Virtual Infrastructures on the Move: Containers and Virtual Network Migration

Euclides Cardoso Jr.*, Charles C. Miers*, Maurício A. Pillon*, Fernando F. Redígolo+, Guilherme P. Koslovski*
*Graduate Program in Applied Computing – Santa Catarina State University – Joinville, SC – Brazil
+Laboratory of Computer Networks and Architecture – University of São Paulo, São Paulo, SP, Brazil
euclides.cj@edu.udesc.br, fernando@larc.usp.br, {mauricio.pillon, charles.miers, guilherme.koslovski}@udesc.br

***Abstract*—Cloud computing model enables users to view computational resources as unlimited, which can be dynamically reserved. In addition, users are charged according to a contracted configuration and effective reservation time. Thus, it became possible to dynamically provision Virtual Infrastructures (VIs) composed of virtual machines, containers, and virtualized network resources. Due to the flexibility in resource provisioning and configuration, VI is being increasingly adopted for hosting 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 ease the migration of virtualized resources between distinct providers, inducing in vendor lock-in. A wide range of cloud providers, APIs, models, and management tools further limit the development of VI migration solutions between providers, characterizing an open challenge. We propose a mechanism to perform VI migration among providers, based on an architecture agnostic to source and destination providers, as well as VI-hosted applications. In addition, our prototype based on Docker containers and OpenStack clouds was developed both to validate the proposed mechanism and to serve as a reference implementation.**

***Index Terms*—Cloud Computing, Virtual Infrastructure, Migration**

I. Introdução

O paradigma de computação em nuvem provocou diversas mudanças na maneira como os recursos de armazenamento, processamento e comunicação são provisionados e gerenciados. Em especial, 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 [1]. Exemplificando, provedores públicos renomados (*e.g.,* Amazon e Google) oferecem IVs com a terminologia de *Virtual Private Cloud* (VPC).

Com a popularização da computação em nuvem ocorreu uma proliferação de provedores IaaS [2]. Internamente, cada provedor realiza a orquestração de sua infraestrutura com ferramentas específicas, 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 clientes acreditam que criam IVs de maneira independente, entretanto as otimizações realizadas pelos provedores criam amarrações nos recursos virtualizados [3].

Eventualmente, clientes de provedores IaaS podem desejar a migração e realocação de IVs para outros provedores ou regiões geográficas. Independentemente das questões burocráticas dos provedores envolvidos, uma migração possui diversos aspectos técnicos desafiadores que devem ser tratados, cada um de modo diferenciado [2]. Por exemplo, as políticas de cobrança podem variar entre provedores e entre regiões, bem como a qualidade do serviço recebido, essencial para o desempenho das aplicações hospedadas nas IVs. Sobretudo, as dificuldades enfrentadas são decorrentes da falta de padronização e até mesmo por problemas legais. Geralmente, abordagens padronizadas são centradas nos provedores e mesmo que padrões / abordagens centradas nos clientes surjam, melhorando a interoperabilidade e a portabilidade, existe a relutância dos grandes provedores em adotar estes padrões. A impossibilidade ou dificuldade de interoperabilidade, por sua vez, pode proteger os interesses individuais de provedores, limitando a migração de clientes entre provedores comercialmente concorrentes [3, 4].

Soluções e tecnologias que auxiliem na realização desta migração encontram-se em um estado inicial de desenvolvimento [5, 6, 7]. Migrar uma IV consiste na transferência, neste caso entre provedores, da aplicação do cliente e reestabelecimento das conexões de rede subjacentes [8, 9, 10]. De forma transparente ao cliente, a migração de redes deve garantir: *(i)* o reestabelecimento das conexões na rede privada interna, assim, após a migração, as aplicações retomam suas execuções sem a intervenção do cliente; e *(ii)* a permanência das comunicações entre a IV migrada e a rede externa. Considerando que endereços *Internet Protocol*s (IPs) públicos estão associados a provedores, o uso do IP externo no provedor de origem não é possível no provedor de destino. Portanto, para manter as conexões em funcionamento, faz-se necessário o redirecionamento temporário do tráfego de rede do IP externo de origem para o novo IP externo de destino.

Os desafios e limitações descritos não estão em conformidade com os requisitos de portabilidade e interoperabilidade propostos pelo NIST [11]. Na perspectiva do cliente, a portabilidade pode ser vista como a capacidade de mover dados e aplicações entre múltiplos provedores, garantindo, durante esta operação, o mínimo de interrupção dos serviços do cliente. Interoperabilidade, por sua vez, refere-se a capacidade de acesso a dados e serviços, por parte do cliente, em provedores distintos através de uma interface unificada [11].

