de nuvem de destino, essa iteração deverá ser realizada por meio de APIs para poder acessar os ambientes de ambos os provedores de nuvem. Essas tarefas são propostas e apresentadas no Capítulo 4.

Conforme discutido no Capítulo 2, uma IV é composta por vários componentes, de maneira que cada um destes podem ser migrados. Realizar a migração de uma IV poderia ser feita considerando somente as aplicações que estejam executando dentro de MVs. Entretanto, para realizar a migração de MVs faz-se necessário um acesso ao hipervisor. Sendo assim, em um ambiente de computação em nuvem, um usuário não poderia realizar a migração de toda a sua IV, pois usualmente o provedor não fornece acesso ao hipervisor no qual podem estar hospedadas diversas MVs de diferentes usuários. Esse tipo de migração beneficia somente ao administrador da nuvem, o qual pode utilizar o recurso de migração para, por exemplo, reduzir os seus custos com energia através de técnicas de consolidação (BARHAM et al., 2003). Contudo, o usuário continuará preso ao seu provedor de nuvem e sem uma maneira fácil de

migrar toda a sua IV (v*endor lock-in*), que já está executando, para outro provedor de nuvem o qual possa atender melhor as suas necessidades. Neste cenário, uma migração em espaço do usuário torna-se necessária. Neste cenário, a virtualização de contêineres pode ser vista como uma possível solução para este problema. Sumarizando, o cenário de migração pode ser representado pela Figura 5, na qual é possível observar uma IV composta por diversos hospedeiros. Nestes hospedeiros encontram-se diversos contêineres, os quais juntamente com a rede virtualizada são migrados para o provedor de destino.

Figura 5 – Problemática da migração.



Fonte: O próprio autor.

Conforme discutido na Seção 2.2.2, a técnica de virtualização de contêineres possibilita que o usuário tenha total controle sobre as suas aplicações e dessa forma, também dá a possibilidade de migrá-las conforme a sua necessidade. Assim, um mecanismo de migração que objetive atender a portabilidade proposta pelo NIST (LIU et al., 2011) e que seja agnóstico ao provedor de nuvem, poderá fazer uso desta técnica de virtualização e propor ao usuário a capacidade de migrar toda a sua IV.

Além da questão de acesso ao hipervisor, que se faz necessária para a migração de MVs, deve-se considerar a migração da rede virtualizada que também compõe a IV. Dentro deste assunto, deve-se levar em consideração dois aspectos importantes: acesso na rede interna e acesso via rede externa. O primeiro aspecto que deve ser considerado é o da rede interna, ou seja, dos canais de comunicação que interligam as aplicações internamente e as configurações de endereços de IP fixos. Ao migrar uma IV, a configuração da rede interna deverá ser mantida, de maneira que, por exemplo, duas aplicações que estejam se comunicando continuem a troca de dados mesmo após a realização da migração da IV para o provedor de nuvem de destino.

Assim, como forma de facilitar a migração, as configurações da rede interna como, por exemplo, endereços de IP locais deverão ser mantidos.

Outro aspecto que deve ser considerado, é referente às configurações de rede externa, ou seja, os pontos nos quais se tem acesso a Internet. Neste aspecto, é notório que cada provedor de nuvem possua sua própria faixa de endereços IP públicos ou flutuantes. Dessa forma, realizar a migração de uma IV que mantenha todos os seus IP flutuantes não é possível de ser realizada. Entretanto, o tráfego de rede proveniente da Internet para a IV deve ser mantido, de maneira que uma comunicação que tenha sido aberta entre um hospedeiro externo à IV e outro pertencente à IV não se perca. Nesse cenário, devem ser consideradas as soluções que redirecionem o tráfego de rede para a nova localização da IV.

Apesar de ser um grande desafio tratar com as conexões externas da IV, existem soluções para tratar desse problema como apresentado em (RAJAGOPALAN et al., 2012) e (WOOD et al., 2011), os quais propõem a manipulação do protocolo *Border Gateway Protocol* (BGP). A manipulação é realizada de modo a obter a resposta de um segundo canal de comunicação quando o canal de comunicação principal deixar de responder. Dessa forma, por ser um assunto amplo e também já estudado pela comunidade, tratar deste problema neste trabalho encontra-se fora do escopo de pesquisa.

3.3 CONSIDERAÇÕES PARCIAIS

Como pode-se observar na Tabela 1, a migração em computação em nuvem é um tema atual e ainda em aberto. Entretanto, nota-se uma lacuna de pesquisa para tratar sobre migração de IVs, ou seja, poucos trabalhos abordam a migração combinada das configurações de rede e aplicações. Logo, o presente trabalho introduz uma proposta de mecanismo para realizar a migração de IVs que seja agnóstico e independente do provedor de nuvem.

De modo objetivo, esse trabalho foca na migração de IV, na qual as aplicações executam dentro de contêineres, os quais estão hospedados em MVs e interconectados por canais de comunicação, ou seja, uma rede virtualizada, conforme indicado na Figura 5. A migração de MVs não é abordada pois a mesma não oferece portabilidade total ao usuário e tampouco. Por fim, o redirecionamento do tráfego externo que está sendo encaminhado para dentro da IV é considerado fora do escopo principal.

4 PROPOSTA DE UM MECANISMO PARA A MIGRAÇÃO DE IVs

Este capítulo tem como objetivo descrever uma proposta para a realização da migração de uma IV entre diferentes provedores, diante da definição do problema apresentada no Capítulo 3. O mecanismo, denominado MigraVI, atua como um corretor de nuvem, agindo como um intermediador no processo de migração. Dessa forma, esse mecanismo deve ter acesso à nuvem de origem e de destino da migração. Ao obter esse acesso, o MigraVI realiza um levantamento da rede da nuvem de origem, com o objetivo de replicar a RV na nuvem de destino e então realizar a migração dos contêineres. Essa proposta é uma solução genérica e agnóstica ao provedor de IaaS, e pode ser implementada de acordo com a necessidade ou nuvem utilizada. Entretanto, uma prova de conceito da implementação é especificada no Capítulo 5.

O restante deste capítulo está organizado da seguinte maneira: a Seção 4.1 descreve o cenário de migração no qual este trabalho está focado, enquanto a Seção 4.3 apresenta a arquitetura proposta Na sequência, a Seção 4.2 apresenta os requisitos funcionais e não funcionais levantados para o mecanismo que gerencia a migração de IVs. Já a Seção 4.4 apresenta e discute os passos necessários propostos para a realização da migração de uma IV entre provedores de nuvem distintos. A Seção 4.6 descreve os planos de testes que são utilizados para avaliar o mecanismo proposto.

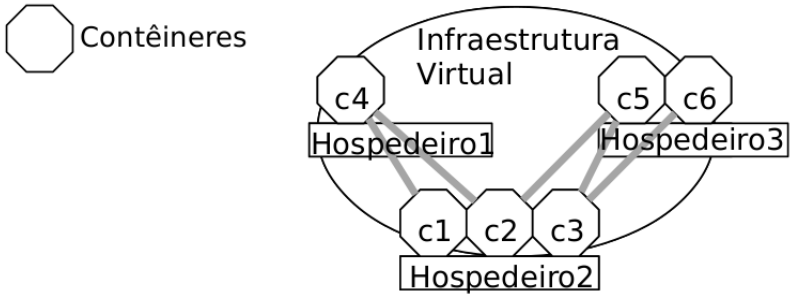
4.1 CENÁRIO DE MIGRAÇÃO DE IVs

O objetivo deste trabalho é realizar a migração de IVs entre diferentes provedores, de maneira que o usuário tenha total capacidade de realizar essa migração sem depender de nenhuma interação com o seu provedor de nuvem. Dessa forma o cenário de execução deste trabalho é composto por duas nuvens, ou seja, nuvem de origem localizada em um *Provedor A* e a nuvem de destino, localizada no *Provedor B* e um mecanismo responsável por realizar a migração da IV da nuvem de origem para a nuvem de destino.

Na nuvem de origem, deverá existir uma IV previamente configurada, a qual será migrada para a nuvem de destino. A IV é composta por um conjunto de hospedeiros, os quais são responsáveis por hospedar diversos contêineres. Esse conjunto de recursos virtualizados são interconectados por uma rede privada e virtualizada. A Figura 6 representa a composição da IV: um conjunto de hospedeiros (*Hospedeiro1, Hospedeiro2* e *Hospedeiro3*) que possuem alocados os contêineres (*c1, c2, c3, c4, c5* e *c6*). Os contêineres são interconectados por enlaces virtualizados, caracterizando

uma rede privada. Para exemplificar foram utilizados três hospedeiros e cinco contêineres, entretanto é importante ressaltar que podem existir diversos hospedeiros e diversos contêineres, tanto quantos o usuário possuir capacidade de alocar. Ou seja, a quantidade de recurso virtualizados não é um limitador deste trabalho.

Figura 6 – Exemplo de IV composta por contêineres hospedados em MVs conectadas por uma VPN.



Fonte: O próprio autor.

O mecanismo proposto age como um corretor de nuvem, ou seja, um intermediador na realização da migração entre os diferentes provedores. Dessa forma, o mecanismo é responsável por realizar a migração de uma IV hospedada em um provedor para outro provedor de destino escolhido pelo usuário. Neste contexto, há algumas hipóteses de situações neste cenário que são discutidas, assim como os passos necessários para realizar a migração de uma IV entre provedores diferentes.

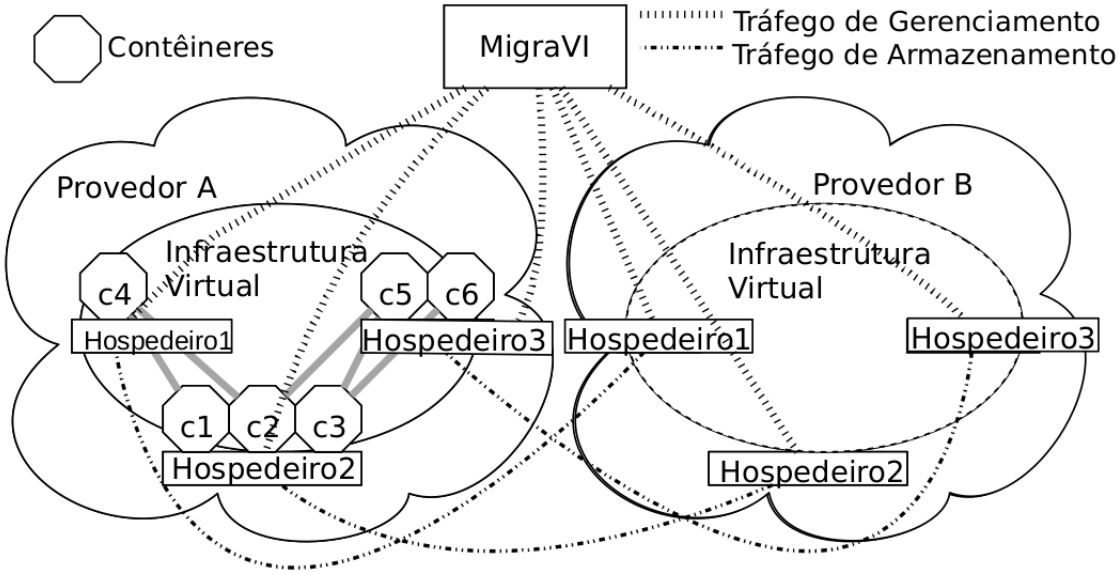
Para realizar o processo de migração, o MigraVI necessita dos dados de autenticação e autorização para realizar o acesso nas nuvens de origem e destino. Consequentemente, o usuário deverá ter total confiança no mecanismo, conforme é comum na configuração de corretores de nuvens já existentes (GROZEV; BUYYA, 2014; TOOSI; CALHEIROS; BUYYA, 2014; ZHANG; WU; CHEUNG, 2013; MANSOUR et al., 2016). Adicionalmente, para que o mecanismo tenha acesso as MVs e assim consiga realizar a migração dos contêineres, as chaves *Secure Shell* (SSH) de acesso às IVs devem ser fornecidas.

É importante ressaltar que o mecanismo proposto não cria e nem gerencia as contas de usuários nos provedores de nuvem de origem e destino. Tal gerenciamento dos dados de autenticação e autorização fica sob responsabilidade do próprio usuário da nuvem. Similarmente, questões relacionadas ao SLA e aos dados de cobrança estão fora do escopo deste trabalho e não são tratados pelo mecanismo aqui proposto.

Conforme é possível observar na Figura 7, após o mecanismo obter acesso as nuvens de origem e de destino, é realizada um levantamento das informações da rede e das MVs da nuvem de origem. Essa leitura tem como objetivo levantar as informações da rede e dos hospedeiros da nuvem de origem, *e.g.,* informações dos roteadores, *switches*, canais de comunicação, endereços de IP, tipo de hospedeiros e imagens utilizadas para criar as MVs. De posse destas informações, o próximo passo é

preparar a nuvem de destino, nesse momento os hospedeiros são recriados de acordo com as informações obtidas na leitura realizada da nuvem de origem.

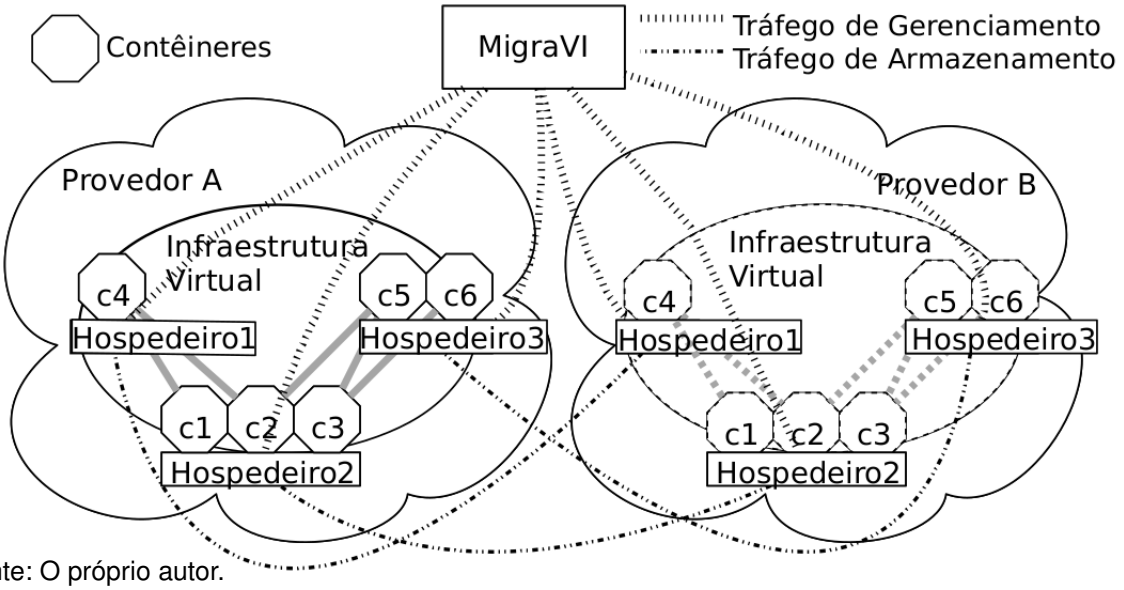
Figura 7 – Preparação do cenário de migração.



Fonte: O próprio autor.

Conforme ilustrado na Figura 8, após criado o hospedeiro, a RV é configurada ou seja, os recursos de rede são recriados e os endereços de IP são atribuídos aos hospedeiros. Com isso, estes hospedeiros estão prontos para receber os contêineres.

Figura 8 – Migração de contêineres e volumes.



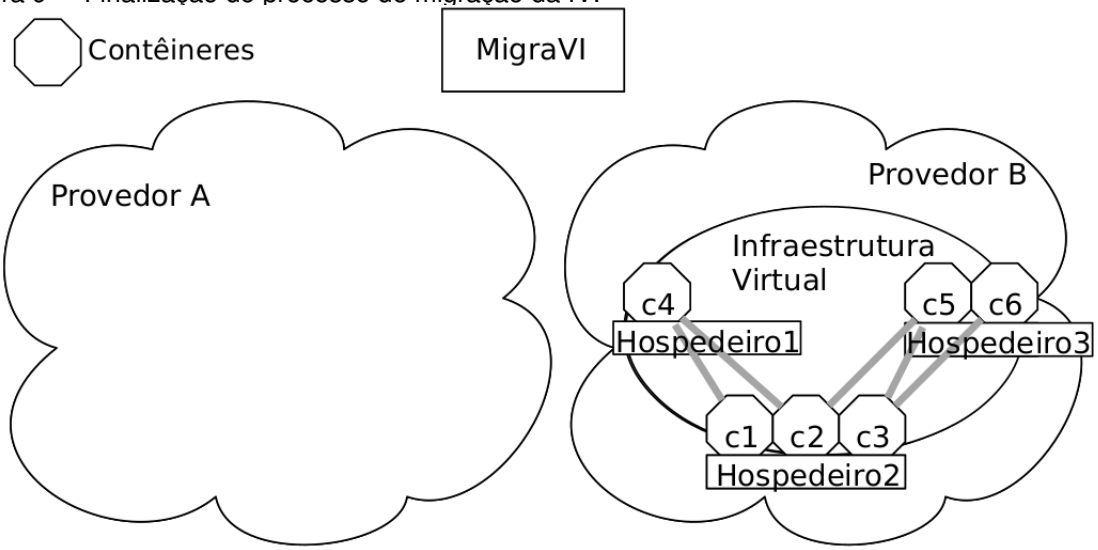
Fonte: O próprio autor.

Na sequência (Figura 8) os contêineres são migrados, cada um respectivamente para a MV na nuvem de destino que é correspondente ao hospedeiro da nuvem de origem. Neste ponto, caso os contêineres possuam volumes associados a eles, estes também são migrados para a nuvem de destino.

Por fim, conforme a Figura 9, o processo de migração é finalizado após todos os contêineres e volumes terem sido migrados e restaurados na nuvem de destino,

finalizando assim o processo de migração. Todos os processos de migração exemplificados nas Figuras 7, 8 e 9 são discutidos na Seção 4.4.

Figura 9 – Finalização do processo de migração da IV.



Fonte: O próprio autor.

Com o processo de migração do mecanismo proposto definido, faz-se necessário definir os pré-requisitos do mecanismo para realizar a migração. Dessa forma, levando em consideração os cenários e passos necessários para realizar a migração apresentados nas Figuras 7, 8 e 9, os pré-requisitos são:

- Pré-requisito 1: a IV de origem deve estar localizada em um único provedor;
- Pré-requisito 2: acesso administrativo ao sistema operacional das MVs da IV para manipulação dos contêineres;
- Pré-requisito 3: acesso às informações da RV na origem; e
- Pré-requisito 4: equivalência de parâmetros entre provedores para configuração das redes. O destino deve oferecer no mínimo as configurações já disponíveis na nuvem de origem.

É importante ressaltar que o primeiro pré-requisito para realizar a migração é que toda a IV esteja localizada em um único provedor e dessa forma, a mesma será migrada completamente para o provedor de destino, mantendo suas características originais. Ou seja, inicialmente, o mecanismo proposto por este trabalho não considera os cenários de *cloud bursting*. De acordo com Nair et al. (2010), *cloud bursting* é a capacidade de empresas possuírem sua própria infraestrutura e de tempos em tempos utilizarem, juntamente com a sua infraestrutura, os recursos de provedores externos, ou seja, nesses cenários a IV estará dividida em um ou mais provedores.

Sendo atendidos todos os pré-requisitos definidos, a migração da IV pode ocorrer, entretanto é necessário observar os requisitos funcionais e não funcionais.

4.2 REQUISITOS FUNCIONAIS E NÃO FUNCIONAIS

Nesta seção são listados os requisitos funcionais e não funcionais que definem os conjuntos de entrada, assim como o comportamento de saída do mecanismo proposto por este trabalho. É importante ressaltar que os requisitos aqui relacionados foram identificados com base no objetivo proposto para o MigraVI.

Requisitos funcionais:

- RF1 - Parâmetros de entrada: O mecanismo deverá receber como parâmetros de entrada os dados de autenticação e de autorização para as duas nuvens IaaS, bem como as chaves SSH para acessar as MVs;
- RF2 - Migração de contêineres: O mecanismo deverá realizar a migração somente de contêineres. Estes por sua vez deverão estar executando em MVs. Isso pois, conforme detalhado nas Seções 2.4.2 e 2.4.3, possibilita a migração sem a necessidade de acesso ao hipervisor;
- RF3 - Migração da rede: O mecanismo deverá realizar a migração das configurações da rede privada e virtualizada entre provedores diferentes;
- RF4 - Armazenamento dos dados de acesso: O mecanismo não deverá armazenar os dados de autorização, autenticação e nem as chaves SSH ao término da migração;
- RF5 - Endereços IPs: Para cada MV deverá ser atribuído um endereço de IP flutuante (público) para realizar a migração e deverá ser mantido o mesmo endereço de IP local;
- RF6 - Canal de comunicação entre as nuvens: O mecanismo deverá criar um canal de comunicação entre as nuvens de destino e de origem para realizar a migração;
- RF7 - Tipo de migração: Por ser uma migração administrativa, o mecanismo irá realizar uma *cold-migration*, ou seja, por um determinado período de tempo todas as aplicações param de executar para serem migradas para a nuvem de destino; e
- RF8 - Destruição da IV: Após o término da migração, o mecanismo destruirá a IV da nuvem de origem.

Dessa forma, RF1 define que o mecanismo deverá receber como entrada os dados de autenticação e também de autorização para ambas as nuvens envolvidas no processo, além das chaves SSH de acesso às MVs. Em seguida, RF2 e RF3 definem o escopo de migração realizada pelo mecanismo. Neste caso, são realizadas somente migrações de contêineres hospedados em MVs e também da configuração da rede privada virtualizada, conforme apresentado na Subseção 2.2.2.

O RF4 trata do armazenamento dos dados de acesso as nuvens e as MVs. O mecanismo desenvolvido não armazena nenhum destes dados e após a finalização da migração, os dados são descartados. Dessa forma, caso o usuário desejar realizar outra migração ou caso uma migração falhe e o usuário venha a tentar realizar a migração novamente, os dados devem ser fornecidos novamente para o mecanismo.

O RF5 define que para cada MV deve ser atribuído um endereço de IP flutuante e o endereço de IP local da origem deve ser mantido na nuvem de destino. Isso é necessário para poder realizar a comunicação direta entre as MVs da nuvem de origem com as MVs da nuvem de destino. A definição deste requisito leva em consideração que a conta do usuário tenha recursos suficiente para alocar um endereço de IP para cada MV e que caso o usuário não tenha essa disponibilidade no pacote de recursos adquiridos tanto do provedor de origem quanto do destino, a migração não será realizada. Após o término da migração os IPs alocados na nuvem de destino se mantém, entretanto, na fase de destruição/desalocação da nuvem de origem os IPs são liberados.

No RF6 é definido que o mecanismo deverá criar um canal de comunicação entre as nuvens. Este canal de comunicação se faz necessário para realizar a migração dos contêineres entre as nuvens de origem e de destino conforme explicado na Seção 4.4. Por fim, (RF7) e (RF8) definem que será realizada uma *cold-migration*, ou seja, há interrupção nos serviços que estiverem executando na IV e que após finalizada a migração, a IV localizada na nuvem de origem é finalizada ou destruída pelo mecanismo.

Além dos requisitos funcionais, foram identificados alguns requisitos não funcionais para o mecanismo proposto. Os requisitos não funcionais são:

- RNF1 - Tempo de migração: Apesar de ser realizada uma *Cold-migration*, o mecanismo deve buscar realizar uma migração com o menor tempo de interrupção dos serviços possível; e
- RNF2 - Local de configuração do mecanismo: O mecanismo pode ser configurado externamente às nuvens ou dentro da nuvem de origem.

O RFN1 aborda o tempo de migração da IV. Apesar de ser uma *Cold-migration*,

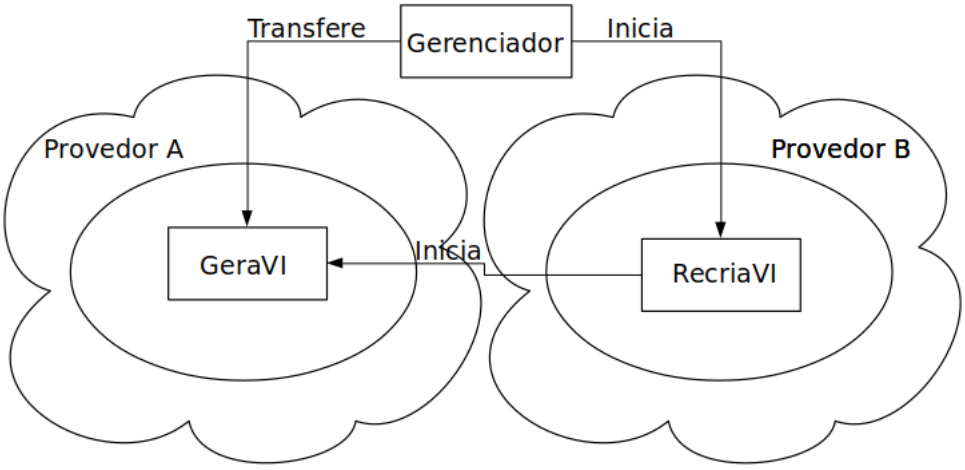
uma ferramenta que necessite diversas horas para realizar a migração de uma IV não será viável para o usuário. Dessa forma, o mecanismo proposto tem como objetivo realizar a migração com um mínimo possível de tempo de interrupção dos serviços que estão executando na IV. Por fim, o RNF2 define que o mecanismo pode ser configurado fora das nuvens envolvidas. Entretanto, caso o usuário deseje, o mesmo poderá ser configurado dentro da nuvem de origem. Neste caso, o mecanismo será destruído ao final da migração da IV. O único local no qual o mecanismo não poderá ser configurado é na nuvem de destino, pois neste caso, seria necessário já existir alguma MV para que o mesmo possa executar.

4.3 MIGRAVI

A arquitetura proposta do MigraVI é composta por três componentes ou módulos. O primeiro módulo é denominado **Gerenciador**, além desse componente existem dois componentes que executam na própria nuvem de origem e na nuvem de destino, são eles **GeraVI** e **RecriaVI**.

A Figura 10 ilustra a arquitetura proposta para o MigraVI. Como é possível observar na Figura 10 o componente **Gerenciador** executa na nuvem de origem, entretanto existe a possibilidade de configurá-lo em um hospedeiro externo às duas nuvens. Para tal, ele exige somente as credenciais de acesso do cliente na nuvem de origem e de destino.

Figura 10 – Arquitetura do MigraVI



Fonte: O próprio autor.

O **Gerenciador** é responsável pela cópia das aplicações, na nuvem de origem, e pelo transporte destas aplicações para a nuvem de destino. De maneira geral o **GeraVI** que executa na nuvem de origem é responsável por realizar o levantamento das informações da RV da nuvem de origem e realizar o *checkpoint* dos contêineres

na nuvem de origem. Já o módulo **RecriaVI**, que executa na nuvem de destino recria a RV e recupera os *checkpoints* dos contêineres. Os módulos **Gerenciador**, **GeraVI** e **RecriaVI** e as aplicações que executam na origem e no destino são explicados de forma detalhada na Seção 5.2.

4.4 SEQUÊNCIA DE EVENTOS PARA MIGRAÇÃO DE UMA IV

Esta seção tem como objetivo apresentar e discutir o fluxo de dados e a sequência de eventos para a migração de IVs. Conforme descrito na Seção 4.1, o cenário de migração é composto por:

1. Nuvem de origem: a nuvem de origem de um provedor A, essa nuvem deve possuir uma IV previamente configurada pelo usuário;
2. Nuvem de destino: nuvem de destino de um provedor B, essa é a nuvem escolhida pelo usuário para a qual a IV será migrada; e
3. Mecanismo de migração: responsável por realizar a migração da IV da nuvem de origem para a nuvem de destino.

A migração da IV desempenhada pelos módulos da arquitetura do MIGRAVI seguem uma sequência única e genérica de eventos. O fluxo de dados para migração da IV é representado na Figura 11. Os passos estão numerados de 1 a 7. Assim que acionado pelo cliente e de posse dos dados de acesso nas nuvens, o **Gerenciador** inicia o processo de migração conectando na nuvem de origem (passo 1) e lançando o **GeraVI** (passo 2). Para os dados da rede, é realizada uma leitura da topologia privada e de seus componentes, enquanto para as MVs são recuperados informações como endereços IP, tipo de MV e até mesmo imagens utilizadas para a sua criação.

Assim que **GeraVI** finaliza, as informações obtidas são enviadas para o **Gerenciador** (passo 3). No passo 4, cabe ao **Gerenciador** a verificação das informações recebidas do **GeraVI**. Neste ponto, as informações de conexão com a nuvem de destino também podem ser validadas e caso algo esteja incorreto a migração pode ser abortada. Na sequência, com as informações coletadas no primeiro passo, o mecanismo se conecta com a nuvem de destino e lança o **RecriaVI** (passo 5). O **RecriaVI** instancia as MVs e cria a rede externa, associando novos IPs flutuantes na faixa do provedor de destino (passo 6). No passo 7, o **RecriaVI** retorna o estado da operação para o **Gerenciador**, sucesso ou fracasso.

Com a IV configurada na nuvem de destino, o **Gerenciador** constrói uma tabela de mapeamento de contêineres/MVs da nuvem de origem para contêineres/MVs na nuvem de destino (passo 8). O passo 9 descreve o estabelecimento de um canal de

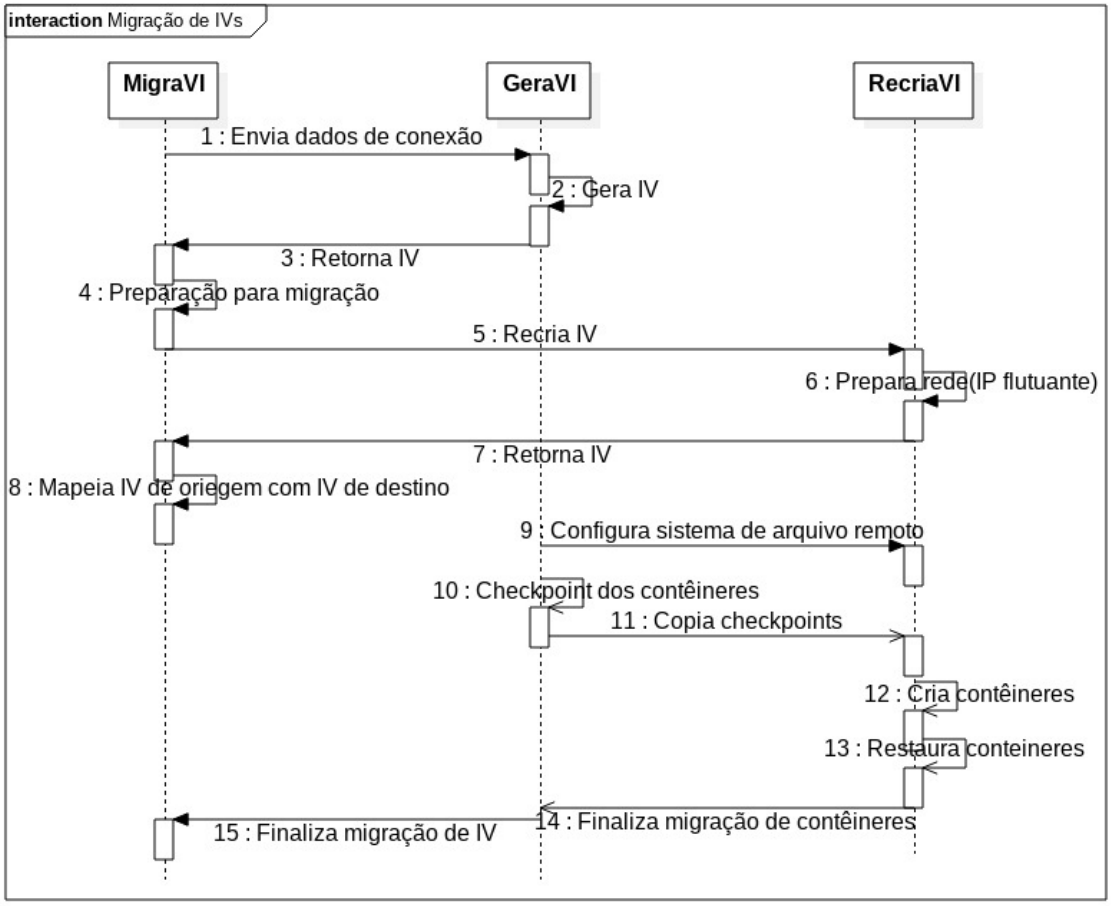


Figura 11 – Diagrama de sequência para realizar a migração de IVs entre provedores distintos.

comunicação entre a nuvem de origem, gerenciado pelo **GeraVI**, e a de destino, gerenciado pelo **RecriaVI**. O passo 10 habilita o sistema de transferência dos dados remoto, permitindo que o **RecriaVI** recupere os estados dos contêineres do **GeraVI**. A partir do momento em que o canal de comunicação entre as nuvens é criado, o processo de *checkpoint* dos contêineres em execução nas MVs da nuvem de origem é iniciado (passo 11). Neste ponto, a execução do *checkpoint* pode ser realizada de forma paralela em cada MV. Com a finalização do *checkpoint* dos contêineres, os mesmos são copiados através do canal de comunicação para a MV da nuvem de destino (passo 12).

Com o final da cópia dos *checkpoints* para a nuvem de destino, os novos contêineres são criados e os *checkpoints* restaurados (passos 13 e 14). Neste ponto, o serviço é totalmente reabilitado na nuvem de destino, reestabelecido com o mesmo estado da interrupção na nuvem de origem. O período entre o último *checkpoint* na nuvem origem e a restauração na nuvem destino compreende o tempo de indisponibilidade da aplicação. Por fim, ao concluir a restauração dos estados dos contêineres a partir dos *checkpoints* gerados, a nuvem de origem pode ser destruída, e com isso

o processo de migração está finalizado (passo 15). Os algoritmos que representam o funcionamento dos três módulos apresentados na Figura 11 são descritos na Seção 4.5.

4.5 FUNCIONAMENTO DO MIGRAVI: ALGORITMOS

Os passos necessários para realizar a migração de uma IV utilizando o MigraVI são implementados e podem ser descritos em três algoritmos distintos. O pseudocódigo do **Gerenciador** é apresentado no Algoritmo 1.

Algoritmo 1: Gerenciador pseudocódigo.

```
   Data: P_src; P_dst; VI_id
 1 template = get_vi_specification(P_src, VI_id);
 2 target_vms = create_hosts(VI_id, template, P_dst);
 3 network = create_network(VI_id, template, P_dst);
 4 for ∀ dst_hosts ∈ target_hosts do
 5 |   dst_host_ip = get_host_ip(dst_hosts);
 6 |   src_host = get_src_host(template, dst_hosts);
 7 |   src_host_ip = get_host_ip(src_host);
 8 |   launch_dst_algorithm(P_dst, dst_host_ip);
 9 |   launch_src_algorithm(P_src, src_host_ip, dst_host_ip);
10 for ∀ dst_hosts ∈ target_hosts do
11 |   join_algorithm(P_dst, dst_host);
12 if migration_accomplished(P_dst) then
13 |   release_vi(P_src, VI_id)
14 else
15 |   release_vi(P_dst, VI_id)
```

Inicialmente, as credenciais de autenticação para as nuvens de origem e de destino são informadas (P_{src} e P_{dst}, respectivamente), além do identificador da IV, representado por VI_{id}. Na posse destes dados, a leitura da especificação da nuvem de origem é gerada. Na sequência, como pode ser observado nas linhas 2 e 3, o **Gerenciador** realiza um acesso na nuvem de destino e recria os hospedeiros e a RV com base no arquivo gerado. Depois, na linha 4, para cada hospedeiro criado na nuvem de destino é realizada uma correspondência com um hospedeiro na nuvem de origem. Essa correspondência é realizada através do endereço de IP local dos hospedeiros que se mantém o mesmo nas duas nuvens. Após realizado esse mapeamento, os algoritmos são iniciados nos hospedeiros de origem e de destino (linhas 5 a 9). É relevante ressaltar que os algoritmos são tarefas que executam em paralelo (linhas 8 e 9) diminuindo assim o tempo total de migração. Nas linhas 10 e 11, quando os algoritmos finalizam a sua execução, seja ela uma execução com sucesso ou uma falha, o processo de migração é finalizado. Por fim, nas linhas 12 a 15, o **Gerenciador** solicita a destruição da IV de origem, ao **GeraVI**, em caso de sucesso, ou de destino, ao **RecriaVI**, em caso de falha.

Os Algoritmos 2 e 3 representam os pseudocódigos dos módulos **GeraVI** e **RecriaVI**, ou seja, algoritmos responsáveis pelas ações na IV de origem e de destino, respectivamente. De maneira geral, os algoritmos que executam nos hospedeiros tanto de origem quanto de destino, recebem como entrada os dados de autenticação para o hospedeiro assim como o seu endereço de IP flutuante. Neste cenário, um sistema de arquivos remoto é utilizado para a sincronização dos dados e a realização da transferência dos *checkpoints* dos volumes e dos contêineres.

Algoritmo 2: Pseudocódigo GeraVI.

```
   Data: P_src, dst_host_ip
1  nfs_path = mount_remote_file_system(dst_host_ip);
2  for ∀ container ∈ get_containers() do in parallel
3      volumes = get_volumes_checkpoints (container, file_server_path);
4      checkpoint = container_checkpoint(container, file_server_path);
```

Algoritmo 3: Pseudocódigo RecriaVI.

```
   Data: P_dst, src_host_ip
1  nfs_path = export_remote_file_system_server_to(src_host_ip);
2  wait_containers_migration(src_host_ip);
3  for ∀ container ∈ nfs_path do in parallel
4      volumes = get_volumes_checkpoints (container, nfs_path);
5      create_container(container, volumes);
```

Como é possível observar no Algoritmo 2, na linha 1, é montado o sistema de arquivo remoto na origem. Em seguida, para cada contêiner executando na MV é realizado o *checkpoint* dos dados, escrevendo-os em um sistema de arquivo compartilhado (linhas 3 e 4). Dessa forma, os dados gerados pelo **GeraVI** podem ser recuperados, quando necessário, pelo **RecriaVI**, no hospedeiro de destino. Por sua vez, o Algoritmo 3 representa o pseudocódigo do **RecriaVI**, executado na nuvem de destino. Na primeira linha, o sistema de arquivo remoto é exportado, posteriormente, o processo de restauração dos contêineres e dos volumes é realizado (linhas 3 a 5). É importante ressaltar que os processos de *checkpoint*, linhas 2 a 4 do Algoritmo 2, e restauração dos contêineres e volumes, linhas 3 a 5 do Algoritmo 3, podem ser executados paralelamente.

Com o objetivo de analisar o mecanismo e os algoritmos descritos, foram elaborados planos de testes. A Seção 4.6 apresenta os planos de testes elaborados.

4.6 PLANO DE TESTES

Com o objetivo de analisar a eficácia do mecanismo proposto neste trabalho, quatro cenários de testes são propostos. A Tabela 2 resume os cenários, indicando as nuvens de origem e destino, bem como a composição da IV.

Tabela 2 – Cenários de testes.

Cenário	Nuvem de origem	Nuvem de destino	Composição da IV	Objetivo
Cenário 1	Provedor A (projeto 1)	Provedor A (projeto 2)	Um contêiner sem rede	Objetivo do teste é obter um tempo mínimo médio na migração de contêineres entres projetos na mesma nuvem.
Cenário 2	Provedor A (projeto 1)	Provedor A (projeto 2)	Dez contêineres, um *switch* e uma rede privada virtual	Teste realizado para analisar o comportamento da migração dos contêineres e também para verificar os tempos de migração.
Cenário 3	Provedor A (projeto 1)	Nuvem externa	Um contêiner sem rede	Objetivo do teste é obter um tempo mínimo médio na migração de contêineres entres provedores distintos.
Cenário 4	Provedor A (projeto 1)	Nuvem externa	Dez contêineres, um *switch* e uma rede privada virtual	Teste realizado com o objetivo de verificar como funciona o comportamento da migração dos contêineres e também para verificar os tempos de migração.

Fonte: O próprio autor.

O primeiro cenário de teste (Cenário 1) proposto compreende a migração entre dois projetos diferentes, de maneira que estes projetos estejam no mesmo provedor (Provedor A). Este cenário de teste realiza somente a migração de um único contêiner sem realizar a migração da RV. A realização deste teste tem como principal objetivo obter uma base tempo necessário para realizar a migração de contêineres. Essa base servirá para realizar a comparação com os demais testes realizados.

O Cenário 2 realiza a migração de dez contêineres assim como a migração da RV. Neste cenário é realizada a migração completa da IV para outro projeto no mesmo Provedor A, todos os passos descritos na Seção 4.4 são reproduzidos sem exceções. A migração entre projetos de um mesmo provedor representa a realocação de uma IV em outra zona ou região.

Os Cenários 3 e 4 representam as migrações realizadas de uma IV para uma nuvem em outro provedor. Assim como no Cenário 1, o Cenário 3 serve como *ba-*

seline, migrando somente um contêiner. Já no Cenário 4, é feita a migração da IV completa de um provedor para outro. Vale ressaltar, novamente, que nos Cenários 2 e 4 os contêineres são migrados paralelamente com o objetivo de otimizar o tempo de migração total da IV.

Os Cenários 1 e 3 servem apenas de base, ou seja, seus resultados são utilizados para traçar um perfil da migração realizada e assim ter uma base de tempo mínimo em que a menor unidade de migração está sendo migrada. Já os cenários 2 e 4 representam a situação real, na qual o usuário utilizará o mecanismo para realizar a migração de sua IV.

Ao contrário dos Cenários 1 e 3, que são utilizados somente como base para os resultados, os Cenários 2 e 4 se aproximam da realidade do usuário. De maneira que, este deseja realizar a migração completa de sua IV já configurada em um provedor para outro provedor de nuvem. Nestes cenários é possível notar a migração da rede que é realizada, conforme apresentado na Seção 3.1, é um tema pouco abordado.

Nos cenários propostos nota-se o atendimento dos pré-requisitos. O primeiro pré-requisito é atendido pois, em cada um dos cenários os quais é realizado a migração da IV, essa IV encontra-se inteiramente em um único provedor. Ao compor a IV com MVs e contêineres que executam sobre estas MVs o segundo pré-requisito é atendido, pois dessa forma os contêineres podem ser migrados acessando o SO da MV. O terceiro pré-requisito deve ser atendido para poder realizar a migração da IV, ou seja, caso não seja possível obter acesso as informações da IV de origem a migração não pode ocorrer. Já o atendimento do quarto pré-requisito pode variar de provedor para provedor e dessa forma, ao ser migrado para um provedor diferente, fica sob responsabilidade do usuário identificar se esse provedor oferece as opções necessárias para que os parâmetros mínimos sejam atendidos.

4.7 CONSIDERAÇÕES PARCIAIS

Este capítulo teve como principal objetivo apresentar a proposta para a realização de migração de IVs entre provedores de nuvens distintos. Para realizar essa migração um mecanismo deve ser utilizado, um dos principais requisitos desse mecanismo é a utilização dos dados de autenticação e autorização das nuvens. Esses dados devem ser fornecidos pelo usuário e o mecanismo proposto não tem como responsabilidade realizar o gerenciamento destes dados. Outro ponto importante é a atribuição de endereços IP flutuantes para cada uma das MVs envolvidas na migração, pois somente com esses endereços de IP é que o mecanismo poderá se conectar nas MVs e assim realizar a migração dos contêineres

Para a realização da migração de IV, alguns passos devem ser seguidos. Esse fluxo proposto (Seção 4.4) é genérico, ou seja, pode ser aplicado para realizar uma migração de uma IV em qualquer provedor, entretanto esse trabalho também apresenta uma implementação de um mecanismo que segue estes passos no Capítulo 5.

5 IMPLEMENTAÇÃO

Esse capítulo tem como objetivo apresentar o MigraVI-OpenStack, um protótipo baseado em OpenStack para implementação da arquitetura do mecanismo MigraVI, discutida no Capítulo 4. O MigraVI-OpenStack[1] é responsável por realizar a migração de uma IV entre provedores distintos e atende o fluxo de migração proposto na Seção 4.4 e os requisitos funcionais e não funcionais propostos na Seção 4.2. As características e funcionamento do MigraVI-OpenStack são apresentados na Seção 5.2.

Visando atender os pré-requisitos levantados na Seção 4.1 foram selecionados o *framework* OpenStack e a ferramenta Flame[2]. Neste sentido, a utilização de uma nuvem OpenStack atende ao primeiro pré-requisito, o qual define que toda a IV deve estar em uma única nuvem. A escolha da nuvem OpenStack deve-se ao fato de que é a nuvem disponível nos ambientes de experimentação (discutidos na Seção 6.1).

. Por ser um ambiente de nuvem IaaS e visando atender aos cenários levantados na Seção 4.1, a ferramenta Docker[3] é selecionada para a entrega de contêineres e dessa forma somente é necessário obter acesso administrativo as MVs (sendo assim atendido o segundo pré-requisito). Ao escolher a nuvem do Laboratório de Processamento Paralelo e Distribuído (LabP2D), e com a utilização do Flame, o terceiro pré-requisito é atendido, pois com essa ferramenta é possível obter as informações da nuvem de origem.

As ferramentas Convoy[4] e *Checkpoint/Restore In Userspace* (CRIU) são utilizadas para realizar o *checkpoint* dos contêineres e seus volumes, atendendo o RF2. Para atender os requisitos RF3 e RF5, quanto a migração de rede e atribuição de IPs, o projeto Heat do OpenStack é utilizado em conjunto com a ferramenta Flame. O requisito RF6 referente a criação do canal de comunicação, sendo atendido através da utilização de um *Network File System* (NFS). Neste sentido, a Seção 5.1 apresenta as ferramentas e tecnologias utilizadas no desenvolvimento do MigraVI-OpenStack.

5.1 FERRAMENTAS E TECNOLOGIAS

Oo protótipo MigraVI-OpenStack é baseado em um conjunto de softwares, linguagem e configurações com o objetivo de atender todos os pré-requisitos e requisitos descritos nas Seções 4.1 e 4.2, respectivamente. Dessa maneira, esta seção

[1] Código fonte do protótipo disponível em: <https://bitbucket.org/micovick/migracao-de-ivs.git>.
[2] Flame - Automatic Heat template generation: <https://github.com/openstack/flame.>
[3] Docker: <https://www.docker.com/>.
[4] Convoy: <https://github.com/rancher/convoy>.

apresenta detalhes técnicos necessários para analisar o protótipo desenvolvido. Os softwares e ferramentas utilizadas são:

- Docker versão 17.05.0-*Community Edition - CE*;
- Convoy versão 0.5.0;
- Protótipo do MIGRAVI foi desenvolvido utilizando a linguagem de programação Python na versão 2.7.12;
- Composição da IV são utilizadas MVs que executam em GNU/Linux Ubuntu 16.04;
- MVs com 1 vCPU e 1 GB de RAM; e
- OpenStack versão Mitaka.

5.1.1 OpenStack

O OpenStack é um conjunto de projetos modelado para gerenciar infraestruturas de nuvens computacionais, sejam elas públicas ou privadas (OPENSTACK, 2017). Dessa forma, pode-se dizer que o OpenStack é um *Cloud Operating System* (COS), responsável por gerenciar e controlar uma grande quantidade de recursos, sendo eles computacionais, de armazenamento e de rede. Além de ser uma ferramenta de código aberto, sua arquitetura foi modelada para permitir que administradores da nuvem controlem os recursos através de ferramentas de monitoramento (OPENSTACK, 2017). Os recursos podem ser provisionados e acessados pelos usuários através de um painel de controle o qual pode ser acessado pela Internet através de uma interface web.

Além dessa interface web, o OpenStack também fornece acesso as suas funcionalidade através de APIs (OPENSTACK, 2017). As APIs do OpenStack recebem requisições dos consumidores e as transformam em ações na nuvem (CORRADI; FANELLI; FOSCHINI, 2014). De maneira geral, essas APIs fazem uso de *webservices*, de maneira que as requisições *Hypertext Transfer Protocol* (HTTP) realizadas a estes *web services* são interpretadas e convertidas para comandos na nuvem.

Uma arquitetura típica do OpenStack é formada pelos seguintes principais componentes: *Compute*, *Storage, Backup & recovery*, *Networking & Contente Delivery*, *Data & Analytics*, *Security, Identity & Compliance*, *Management Tools*, *Deployment Tools*, *Application Services* e *Monitoring & Metering*. Cada um destes componentes são formados por diversos projetos, dos quais os principais são:

- *Nova* e *Glance* do componente *Compute*;

- *Neutron* do componente *Networking*;
- *Swift* e *Cinder* do componente *Storage, Backup & Recovery*;
- *Keystone* pertencente ao componente *Security, Identity & Compliance*;
- *Ceilometer* do componente *Monitoring & Metering*; e
- *Horizon* do componente *Management Tools*.

O projeto *Nova* é responsável por fornecer MVs sob demanda e interagir com os hipervisores. Esse projeto possui suporte para diversos hipervisores, *e.g.,* KVM, Xen VMware e Hyper-V. Dessa forma, através da utilização de APIs, o *Nova* fornece meios de orquestrar as instâncias que estão executando, redes e também controle de acesso (OPENSTACK, 2017).

O projeto *Glance*, que assim como o *Nova* faz parte do componente *Compute*, fornece meios de pesquisa e recuperação de imagens de MVs. Ao utilizar a API do Nova, o administrador tem a possibilidade de criar e gerenciar bibliotecas de imagens de MVs, assim como obter metadados destas imagens.

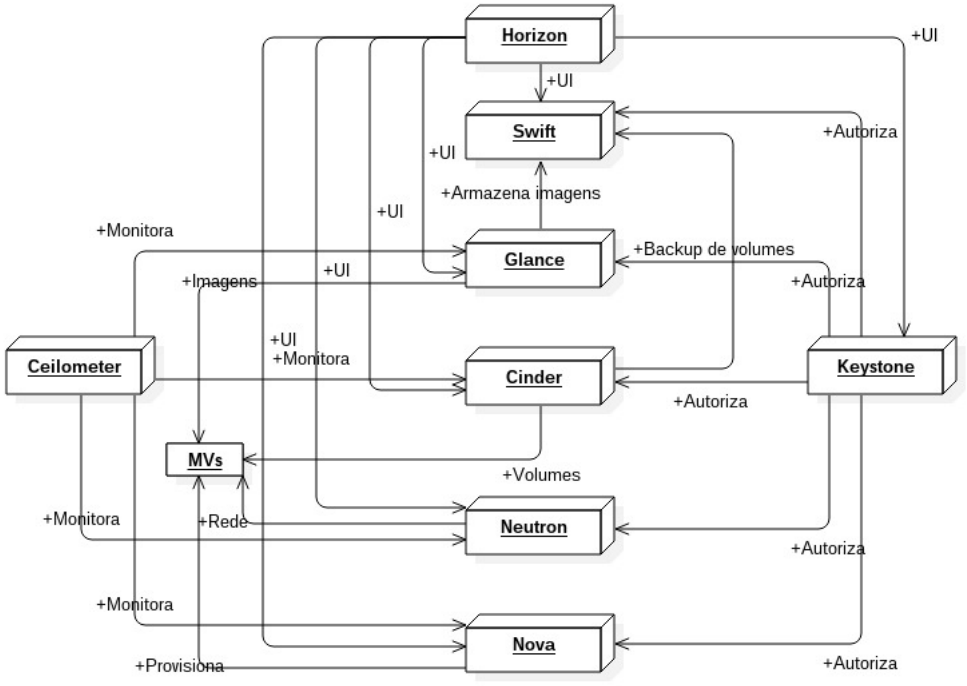
Tratando da rede do OpenStack, o projeto *Neutron* fornece um serviço de conectividade entre os dispositivos gerenciados por outros projetos do OpenStack como, por exemplo, o *Nova*. Esse projeto também permite o gerenciamento e a configuração de endereços de IP, VLAN assim como o *Dynamic Host Configuration Protocol* (DHCP). A conectividade e o relacionamento entre estes principais projetos é apresentada na Figura 12. Estes componentes juntamente com os projetos apresentados formam a arquitetura básica do OpenStack.

Já em relação ao componente de armazenamento, o *Swift* disponibiliza o armazenamento de dados distribuídos e dessa forma possuí uma tolerância a falha. O *Swift* pode ser utilizado juntamente com o *Cinder* e assim realizar o backup dos volumes das MVs. O projeto *Cinder* fornece o armazenamento persistente dos dados da MVs em volumes.

O projeto *Keystone* prove serviços de acesso e autenticação ao OpenStack. Esse projeto também provê ferramentas para o descobrimento de outros serviços, assim como essas políticas de acesso e de autorização podem ser aplicadas para outros serviços do OpenStack.

A funcionalidade de acesso aos demais serviços proporcionados pelo OpenStack é fornecida pelo *Horizon*. Esse projeto disponibiliza um painel de controle o qual pode ser acessado através da Internet por um navegador. Por fim, o projeto *Ceilometer* fornece meios de monitorar, coletar e normalizar dados de outros projetos do OpenS-

Figura 12 – Arquitetura do OpenStack



Fonte: O próprio autor.

tack. Com a utilização deste projeto é possível identificar e assim resolver diversos problemas de telemetria.

5.1.2 Flame

No OpenStack, a orquestração de IV pode ser realizada através do projeto HEAT, o qual descreve a IV e seus recursos através de modelos denominados *Heat Orchestration Template* (HOT) (OPENSTACK, 2017). Entretanto, após uma IV ser criada, o OpenStack não fornece uma ferramenta para recuperar as informações da IV de forma automatizada. Neste cenário, o Flame é uma ferramenta a qual se propõe a gerar o modelo HOT de uma IV já existente (HERVE; REZMERITA, 2017).

O Flame faz uso das APIs Nova do OpenStack para se conectar aos serviços e recuperar as informações dos recursos que estão sendo utilizados pelo projeto. O primeiro passo necessário para utilizar o Flame é fornecer os dados de acesso ao projeto, ou seja, o usuário, o nome do projeto, senha de acesso ao projeto e a *Uniform Resource Locator* (URL) de autenticação. Com esses dados o Flame conecta-se nos serviços do OpenStack e lista os recursos utilizados, produzindo uma saída no modelo HOT.

Dentre os tipos de recursos que o Flame possui, é importante ressaltar a geração da rede que compõe a IV e a alocação dos IPs flutuantes. Para exemplificar a

extração de tais informações, o bloco de código 5.1 descreve como é feita a alocação de IPs flutuantes.

Código 5.1 – Alocação de IPs flutuantes.

```
1  floatingip_0:
2          properties:
3            floating_network_id:
4              get_param: external_network_for_floating_ip_0
5          type: OS::Neutron::FloatingIP
```

Como pode ser observado, a alocação dos recursos inicia-se pela definição de um identificador único, neste caso $floatingip\_0$ (linha 1) e finaliza com a definição de um tipo para o recurso que está sendo definido, neste caso, $OS :: Neutron :: FloatingIP$ (linha 5). Como os IPs são alocados de uma rede, um identificador da rede deve ser fornecido, entretanto para não fixar o identificador, usa-se um parâmetro para identificar uma rede externa ($external\_network\_for\_floating\_ip\_0$) (linha 4), que será provida na criação da pilha pelo HEAT. Já o bloco de código apresentado em 5.2, representa a alocação de uma rede.

Código 5.2 – Alocação de rede.

```
1   network_0:
2     properties:
3       admin_state_up: true
4       name: rede_1
5       shared: false
6     type: OS::Neutron::Net
7   port_175:
8     properties:
9       admin_state_up: true
10      device_owner: compute:nova
11      fixed_ips:
12      - ip_address: 192.168.1.12
13        subnet_id:
14          get_resource: subnet_0
15      mac_address: fa:16:3e:a9:92:6a
16      network_id:
17        get_resource: network_0
18      security_groups:
19      - get_resource: security_group_0
20    type: OS::Neutron::Port
21  port_268:
22    properties:
23      admin_state_up: true
24      device_owner: compute:nova
25      fixed_ips:
```

```
26        - ip_address: 192.168.1.11
27          subnet_id:
28            get_resource: subnet_0
29        mac_address: fa:16:3e:28:d8:6e
30        network_id:
31          get_resource: network_0
32        security_groups:
33        - get_resource: security_group_0
34      type: OS::Neutron::Port
```

Ao observar o código em 5.2 é possível notar a definição de portas (linhas 7 e 21). Essas portas são utilizadas para definir os endereços de IPs locais das MVs (linhas 12 e 26). Após definida a rede, os hospedeiros podem ser definidos. O bloco de código 5.3 apresenta a alocação de um hospedeiro considerando as portas disponíveis na definição da rede. Como é possível observar, pode-se configurar o tamanho (*flavor*), a imagem utilizada, a chave de acesso e o nome do hospedeiro (linhas 6,8 e 10 respectivamente), além disso a definição do endereço IP local é feita com base na porta já previamente definida na rede (linha 14), neste caso foi associado a porta 175 para este hospedeiro.

Código 5.3 – Alocação de hospedeiros.

```
1  server_0:
2      properties:
3        config_drive: 'True'
4        diskConfig: AUTO
5        flavor:
6          get_param: server_0_flavor
7        image:
8          get_param: server_0_image
9        key_name:
10         get_param: server_0_key
11       name: vm_destino
12       networks:
13       - port:
14           get_resource: port_175
15     type: OS::Nova::Server
```

Além da definição do IP flutuante, do servidor e da rede é necessário atribuir o IP flutuante para o hospedeiro. A atribuição deste IP flutuante é apresentada no bloco de código 5.4.

Código 5.4 – Atribuição de IP flutuante a um hospedeiro.

```
1  floatingip_association_1:
2      properties:
3        floatingip_id:
```

```
4           get_resource: floatingip_0
5         port_id:
6           get_resource: port_175
7       type: OS::Neutron::FloatingIPAssociation
```

Como é possível observar, na linha 4 é utilizado o primeiro IP flutuante definido pelo identificador $floatingip_0$. Na sequência, na linha 6, é feita a associação deste IP flutuante com a porta 175 definida na rede. Dessa forma, como o hospedeiro $server_0$ também está associado a porta 175, ele também receberá o IP flutuante que foi associado a esta porta. Dessa maneira, com a utilização do projeto Heat e do Flame para a geração do modelo HOT, é possível recriar a IV na nuvem de destino, mantendo assim os mesmos endereços de IP locais e dessa forma o requisito funcional cinco (RF5) é atendido. Devido a estas características o Flame é utilizado pelo componente **GeraVI**, já o Heat é utilizado tando pelo **GeraVI** quanto pelo **RecriaVI**.

Finalizando, a parte de levantamento das informações da rede, os contêineres devem ser migrados. Neste trabalho, para a implantação dos contêineres é utilizado a ferramenta Docker em conjunto com o Convoy para gerenciamento dos volumes.

5.1.3 Docker e Convoy

O Docker é uma ferramenta para a entrega e gerenciamento de contêineres em ambientes locais e em nuvens híbridas (DOCKER, 2017). Com o Docker, caso o usuário deseje persistir os dados, por exemplo, um banco de dados, deve ser utilizado um volume. Dessa forma os dados salvos nos volumes são persistidos, mesmo que o contêiner seja destruído os dados ainda estarão disponíveis nos volumes e poderão ser acessados por outros contêineres. Entretanto, os volumes do Docker são atrelados ao hospedeiro no qual foram inicialmente criados e dessa forma, caso o contêiner seja migrado para outro hospedeiro os volumes não são migrados e assim, não estarão mais disponíveis para o contêiner.

Para lidar com esse problema existem *drivers* para gerenciar os volumes do Docker e assim obter a portabilidade dos volumes também, dessa forma o MigraVI-OpenStack faz uso do Convoy. O Convoy é um *driver* de código aberto utilizado para criar e gerenciar volumes persistentes. Com a utilização do Convoy é possível realizar *snapshots*, cópias de segurança e também a restauração destas cópias em outros hospedeiros. Para realizar a sincronização dos *snapshots*, gerados pelo Convoy, com o hospedeiro de destino é utilizado um NFS. Dessa forma os *snapshots* são sincronizados, assim como os *checkpoints* dos contêineres também são sincronizados através deste mesmo canal de comunicação. Foi escolhido a utilização de uma NFS pois o Convoy já possui suporte nativo para essa tecnologia.

Assim, o Docker e o Convoy são utilizados pelos componentes **GeraVI** e **RecriaVI**. Isso ocorre pois, os contêineres, ao iniciar a migração, estão presentes na nuvem de origem e na nuvem de destino ao finalizar a migração.

5.2 MIGRAVI-OPENSTACK: UM MECANISMO PARA MIGRAÇÃO DE IVs

O mecanismo é composto por dois módulos distintos que tem como objetivo atender o fluxo proposto nas Seções 4.3 e 4.4, o primeiro módulo é o que realiza a leitura ou o levantamento das informações da IV, ou seja faz a especificação da IV. Ainda, o primeiro módulo também é o responsável por utilizar essas informações obtidas e replicá-las na nuvem de destino. Por fim, o segundo módulo, também faz uso do primeiro para obter as informações da nuvem de destino, por exemplo, endereços de IP designados para as MVs e assim realizar a migração dos contêineres para a nuvem de destino.

Conforme descrito na Seção 4.2, o MigraVI-OpenStack deve receber como entrada os dados de autenticação para as nuvens envolvidas no processo de migração. Após isso, o mecanismo realiza um levantamento de informações da nuvem de origem. Esse levantamento de dados é realizado pelo MigraVI-OpenStack com o auxílio da ferramenta Flame.

5.2.1 Recuperação das informações da IV

O módulo do MigraVI-OpenStack é responsável por recuperar todas as informações da IV, seja da IV configurada na nuvem de destino ou na nuvem de origem. As informações da IV são dos servidores, endereços de IPs flutuantes, grupos de segurança, usuários, *etc.*

No OpenStack, a especificação e o gerenciamento de IVs pode ser realizada pelo projeto OpenStack Heat. O projeto Heat é fundamental na orquestração de nuvens com OpenStack, pois cria um serviço acessível para humanos e máquinas possibilitando o gerenciamento de todo o ciclo de vida de IVs em nuvens OpenStack (HEAT, 2017). O projeto Heat possibilita a descrição de um IV em um arquivo de texto. O arquivo é denominado HOT e através do qual é possível especificar os recursos da IV seus relacionamentos (HEAT, 2017).

Dessa forma, o MigraVI-OpenStack depende do Flame, uma ferramenta que tem como objetivo a geração automática do HOT. O Flame interage com o OpenStack através de APIs para coletar as informações necessárias da IV e assim montar HOT. Ao utilizar o Flame, foi constatado que o mesmo não possuía suporte para lidar com os mecanismos de autenticação e autorização atualizados do OpenStack. Tais limita-

ções não foram documentadas no repositório oficial da ferramenta. Sendo assim, para compor o MigraVI-OpenStack, foi adicionado ao Flame o suporte para *Keystone v3*.

Assim, de acordo com o fluxo proposto na Seção 5.2, o MigraVI-OpenStack utiliza o Flame para gerar a especificação da IV e do Heat para requisitar a criação da IV na nuvem no provedor de destino, incluindo as configurações de endereços IP flutuantes. Como é utilizado o Heat para a migração, as MVs são recriadas com as mesmas configurações da nuvem de destino, fazendo com que o usuário não tenha a possibilidade de alterar nenhuma das configurações durante o processo de migração. Além disso, o usuário deverá ter no provedor de destino os mesmos recursos disponíveis na origem, para poder recriar as MVs. Com a rede e as MVs recriadas na nuvem de destino é necessário realizar a migração dos contêineres.

5.2.2 Migração de contêineres no MigraVI-OpenStack

Esse módulo do sistema tem como responsabilidade realizar a migração dos contêineres que estão executando na nuvem de origem para a nuvem de destino. É importante ressaltar que este módulo do sistema somente executará após o módulo definido na Subseção 5.2.1 ter finalizado a sua execução completa e com sucesso. Caso contrário, não é possível migrar os contêineres, pois se por alguma razão, a rede e as MV não foram recriadas corretamente na nuvem do provedor de destino os contêineres não poderão ser migrados. Esse módulo do sistema faz uso de outras ferramentas para auxiliar a alcançar seu objetivo, ou seja, migrar os contêineres. As ferramentas utilizadas são: Docker, Convoy e CRIU, detalhadas na Subseção 5.1.3.

Uma característica importante do MigraVI-OpenStack é que o mesmo funciona somente com contêineres Docker, descrito na Subseção 5.1.3. Vale lembrar que apesar do Docker fornecer meios para persistir dados em volumes, estes não são migrados juntamente com os contêineres e dessa forma os dados são perdidos. Dessa forma fez-se necessário a utilização de *plugins* para a realização da migração dos volumes juntamente com os contêineres, neste caso foi utilizado o Convoy, descrito na Seção 5.1.3. Por fim, a última ferramenta utilizada pelo MigraVI-OpenStack é o CRIU[5]. O CRIU é utilizado para realizar o *checkpoint*/*restore* dos contêineres em espaço do usuário. Com a utilização desta ferramenta, o estado dos contêineres é capturado e salvo em uma imagem, a qual pode ser restaurada em outro hospedeiro e assim o contêiner é restaurando com base nessa imagem.

Conforme apresentado na Seção 4.4, um canal de comunicação é necessário para realizar a migração. O canal de comunicação utilizado pelo MigraVI-OpenStack é o NFS. Foi escolhido NFS devido ao Convoy possuir integração com essa tecnologia. Os volumes criados são compartilhados com a nuvem de destino através do NFS,

[5] CRIU: <https://criu.org/Docker>.

assim como as imagens dos contêineres geradas através do CRIU. Logo, para que o sistema funcione, o usuário deverá configurar o servidor do NFS no momento em que criar os seus contêineres na nuvem de origem. Ao ser utilizado o MigraVI-OpenStack para a migração da IV, o mesmo será responsável por montar o NFS na nuvem de destino. Ainda, é importante ressaltar que, deverá ser utilizado um NFS para cada MV presente na IV.

Estes módulos do MigraVI-OpenStack executam tanto na nuvem de origem quando na nuvem de destino. Dessa forma, o MigraVI-OpenStack pode ser dividido em dois algoritmos (Seção 4.5), sendo que um é executado na nuvem de origem e outro na nuvem de destino.

5.3 ATENDIMENTO DOS REQUISITOS

Essa seção tem como objetivo apresentar o atendimento dos requisitos funcionais pelas ferramentas utilizadas. Conforme apresentado na Seção 5.1, diversas ferramentas foram utilizadas no desenvolvimento do protótipo MigraVI-OpenStack. Dessa forma, a Tabela 3 apresenta o relacionamento das ferramentas utilizadas com a sua finalidade em atender os requisitos funcionais apresentados na Seção 4.2.

Tabela 3 – Atendimento aos requisitos funcionais.

RF	Ferramenta	Dependências tecnológicas
RF1	Linguagem de programação Python.	
RF2	Convoy, Docker, CRIU	Utilizado Convoy para realizar a migração dos dados salvos nos volumes dos contêineres.
RF3	Flame, Heat, OpenStack	Utilizado Flame para realizar o levantamento dos dados da rede.
RF4	–	
RF5	Heat	Utilizado Heat para a orquestração da rede.
RF6	NFS	Utilizado NFS pois o Convoy sincroniza os dados entre os provedores através de um NFS.
RF7	–	
RF8	Heat	

Fonte: O próprio autor

A Tabela 3 também apresenta a dependência tenológica que a utilização de uma determinada ferramenta gerou no desenvolvimento do protótipo. Conforme é possível observar, somente os requisitos RF4 e RF7 não foram utilizadas nenhuma ferramenta para atende-los. Isso ocorre pois para atender o RF4 o sistema por padrão não armazena os dados fornecidos pelo usuários e o atendimento do RF7 faz-se devido a forma da migração realizada. Ou seja, em algum determinado momento todas as aplicações serão pausadas para serem transferidas para a nuvem de destino.

5.4 CONSIDERAÇÕES PARCIAIS

Esse capítulo teve como objetivo apresentar o MigraVI-OpenStack, um protótipo para a realização de migração de IV entre provedores diferentes. A aplicação foi desenvolvida utilizando a linguagem de programação Python e depende outras de ferramentas para alcançar seu objetivo. A leitura da especificação da rede é realizada

pelo Flame, o qual é responsável por gerar o modelo HOT que descreve a IV. Já para o gerenciamento dos volumes dos contêineres é utilizado o Convoy, o qual permite que os dados dos contêineres salvos em volumes sejam migrados juntamente com o contêiner.

Todas as funcionalidades do MigraVI-OpenStack estão dividas em três módulos principais. Um módulo gerenciador, um módulo para leitura da IV e outro para a migração dos contêineres.

Dessa forma, apesar de ser um protótipo baseado em OpenStack, se forem realizadas as devidas alterações para realizar a leitura da especificação da IV em outros provedores de nuvem como, por exemplo, a Amazon, o MigraVI pode ser utilizado para migrar para provedores que não utilizem o Heat como mecanismo de modelagem de infraestruturas.

6 ANÁLISE EXPERIMENTAL

Este capítulo tem como objetivo descrever a análise experimental realizada com o protótipo MigraVI-OpenStack. Na Subseção 6.1 são apresentados os ambientes onde é realizado os testes de migração. Neste caso, foram utilizados os ambientes da Nuvem privada Tche e da nuvem do CloudLab. Na sequencia, a Seção 6.2 são apresentados os cenários de migração utilizado na realização dos testes. São apresentados quatro cenários que tem como objetivo representar migrações de IV entre zonas diferentes de um mesmo provedor e migração entre provedores distintos. As métricas utilizadas para a análise dos testes e os resultados são apresentados nas Seções 6.3 e 6.4 respectivamente. Por fim, a Seção 6.5 apresenta as considerações parciais do capítulo.

6.1 AMBIENTES DE EXPERIMENTAÇÃO

Para análise do protótipo desenvolvido, dois ambientes de experimentação foram selecionados. Inicialmente, a privada Tche (Subseção 6.1.1) foi utilizada para realizar os testes iniciais do protótipo. Em um segundo momento, a nuvem CloudLab (Subseção 6.1.2) foi selecionada para verificar a aplicabilidade da proposta em nuvens conectadas por enlaces de longa distância. Embora o protótipo seja baseado em OpenStack, a arquitetura proposta não é limitada pelas configurações internas dos provedores.

6.1.1 Nuvem Tche

Nesta seção são apresentados os cenários de migração realizados utilizando o MigraVI-OpenStack, utilizando a infraestrutura da Nuvem Tche. A nuvem utilizada para a realização dos experimentos dos Cenários 1 e 2 (Seção 6.2 é denominada Nuvem Tche. Essa nuvem é mantida no Laboratório de Processamento Paralelo e Distribuído (LabP2D)[1] localizado na Universidade do Estado de Santa Catarina (UDESC).

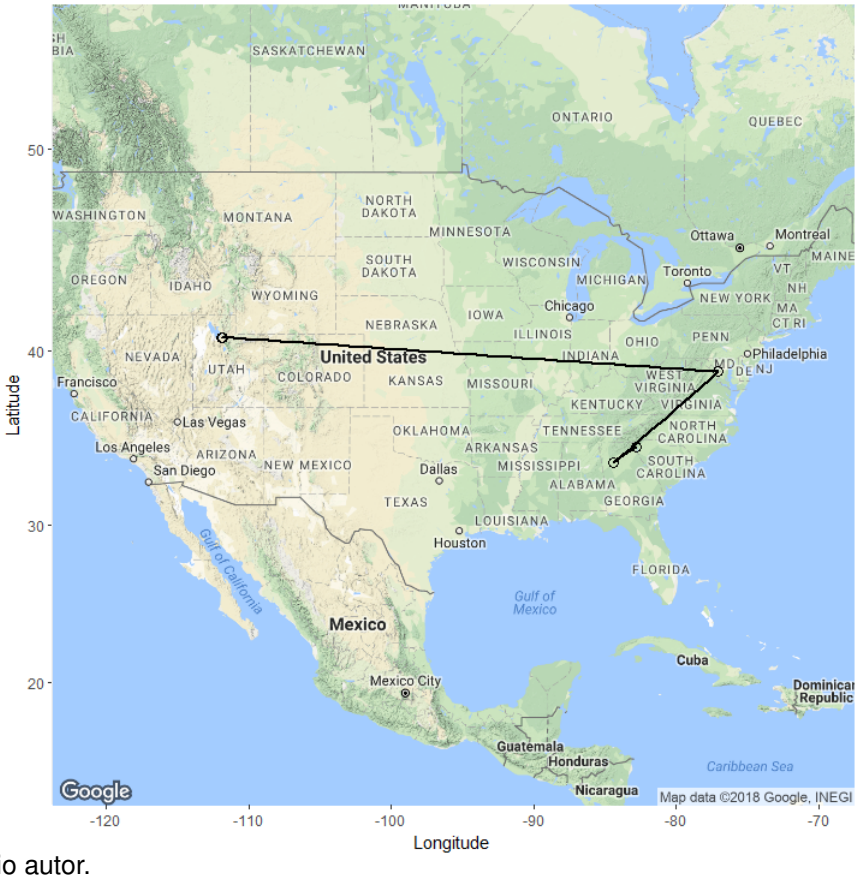
A Nuvem Tche é composta por trinta servidores (*desktops* e servidor de *rack*) e é gerenciada pelo OpenStack. Quanto ao gerenciamento dos recursos, a nuvem utiliza um servidor para gerenciamento de imagens e instalações (Fuel (FUEL, 2017)), dois controladores e três nós de armazenamento. Para a virtualização das MVs é utilizado o hipervisor KVM. Os recursos são interconectados por *switches* 1Gbps. Referente a rede interna da Nuvem Tche, ela é totalmente baseada nos princípios do paradigma SDN (KREUTZ et al., 2015).

[1] Informações disponíveis em: <http://labp2d.joinville.udesc.br/>.

6.1.2 CloudLab

A infraestrutura de pesquisa em computação em nuvem utilizada para a realização dos Cenários 3 e 4 6.2 foi o CloudLab, constituída, atualmente, por 3 sítios distribuídos geograficamente nos Estados Unidos da América (CLOUDLAB, 2018). As nuvens escolhidas para realização dos testes foram: a localizada no estado de Utah, como origem, e Clemson a localizada no estado de South Caroline, como destino. Com esta escolha, pode-se caracterizar a migração da IV de Utah (oeste americano) para South Caroline (leste americano) como uma migração de longa distância apoiada em uma rede *Wide Area Network* (WAN). A latência observada entre os dois sítios foi de aproximadamente $9187ms$ com um RTT médio de 51.853. A Figura 13 apresenta o caminho da migração realizada, ou seja, nela são indicados os roteadores pelos quais os dados passam saindo de Utah até chegar em Clemson.

Figura 13 – Rota de migração.



Fonte: O próprio autor.

Para verificar o caminho realizado pela migração foi utilizada a ferramenta *tracerout*. Através do uso dessa ferramenta foi possível mapear por quais roteadores os pacotes de dados da migração passaram. O caminho apresentado visualmente na Figura 13, é o resultado agrupado da Tabela 4, a qual apresenta todos os roteadores por quais os pacotes de dados trafegaram.

Tabela 4 – Roteadores de Utah a Clemson.

IP	**Região - Cidade**	RTT
host-10-11-10-3.migravi-src.migrationivs-pg0.utah.cloudlab.us (10.11.10.3)	Utah - Salt Lake City	0.403 ms
128.110.103.241	Utah - Salt Lake City	0.946 ms
140.197.247.214	Utah - Salt Lake City	1.226 ms
140.197.253.0	Utah - Salt Lake City	1.598 ms
et-4-3-0.1011.rtsw.salt.net.internet2.edu (198.71.45.230)	Distrito de Columbia - Washington	1.113 ms
ae-5.4079.rtsw.kans.net.internet2.edu (162.252.70.144)	Distrito de Columbia - Washington	21.081 ms
ae-3.4079.rtsw.chic.net.internet2.edu (162.252.70.140)	Distrito de Columbia - Washington	32.005 ms
ae-2.4070.rtsw.atla.net.internet2.edu (198.71.45.61)	Distrito de Columbia - Washington	49.286 ms
sox-to-i2-100g.sox.net (143.215.193.3)	Distrito de Columbia - Washington	49.307 ms
xe-1-1-1-816-t01-cle.culr.net (205.186.63.1)	Carolina do Sul - Clemson	51.909 ms
205-186-62-65.generic.c-light.net (205.186.62.65)	Carolina do Sul - Clemson	52.430 ms
205-186-62-66.generic.c-light.net (205.186.62.66)	Carolina do Sul - Clemson	52.783 ms
130.127.132.213 (130.127.132.213)	Carolina do Sul - Clemson	52.245 ms

Fonte: O próprio autor

Conforme é possível observar, na Tabela 4, foram necessários um total de treze saltos até chegar no destino da migração. Esses saltos ocorreram em três áreas Utah, Washington e Clemson. Dessa forma, ao percorrer todo esse trajeto para realizar a migração, esse cenário se assemelha a uma migração real.

6.2 CENÁRIOS DE MIGRAÇÃO

Esta seção tem como objetivo identificar como os cenários de testes propostos fora aplicados no protótipo MigraVI-OpenStack. No total são propostos quatro cenários, de maneira que os Cenários 1 e 2 são executados utilizando a infraestrutura de Nuvem Tche 6.1.1 e os Cenários 3 e 4 são executados na nuvem criada no CloudLab. Os Cenários 1 e 2 são executados na Nuvem Tche pois representam uma migração entre zonas distintas de um mesmo provedor. Já os Cenários 3 e 4 representam migrações entre provedores distintos. Dessa forma, é importante ressaltar que uma comparação cruzada entre os testes não é realizada, pois representam testes realizados em infraestruturas diferentes.

O Cenário 1, no qual é realizada a migração somente de um único contêiner, é composto por um *switch* que interconecta duas MVs, $MV1$ e $MV2$ respectivamente. Além destes recursos, neste cenário, tem-se um contêiner executando na $MV1$, o qual será migrado para a $MV2$ através da *Local Area Network* (LAN).

O Cenário 2, é composto por dois projetos diferentes ambos localizados na Nuvem Tche, projetos $PRJ1$ e $PRJ2$ respectivamente. Em um primeiro momento o projeto $PRJ2$ estará vazia, ou seja, nenhum IV está configurada nele, este projeto será o destino da migração. Já o projeto $PRJ1$, tem uma IV configurada. Essa IV é composta por um *switch*, o qual interconecta duas MVs, $MV1$ e $MV2$ respectivamente. Para realização do teste, 10 contêineres estão sendo executados dentro da $MV1$ e nenhum contêiner está executando na $MV2$. Ao ser realizada a migração da IV do projeto $PRJ1$ para o projeto $PRJ2$, uma IV' deve ser criada. A IV', criada no

$PRJ2$, deve possuir um *switch* que interconecta as MVs $MV1'$ e $MV2'$ que são correspondentes as $MV1$ e $MV2$ do projeto $PRJ1$. Neste cenário de teste é importante observar que os contêineres executando dentro da $MV1$ devem ser migrados para a sua MV correspondente no projeto $PRJ2$, que neste caso é a $MV1'$.

Os Cenários 3 e 4 são similares aos cenários 1 e 2 respectivamente, mas diferem-se por tratar de uma migração através da WAN. Além disso, é importante ressaltar que durante os testes realizados cada contêiner criado possuía um tamanho aproximado de 30 mb. As métricas utilizadas nos testes preliminares e os resultados obtidos são descritos nas Seções 6.3 e 6.4 respectivamente.

6.3 MÉTRICAS PARA ANÁLISE

O protótipo do MIGRAVI-OPENSTACK foi avaliado sob a ótica de duas métricas principais:

1. Tempo total de migração de uma IV; e
2. Consumo de rede para migração.

A métrica tempo total de migração foi estratificada por etapa. O objetivo é possibilitar o mapeamento tempo/etapa ao longo da migração. Embora a migração de IV corresponda a uma tarefa administrativa com planejamento prévio, a análise temporal permite identificar o período de indisponibilidade do serviço. Com a estimativa de indisponibilidade do serviço, o cliente pode avaliar o impacto deste tempo na sua aplicação e, por sua vez, a possibilidade ou não de realizar a migração de sua IV entre os provedores previamente definidos.

6.4 RESULTADOS

Faz-se importante ressaltar que, além dos resultados obtidos com o protótipo, um corretor que realize a migração de IV entre provedores de nuvens distintos, auxilia o usuário no momento de decisão de migração para um novo provedor. Isso ocorre devido ao fato de que um corretor automatiza o processo de recriação da IV no provedor de destino, fazendo com que o usuário tenha que empreender uma quantidade menor de tempo na tarefa de migração. Além disso, ao automatizar o processo de migração, reduz-se o risco de erros e falhas que possam ocorrer devido a falta ou até mesmo a má configuração de algum recurso computacional pertencente a IV. Dessa forma, com o objetivo de representar a perspectiva do usuário da nuvem, o tempo total da migração de uma IV foi contabilizado.

6.4.1 Resultados Cenários 1 e 2

Conforme apresentado na Seção 4.4, diversas tarefas devem ser cumpridas para realizar a migração, dessa forma o tempo total da migração de uma IV foi decomposto por tarefa e analisado. A apresentação destes resultados destes testes e a análise dos mesmos são apresentados. O método utilizado compreende na execução de 10 migrações para os Cenários 1 e 2. Devido a maior variabilidade dos resultados (WAN), para os Cenários 3 e 4 foram realizadas 30 migrações.

A migração de um único contêiner é um processo rápido, ficando com uma média de execução de 3.4125 segundos e com um desvio padrão de 0.37999 segundos. No Cenário 2 o objetivo é a realizar a migração de uma IV composta por um *switch*, duas MVs, uma rede virtual de maneira que em uma das MVs estão executando dez contêineres. Neste cenário a migração da IV ocorre entre projetos diferentes, na mesma nuvem. Neste cenário as migrações levam em média aproximadamente 2 minutos e 56 segundos com um desvio padrão de 10 segundos.

6.4.2 Resultados Cenários 3 e 4

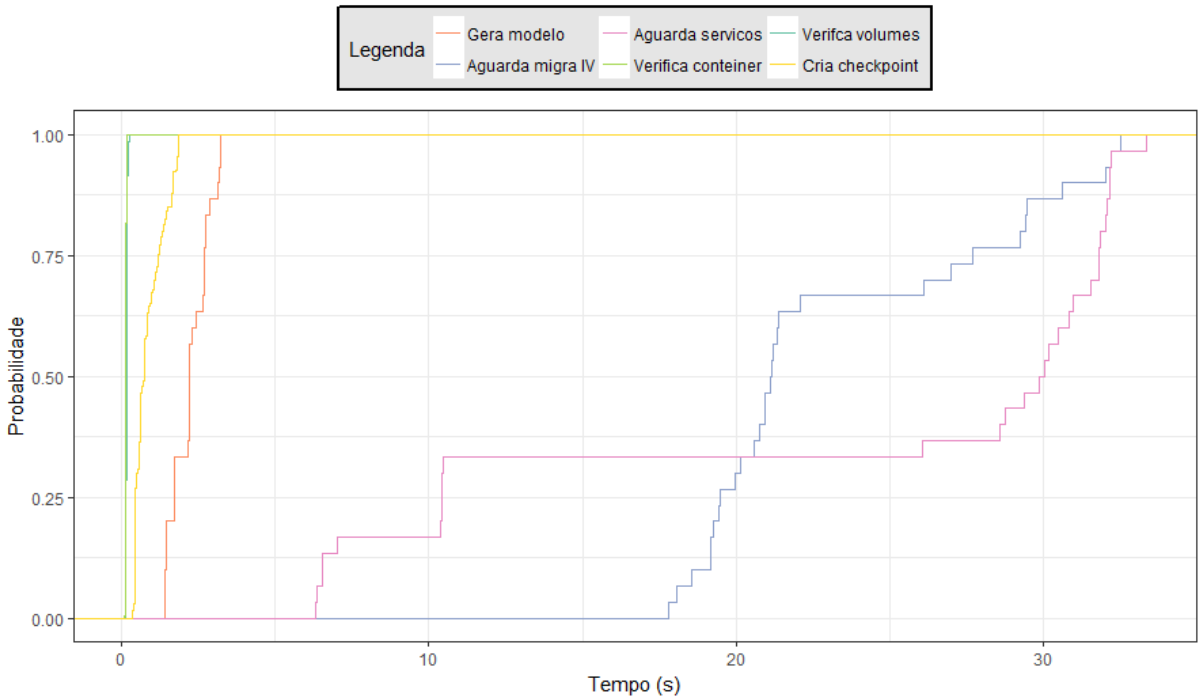
Os Cenários 3 e 4, correspondem a migração entre nuvens distintas, dessa forma são os cenários que mais se assemelham a um ambiente real e, por isso, mais execuções foram feitas para analisar também o consumo da rede. No Cenário 3, constatou-se que a média das execuções da migração dos contêineres ficou em $97,34$ segundos e um desvio padrão de 7.05 segundos. A execução desta etapa restringe-se a transferência dos contêineres da origem para o destino, sem adequações na configuração das IVs. Nesse cenário já é possível observar a ocorrência de um aumento no tempo da migração de um contêiner em comparação ao Cenário 1. Esse aumento deve-se ao fato de houve um maior consumo de tempo para realizar a sincronização dos arquivos entre as nuvens de origem e destino. Por fim, o Cenário 4 representa a migração de uma IV entre provedores. Esse cenário apresentou um tempo médio de $172,81$ segundos com um desvio padrão de 13.01 segundos.

Com o objetivo de exemplificar o tempo de migração de uma IV, as duas etapas da migração realizadas pelo **GeraVI** e **RecriaVI**, são analisadas individualmente, nas Figuras 14 e 15, respectivamente. Ambas apresentam, no eixo x, o tempo em segundos gasto por cada atividade e, no eixo y, a probabilidade de cada tarefa executar em um tempo específico, ao longo da migração. A fase **GeraVI** é decomposta em 6 atividades e a **RecriaVI** em 5. Dessa forma, foi aplicada uma função de distribuição acumulada (CFD) com objetivo de representar a probabilidade de cada tarefa executada na migração aconteça em qual instante de tempo. Com os resultados da aplicação da função de distribuição acumulada é possível analisar todas as atividades e identificar qual deles consome maior tempo da migração e assim identificar possíveis

pontos de melhoras no protótipo.

Como resultado da análise da Figura 14, pode-se observar que as atividades *Gera Modelo*, *Verifica Contêiner*, *Cria checkpoint* e *Verifica Volume* finalizam em menos de 3 segundos. Todas executam inteiramente na nuvem de origem orquestrada pelo **GeraVI**. Por outro lado, as duas atividades restantes, *Aguarda migra IV* e *Aguarda Serviços*, são as mais custosas do ponto de vista tempo de execução. A característica comum entre elas é o fato de interagirem com **RecriaVI** (localizado na nuvem de destino). A atividade *Aguarda migra IV* é a fase em que a GeraIV espera a criação da IV na nuvem de destino, para posteriormente prosseguir com a verificação dos contêineres e volumes e com a criação dos *checkpoints*. A atividade *Aguarda Serviços*, na nuvem de origem, aguarda a estabilização dos serviços na nuvem de destino, por exemplo, o SSH.

Figura 14 – Tempo de execução das tarefas efetuadas no provedor de origem pelo módulo **GeraVI**.

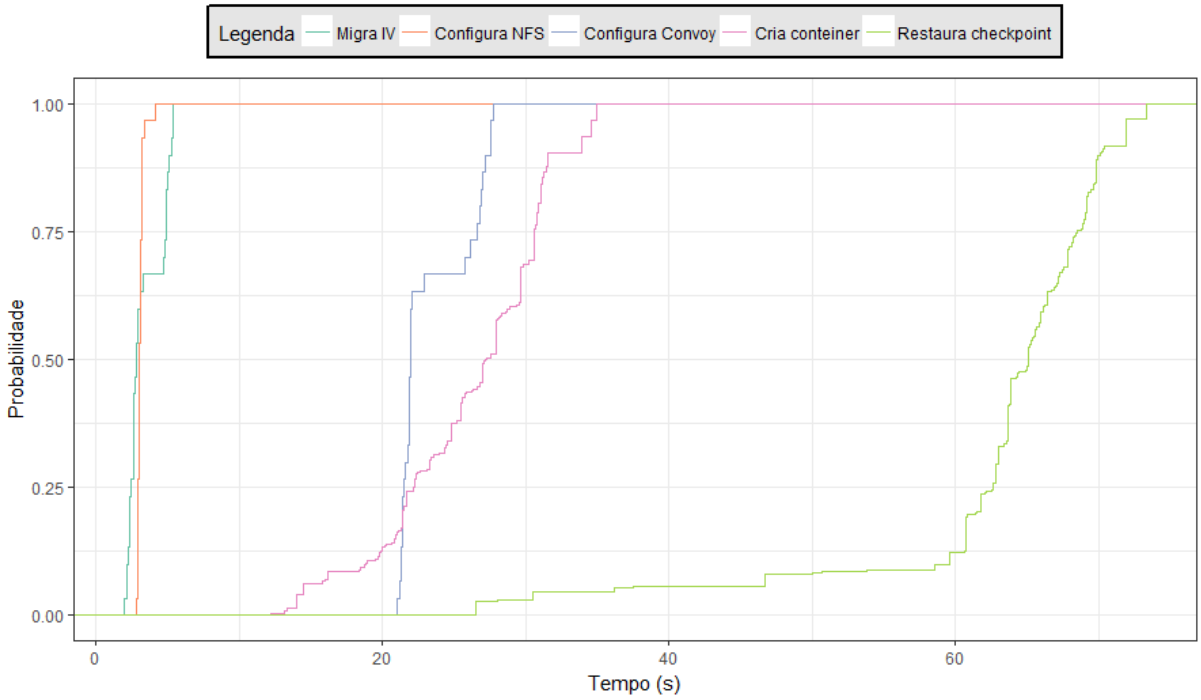


Fonte: O próprio autor.

A Figura 15 apresenta o resultado da função distribuição aplicada sobre os tempos de execução das atividades realizadas pelo **RecriaVI**. A atividade *Aguarda migra IV*, modelado na nuvem de origem, é decomposto nas atividades: *Configura NFS*, *Configura Convoy* e *Migra IV*, na nuvem de destino. A atividade *Migra IV* é responsável pela criação das MV, grupos de segurança e da rede, configuração das chaves de acessos às MVs . Constatou-se que os menores tempos entre atividades do **RecriaVI** foram com as atividades de configuração do NFS e *Migra IV*, obtendo 7 segundos, no pior caso. Em contrapartida, a atividade de configuração do Convoy atingiu valores

entre 25 e 30 segundos e *Cria contêineres* entre 12 e 35 segundos. Os valores mais instáveis obtidos foram com a atividade *cria checkpoint* que variou de 30 segundos a 1 minuto. O maior grau de probabilidade indica o tempo desta atividade mais próximo a 1 minuto. A atividade de *checkpoint* é sensível a configuração de intervalo entre os *checkpoint* na nuvem de origem. Caso o conteúdo do último *checkpoint* já tenha sido transferido para a nuvem de destino, a restauração é praticamente instantânea (próxima a 30 segundos). Caso contrário, ele deve aguardar a transferência dos dados da nuvem de origem, podendo atingir até 1 minuto. Ressalta-se, ainda, que o aumento no tamanho dos arquivos de *checkpoint* e/ou dos volumes a serem migrados, deve provocar degradação no desempenho da migração da IV.

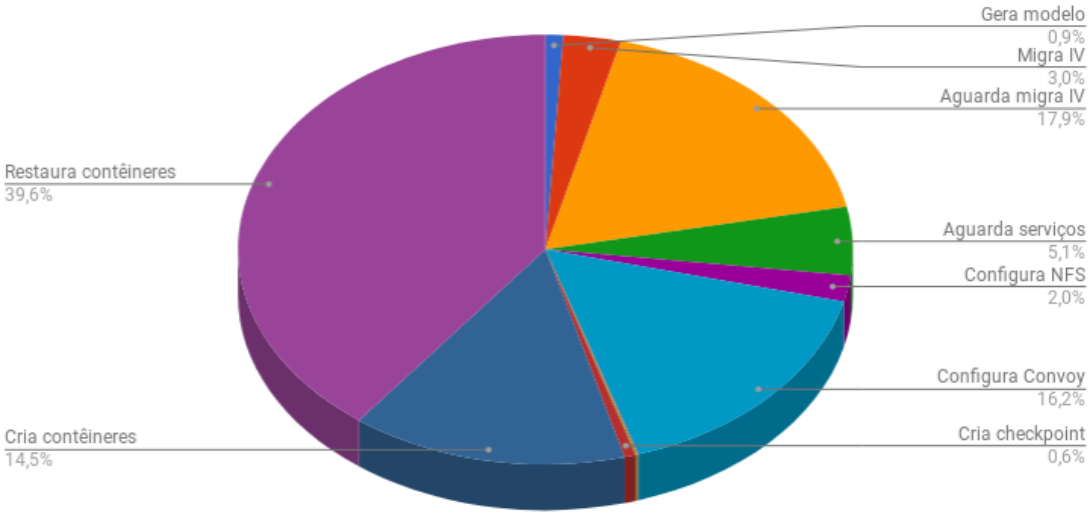
Figura 15 – Tempo de execução das tarefas efetuadas no provedor de destino pelo módulo **RecriaVI**.



Fonte: O próprio autor.

As análises das Figuras 14 e 15 basearam-se na probabilidade de ocorrência de cada atividade. Por sua vez, os resultados subsequentes focaram na quantificação do uso de rede e do tempo associado a cada atividade. Na Figura 16, tem-se o tempo consumido por cada atividade durante a migração de uma IV. Importante ressaltar que na Figura 16 as atividades *Verifica contêiner* e *Verifica volume* não são apresentadas pois somando as duas representam um porcentual ínfimo comparadas com as demais, aproximadamente $0,13\%$. Os resultados em percentuais, permite destacar dois comportamentos distintos, as atividades *Restaura contêiner*, *Cria Contêiner*, *Configura Convoy*, *Aguarda Migra IV* são as maiores consumidoras de tempo, tendo as demais, valores inferiores a $5,1\%$.

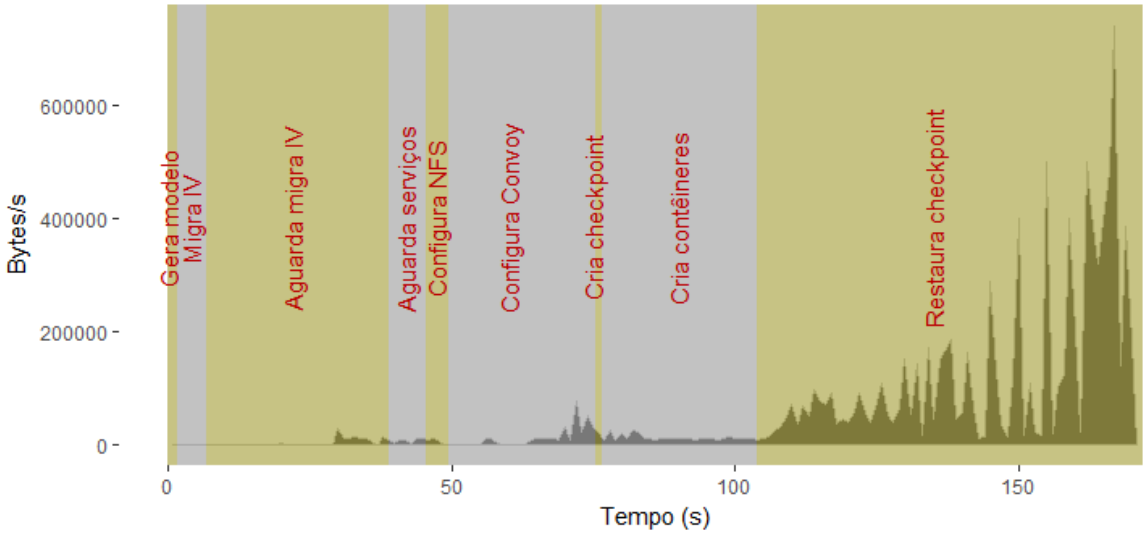
Figura 16 – Percentual dos tempos de execuções das tarefas durante a migração de uma IV.



Fonte: O próprio autor.

A Figura 17 apresenta a vazão da rede durante o período de migração de uma IV, associando as atividades de migração ao consumo de rede. Para este gráfico, escolheu-se ao acaso, 1 entre os 10 testes da amostra.

Figura 17 – Utilização da rede durante a migração de uma IV.



Fonte: O próprio autor.

Dessa forma, é possível observar o quanto cada atividade pode impactar no uso da rede durante a migração. Como é possível observar, a atividade que mais consume rede é a de restauração do contêineres (*restaura checkpoint*), decorrente da transferência dos *checkpoint*, via NFS. O consumo de rede das demais atividades é

ínfimo (entre 224 e 76780 Bytes/s, se comparado com a atividade *restaura checkpoint* (pico de 594500 Bytes/s).

6.5 CONSIDERAÇÕES PARCIAIS

Este capítulo teve como principal objetivo realizar uma análise experimental do protótipo implementado. Para a realização da análise quatro cenários de migração foram propostos. Os cenários de migração de contêineres tem como objetivo definir uma base de tempo para a migração de contêineres e assim saber o porcentual que essa etapa da migração tem representa na migração completa. Os cenários de migração de uma IV, são os cenários nos quais é possível avaliar o processo de migração como um todo. Nestes cenários constatou-se que a sincronização de arquivos entre as nuvens é a etapa que mais impacta na migração. Mostrando dessa forma que outras tecnologias ou ferramentas possam ser estudadas e consideradas para a realização dessa sincronização de maneira que tenha um impacto menor no processo de migração.

7 CONSIDERACÕES FINAIS

Em ambientes de computação em nuvem, os usuários podem reservar e gerenciar recursos computacionais de acordo com os requisitos de suas aplicações. Nesse cenário, os recursos são provisionados e tarifados de acordo com o uso, sendo economicamente atrativos para provedores e usuários. Entretanto, a ampla aceitação e popularização do modelo de computação em nuvem resultaram na proliferação de provedores de serviço. Atualmente, diversos provedores de nuvem (*e.g.,* Amazon, Google e Microsoft) oferecem serviços e recursos em um mercado competitivo. Uma das alternativas para obter destaque em tal mercado é diferenciação dos serviços e a otimização interna dos recursos computacionais e de comunicação. Embora promissoras, as abordagens criam âncoras de provisionamento pois são específicas para cada provedor.

É evidente que os usuários enfrentam dificuldades em migrar suas IVs entre diferentes provedores. A falta de padronização das APIs aliada com as abordagens atuais de gerenciamento centradas nos provedores, criam uma barreira. Diante dos desafios observados, o presente trabalho introduziu uma proposta para a realização da migração de IVs entre provedores distintos, denominada MigraVI.

Cada recurso virtualizado de um IaaS é uma unidade de migração que pode ser migrado separadamente como, por exemplo, aplicação, MVs, contêineres e por fim, toda a IV. A migração da IV agrega os desafios da migração de cada uma dessas unidades ao desafio de migrar as configurações da RV. É importante ressaltar o desafio relacionado com a migração da RV: as aplicações hospedadas na IV que está sendo migrada, devem se comunicar novamente de maneira transparente para os usuários (administrativo e final). Ou seja, a migração deve ser realizada e as configurações da rede devem ser mantidas inalteradas, sem a necessidade de que o usuário reconfigure estas configurações (*e.g.*, endereços de IPs e canais de comunicação).

Durante a implementação do protótipo diversos provedores foram testados para realizar a migração. Entretanto as migrações só foram factíveis na nuvem Tche e nos provedores do CloudLab. Ao realizar testes com o RackSpace, constatou-se que o mesmo não fornecia suporte para a criação e configuração de redes externas. Era necessário utilizar as configurações fornecidas pelo provedor, isso impactou na migração pois, dessa maneira não é possível replicar todas as configurações do provedor de origem no de destino, neste caso o RackSpace. Entretanto, observou-se que neste caso o RackSpace poderia ser utilizado com nuvem de origem para a migração, desde que a nuvem de destino prove-se mais recursos de configuração do que a nuvem de

origem. Outro provedor utilizado foi o YellowCircle, entretanto a migração para este provedor também não foi possível devido à dificuldade em enviar uma imagem com mais de 3 GB para ser utilizada pelas MVs. Dessa forma estes dois provedores foram descartados dos testes e foi utilizado somente a Nuvem Tche e o CloudLab.

Conforme apresentado no Capítulo 5 o protótipo MigraVI-OpenStack foi implementado em módulos, os quais são responsáveis por realizar todas as tarefas definidas como essenciais para a solução. Entretanto, o protótipo foi desenvolvido baseado em nuvens OpenStack. Com a utilização do OpenStack e do projeto Heat foi possível realizar a migração das configurações da rede interna da IV. Porém, com as devidas alterações no módulo de levantamento de informações da IV, o protótipo poderá ser utilizado para outras nuvens.

É importante ressaltar que todos os pré-requisitos e requisitos levantados no Capítulo 4 foram atendidos em sua totalidade. Além disso, o objetivo de realizar uma migração de uma IV entre provedores distintos também foi atingido e com a finalidade de analisar essa migração, diversos testes foram realizados com o protótipo.

O testes realizados com protótipo mostraram que durante a migração de uma IV, composta por um *switch* e dez contêineres, a aplicação hospedada ficou indisponível por aproximadamente $39,6\%$ do tempo total de uma migração entre provedores distintos através da WAN. É importante ressaltar que a migração realizada não é *live*, ou seja, durante o processo todas as aplicações são interrompidas e, após a transferência dos dados, elas são recuperadas na nuvem de destino.

7.1 TRABALHOS FUTUROS

Conforme apresentado no Capítulo 5, o protótipo implementado neste trabalho está focado no OpenStack. Entretanto, a arquitetura proposta é independente de tecnologias. Como trabalhos futuros, novos protótipos podem ser implementados ou uma solução canônica, completamente baseada na arquitetura proposta.

Especificamente sobre o protótipo em OpenStack, de acordo com o apresentado na Seção 6.4, a utilização de um NFS para a sincronização dos dados entre as duas nuvens pode impactar no tempo de migração, tornando assim necessária uma análise para a utilização de outras ferramentas que possam atender os requisitos da proposta.

Adicionalmente, um trabalho relevante é analisar a possibilidade de realizar a migração de contêineres que sejam gerenciados por ferramentas de orquestração de contêineres, *e.g.,* Kubernetes e Swarm.

7.2 PUBLICAÇÕES

Durante o desenvolvimento do presente trabalho, quatro publicações foram realizadas. São elas:

- Revista Brasileira de Computação Aplicada (RBCA). Título: Uma Taxonomia para Corretagem de Nuvens IaaS baseada na Migração de Infraestruturas Virtuais. Autores: Euclides C. Júnior, Charles C. Miers, Guilherme P. Koslovski. Ano: 2016;
- Escola Regional de Alto Desempenho (ERAD). Título: Um Mecanismo para Migração de Infraestruturas Virtuais entre Provedores de Nuvem IaaS. Autores: Euclides C. Júnior, Charles C. Miers, Guilherme P. Koslovski. Ano: 2017;
- ERAD. Título: MigraVI: Uma proposta para Migração de Infraestruturas Virtuais em Ambientes de Computação em Nuvem. Autores: Euclides C. Júnior, Charles C. Miers, Guilherme P. Koslovski. Ano: 2018; e
- Conferência Latino-americana de Informática (CLEI): Virtual Infrastructures on the Move: Containers and Virtual Network Migration. Autores: Euclides C. Júnior, Charles C. Miers, Maurício A. Pillon, Fernando F. Redigolo, Guilherme P. Koslovski. Ano 2018.

REFERÊNCIAS

AKOUSH, S. et al. Predicting the performance of virtual machine migration. In: **2010 IEEE International Symposium on Modeling, Analysis and Simulation of Computer and Telecommunication Systems**. [S.l.: s.n.], 2010. p. 37–46. ISSN 1526-7539.

ALI, H.; MOAWAD, R.; HOSNI, A. A. F. A cloud interoperability broker (cib) for data migration in saas. **Future Computing and Informatics Journal**, Elsevier, 2017.

ANHALT, F.; KOSLOVSKI, G.; PRIMET, P. V.-B. Specifying and provisioning virtual infrastructures with hipernet. **International Journal of Network Management**, John Wiley & Sons, Ltd., v. 20, n. 3, p. 129–148, 2010. ISSN 1099-1190. Disponível em: <http://dx.doi.org/10.1002/nem.732>.

BARHAM, P. et al. Xen and the art of virtualization. In: ACM. **ACM SIGOPS operating systems review**. [S.l.], 2003. v. 37, n. 5, p. 164–177.

BHARDWAJ, S.; JAIN, L.; JAIN, S. Cloud computing: A study of infrastructure as a service (iaas). **International Journal of engineering and information Technology**, v. 2, n. 1, p. 60–63, 2010.

BIEDERMAN, E. W.; NETWORX, L. Multiple instances of the global linux namespaces. In: CITESEER. **Proceedings of the Linux Symposium**. [S.l.], 2006. v. 1, p. 101–112.

BUYYA, R.; VECCHIOLA, C.; SELVI, S. T. **Mastering Cloud Computing Foundations and Applications Programming**. Morgan Kaufmann, 2013. ISBN 978-0124114548. Disponível em: <http://gen.lib.rus.ec/book/index.php?md5=A282DF11A1FB70360F3642094EFAEB13>.

CARRASCO, J.; DURÁN, F.; PIMENTEL, E. Component-wise application migration in bidimensional cross-cloud environments. ScitePress, 2017.

CELESTI, A. et al. Exploring container virtualization in iot clouds. **2016 IEEE International Conference on Smart Computing (SMARTCOMP)**, p. 1–6, 2016.

CHOWDHURY, N. M. M. K.; BOUTABA, R. Network virtualization: State of the art and research challenges. **Comm. Mag.**, IEEE Press, Piscataway, NJ, USA, v. 47, n. 7, p. 20–26, jul. 2009. ISSN 0163-6804. Disponível em: <http://dx.doi.org/10.1109/MCOM.2009.5183468>.

CHOY, S. et al. The brewing storm in cloud gaming: A measurement study on cloud to end-user latency. In: **2012 11th Annual Workshop on Network and Systems Support for Games (NetGames)**. [S.l.: s.n.], 2012. p. 1–6. ISSN 2156-8138.

CLARK, C. et al. Live migration of virtual machines. In: USENIX ASSOCIATION. **Proceedings of the 2nd Conference on Symposium on Networked Systems Design & Implementation-Volume 2**. [S.l.], 2005. p. 273–286.

CLOUDLAB. 2018. <https://www.cloudlab.us/index.php>. Acessado em: 14-04-2018.

CORRADI, A.; FANELLI, M.; FOSCHINI, L. Vm consolidation: A real case based on openstack cloud. **Future Generation Computer Systems**, Elsevier, v. 32, p. 118–127, 2014.

DESHPANDE, U.; KEAHEY, K. Traffic-sensitive live migration of virtual machines. **Future Generation Computer Systems**, v. 72, p. 118 – 128, 2017. ISSN 0167-739X. Disponível em: <http://www.sciencedirect.com/science/article/pii/S0167739X16301133>.

DOCKER. **Docker - Build, Ship, and Run Any App, Anywhere**. 2017. <Docker:https://www.docker.com/>. Acessado em: 10-11-2017.

DUA, R.; RAJA, A. R.; KAKADIA, D. Virtualization vs containerization to support paas. In: IEEE. **Cloud Engineering (IC2E), 2014 IEEE International Conference on**. [S.l.], 2014. p. 610–614.

DUGGAN, M. et al. A network aware approach for the scheduling of virtual machine migration during peak loads. **Cluster Computing**, Springer, p. 1–12, 2017.

EC2, A. **Amazon EC2**. 2017. <https://aws.amazon.com/pt/ec2/>. Acessado em: 14-10-2017.

FUEL. **Fuel**. 2017. <https://wiki.openstack.org/wiki/Fuel#What_is_Fuel.3F>. Acessado em: 27-11-2017.

GARCÍA, A. L.; CASTILLO, E. Fernández del; FERNáNDEZ, P. O. Standards for enabling heterogeneous iaas cloud federations. **Comput. Stand. Interfaces**, Elsevier Science Publishers B. V., Amsterdam, The Netherlands, The Netherlands, v. 47, n. C, p. 19–23, ago. 2016. ISSN 0920-5489. Disponível em: <http://dx.doi.org/10.1016/j.csi.2016.02.002>.

GARFINKEL, T.; ROSENBLUM, M. et al. A virtual machine introspection based architecture for intrusion detection. In: **Ndss**. [S.l.: s.n.], 2003. v. 3, n. 2003, p. 191–206.

GELAIN, A. et al. Uma breve história da computação aplicada no brasil. **Revista Brasileira de Computação Aplicada**, v. 6, n. 2, p. 123–135, 2014.

GROZEV, N.; BUYYA, R. Inter-cloud architectures and application brokering: taxonomy and survey. **Software: Practice and Experience**, v. 44, n. 3, p. 369–390, 2014. ISSN 1097-024X. Disponível em: <http://dx.doi.org/10.1002/spe.2168>.

HANDLEY, M. Why the internet only just works. **BT Technology Journal**, Springer, v. 24, n. 3, p. 119–129, 2006.

HEAT. **Heat - OpenStack**. 2017. <https://wiki.openstack.org/wiki/Heat>. Acessado em: 10-11-2017.

HERVE, T.; REZMERITA, A. **FLAME: Automatic Heat template generation**. 2017. <https://github.com/openstack/flame>. Acessado em: 20-11-2017.

HINES, M. R.; DESHPANDE, U.; GOPALAN, K. Post-copy live migration of virtual machines. **SIGOPS Oper. Syst. Rev.**, ACM, New York, NY, USA, v. 43, n. 3, p. 14–26, jul 2009. ISSN 0163-5980. Disponível em: <http://doi.acm.org/10.1145/1618525.1618528>.

HOFMANN, P.; WOODS, D. Cloud computing: The limits of public clouds for business applications. **IEEE Internet Computing**, v. 14, n. 6, p. 90–93, Nov 2010. ISSN 1089-7801.

JAIN, R.; PAUL, S. Network virtualization and software defined networking for cloud computing: a survey. **IEEE Communications Magazine**, IEEE, v. 51, n. 11, p. 24–31, 2013.

JAMSHIDI, P.; AHMAD, A.; PAHL, C. Cloud migration research: a systematic review. **IEEE Transactions on Cloud Computing**, IEEE, 1, n. 2, p. 142–157, 2013.

JUNIOR, E. C.; MIERS, C. C.; KOSLOVSKI, G. P. Uma taxonomia para corretagem de nuvens iaas baseada na migração de infraestruturas virtuais. **Revista Brasileira de Computação Aplicada**, v. 9, n. 1, p. 15–30, 2017.

KAUR, K.; SHARMA, D. S.; KAHLON, D. K. S. Interoperability and portability approaches in inter-connected clouds: A review. **ACM Comput. Surv.**, ACM, New York, NY, USA, v. 50, n. 4, p. 49:1–49:40, out. 2017. ISSN 0360-0300. Disponível em: <http://doi.acm.org/10.1145/3092698>.

KIRKPATRICK, K. Software-defined networking. **Communications of the ACM**, ACM, v. 56, n. 9, p. 16–19, 2013.

KOSLOVSKI, G. P. **Dynamically provisioned virtual infrastructures: specification, allocation and execution**. Tese (Doutorado) — Ecole normale supérieure de lyon-ENS LYON, 2011.

KREUTZ, D. et al. Software-defined networking: A comprehensive survey. **Proceedings of the IEEE**, Ieee, v. 103, n. 1, p. 14–76, 2015.

LIN, Y.; SHEN, H. Cloudfog: Leveraging fog to extend cloud gaming for thin-client mmog with high quality of service. **IEEE Transactions on Parallel and Distributed Systems**, v. 28, n. 2, p. 431–445, Feb 2017. ISSN 1045-9219.

LIU, F. et al. Nist cloud computing reference architecture. **NIST special publication**, v. 500, n. 2011, p. 292, 2011.

LIU, F. et al. **NIST Cloud Computing Reference Architecture: Recommendations of the National Institute of Standards and Technology (Special Publication 500-292)**. USA: CreateSpace Independent Publishing Platform, 2012. ISBN 1478168021, 9781478168027.

MAHMUD, R.; KOTAGIRI, R.; BUYYA, R. Fog computing: A taxonomy, survey and future directions. In: **Internet of everything**. [S.l.]: Springer, 2018. p. 103–130.

MANSOUR, I. et al. Interoperability in the heterogeneous cloud environment: A survey of recent user-centric approaches. In: **Proceedings of the International Conference on Internet of Things and Cloud Computing**. New York, NY, USA: ACM, 2016. (ICC '16), p. 62:1–62:7. ISBN 978-1-4503-4063-2. Disponível em: <http://doi.acm.org/10.1145/2896387.2896447>.

MASHTIZADEH, A. J. et al. Xvmotion: Unified virtual machine migration over long distance. In: **2014 USENIX Annual Technical Conference (USENIX ATC 14)**. Philadelphia, PA: USENIX Association, 2014. p. 97–108. ISBN 978-1-931971-10-2. Disponível em: <https://www.usenix.org/conference/atc14/technical-sessions/presentation/mashtizadeh>.

MEDINA, V.; GARCÍA, J. M. A Survey of Migration Mechanisms of Virtual Machines. **ACM Comput. Surv.**, ACM, New York, NY, USA, v. 46, n. 3, p. 30:1–30:33, jan 2014. ISSN 0360-0300. Disponível em: <http://doi.acm.org/10.1145/2492705>.

MELL, P. M.; GRANCE, T. **SP 800-145. The NIST Definition of Cloud Computing**. Gaithersburg, MD, United States, 2011.

MERKEL, D. Docker: Lightweight linux containers for consistent development and deployment. **Linux J.**, Belltown Media, Houston, TX, v. 2014, n. 239, mar 2014. ISSN 1075-3583. Disponível em: <http://dl.acm.org/citation.cfm?id=2600239.2600241>.

MICROSOFT. **Microsoft Azure**. 2017. <https://azure.microsoft.com/pt-br/>. Acessado em: 14-10-2017.

NAIR, S. K. et al. Towards secure cloud bursting, brokerage and aggregation. In: IEEE. **Web services (ecows), 2010 ieee 8th european conference on**. [S.l.], 2010. p. 189–196.

OPENSTACK. **OpenStack**. 2017. <https://www.openstack.org/>. Acessado em: 10-11-2017.

RACKSPACE. **Rackspace**. 2017. <https://www.rackspace.com/>. Acessado em: 14-10-2017.

RAJAGOPALAN, S. et al. SecondSite: disaster tolerance as a service. **ACM SIGPLAN Notices**, ACM, 47, n. 7, p. 97–108, 2012.

RAUGUST, A. et al. Towards secure cloud bursting, brokerage and aggregation. In: IEEE. **32nd IEEE International Conference on Advanced Information Networking and Applications (IEEE AINA-2018)**. [S.l.], 2018.

RDO. **Difference between Floating IP and private IP**. 2017. <https://www.rdoproject.org/networking/difference-between-floating-ip-and-private-ip/>. Acessado em: 01-11-2017.

RISTOV, S.; KOSTOSKA, M.; GUSEV, M. P-tosca portability demo case. In: IEEE. **Cloud Networking (CloudNet), 2014 IEEE 3rd International Conference on**. [S.l.], 2014. p. 261–263.

RODRIGUEZ, M. A.; BUYYA, R. A taxonomy and survey on scheduling algorithms for scientific workflows in iaas cloud computing environments. **Concurrency and Computation: Practice and Experience**, Wiley Online Library, v. 29, n. 8, 2017.

ROSENBLUM, M.; GARFINKEL, T. Virtual machine monitors: Current technology and future trends. **Computer**, IEEE, v. 38, n. 5, p. 39–47, 2005.

RUCK, D. et al. Eavira: Energy-aware virtual infrastructure reallocation algorithm. In: **2017 VII Brazilian Symposium on Computing Systems Engineering (SBESC)**. [S.l.: s.n.], 2017.

SCHEEPERS, M. J. Virtualization and containerization of application infrastructure: A comparison. In: **21st Twente Student Conference on IT**. [S.l.: s.n.], 2014. p. 1–7.

SEO, K.-T. et al. Performance comparison analysis of linux container and virtual machine for building cloud. **Advanced Science and Technology Letters**, v. 66, p. 105–111, 2014.

SMITH, J. E.; NAIR, R. The architecture of virtual machines. **Computer**, IEEE, v. 38, n. 5, p. 32–38, 2005.

SOKOL, A. W.; HOGAN, M. D. Nist cloud computing standards roadmap. **Special Publication (NIST SP)-500-291r2**, 2013.

STANDARD, O. **Topology and orchestration specification for cloud applications version 1.0**. [S.l.]: Tech. rep., OASIS Standard (November 2013). url:{http://docs. oasis-open. org/tosca/TOSCA/v1. 0/os/TOSCA-v1. 0-os. html}, 2013.

TANEMBAUM, A. S.; WETHERRAL, D. J. **Computer Networks 5th**. Upper Saddle River, NJ, USA: Prentice Hall Press, 2010.

TANG, Z. et al. Migration modeling and learning algorithms for containers in fog computing. **IEEE Transactions on Services Computing**, p. 1–1, 2018. ISSN 1939-1374.

TOOSI, A. N.; CALHEIROS, R. N.; BUYYA, R. Interconnected cloud computing environments: Challenges, taxonomy, and survey. **ACM Comput. Surv.**, ACM, New York, NY, USA, v. 47, n. 1, p. 7:1–7:47, maio 2014. ISSN 0360-0300. Disponível em: <http://doi.acm.org/10.1145/2593512>.

TRAN, V. et al. Application migration to cloud: a taxonomy of critical factors. In: ACM. **Proceedings of the 2nd international workshop on software engineering for cloud computing**. [S.l.], 2011. p. 22–28.

TSAKALOZOS, K. et al. Live vm migration under time-constraints in share-nothing iaas-clouds. **IEEE Transactions on Parallel and Distributed Systems**, IEEE, v. 28, n. 8, p. 2285–2298, 2017.

WOOD, T. et al. CloudNet: dynamic pooling of cloud resources by live WAN migration of virtual machines. In: ACM. **ACM Sigplan Notices**. [S.l.], 2011. 46, n. 7, p. 121–132.

XAVIER, M. G. et al. Performance evaluation of container-based virtualization for high performance computing environments. In: IEEE. **Parallel, Distributed and Network-Based Processing (PDP), 2013 21st Euromicro International Conference on**. [S.l.], 2013. p. 233–240.

XU, F. et al. Managing performance overhead of virtual machines in cloud computing: A survey, state of the art, and future directions. **Proceedings of the IEEE**, IEEE, v. 102, n. 1, p. 11–31, 2014.

ZHANG, F.; FU, X.; YAHYAPOUR, R. CBase: A New Paradigm for Fast Virtual Machine Migration across Data Centers. In: IEEE Press. **Proceedings of the 17th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing**. [S.l.], 2017. p. 284–293.

ZHANG, J.; LU, X.; PANDA, D. K. Performance Characterization of Hypervisor-and Container-Based Virtualization for HPC on SR-IOV Enabled InfiniBand Clusters. In: IEEE. **Parallel and Distributed Processing Symposium Workshops, 2016 IEEE International**. [S.l.], 2016. p. 1777–1784.

ZHANG, Q.; CHENG, L.; BOUTABA, R. Cloud computing: state-of-the-art and research challenges. **Journal of Internet Services and Applications**, v. 1, n. 1, p. 7–18, May 2010. ISSN 1869-0238. Disponível em: <https://doi.org/10.1007/s13174-010-0007-6>.

ZHANG, Z.; WU, C.; CHEUNG, D. W. A survey on cloud interoperability: Taxonomies, standards, and practice. **SIGMETRICS Perform. Eval. Rev.**, ACM, New York, NY, USA, v. 40, n. 4, p. 13–22, abr. 2013. ISSN 0163-5999. Disponível em: <http://doi.acm.org/10.1145/2479942.2479945>.

ZHAO, J.-F.; ZHOU, J.-T. Strategies and methods for cloud migration. **international Journal of Automation and Computing**, Springer, 11, n. 2, p. 143–152, 2014.

ZHAO, Y. et al. Virtual network migration on the geni wide-area sdn-enabled infrastructure. **arXiv preprint arXiv:1701.01702**, 2017.