O presente trabalho propõe uma solução para migração de IV entre provedores de nuvens distintos, privados ou públicos. A proposta caracteriza-se como um corretor de nuvem (*Cloud Broker*). As principais contribuições do trabalho consistem na: *(i)* concepção da arquitetura do corretor de nuvem; e *(ii)* prototipação desta arquitetura em um ambiente real, baseado em OpenStack. O protótipo encontra-se operacional e confirmou a aplicabilidade da solução proposta migrando IVs entre provedores distintos.

O restante do trabalho está organizado da seguinte forma: A Seção II apresenta a motivação e a definição do problema. A arquitetura do corretor é proposta na Seção III enquanto a análise experimental, detalhando um protótipo baseado em OpenStack, é discutida na Seção IV. Os trabalhos relacionados são revisados na Seção V e as considerações finais são apresentadas na Seção VI.

II. Motivação e Definição do Problema

Uma Infraestrutura Virtual (IV) pode ser definida como um conjunto de recursos virtuais conectados por serviços de comunicação. Tais serviços, utilizados para interconectar Máquinas Virtuais (MVs) e contêineres, oferecem roteadores, comutadores e canais de comunicação, todos virtualizados [12, 13, 14]. Com a virtualização destes componentes de rede, surge a possibilidade de provisionamento de acordo com as necessidades das aplicações hospedadas na IV.

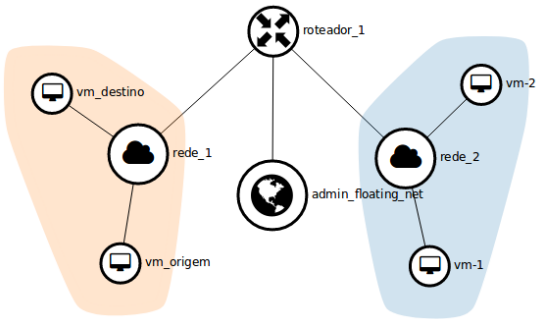

Figura 1. Exemplo de uma Infraestrutura Virtual.

Na Figura 1 é possível visualizar uma representação gráfica de uma IV. Diversas MVs foram alocadas pelo cliente, 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 cliente 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 [15].

Dessa forma, tem-se que uma IV é composta por recursos virtualizados, 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 [16].

A. Migração de Recursos entre Provedores IaaS

Migrar recursos virtualizados entre provedores IaaS é um desafio em aberto na literatura especializada (Seção V). Sobretudo, existem diferentes unidades de migração em nuvens IaaS, sendo elas: contêineres, MVs e IVs. Os pontos positivos e negativos da migração de cada unidade são elencados.

1) Migração de Máquinas Virtuais (MVs): Através da técnica de virtualização de MVs, os provedores de nuvens fornecem para os clientes a ilusão de um hospedeiro completamente privado. Um dos principais benefícios da migração de MVs é o balanceamento de carga em *data centers* [17]. Ainda, 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 [18, 19]. 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 [20, 21]. Para realizar a migração entre nuvens de pequeno porte, um armazenamento compartilhado pode ser utilizado como solução. Entretanto, quando se trata de nuvens de larga escala, a abordagem pode ser ineficiente devido ao elevado volume de dados trafegados.

Existem propostas para melhorar o desempenho da migração de MVs [20, 22, 23], que exploram mecanismos como a pré-cópia [20] e pós-cópia [22] dos dados em memória. Todavia, esses mecanismos são focados na migração de MVs em *data centers* privados [23] e não na migração de MVs através de diferentes provedores IaaS, ou seja, sobre a Internet. Além disso, para realizar a migração de MVs, é necessário ter acesso administrativo ao hipervisor, restringindo assim o processo de migração somente para o gerenciador da nuvem. Dessa forma o cliente ainda está suscetível ao *vendor lock-in*.

2) Migração de Contêineres: Diferente da técnica de MVs, contêineres virtualizados compartilham o mesmo *kernel* do Sistema Operacional (SO). Assim, migrar contêineres consiste em desacoplar processos do SO e disponibilizá-los, de forma integral, em um novo hospedeiro [16]. Um dos pontos positivos da migração de contêineres é que estes possuem todos os componentes necessários ao seu funcionamento, tais como arquivos e bibliotecas. Além disso, na migração de contêineres não é necessário ter acesso ao hipervisor, possibilitando assim que o próprio cliente realize a migração sem a necessidade de contato com o provedor de serviço. Devido às características dos contêineres, o provedor de nuvem encontra dificuldades em criar ancoragens que prendam o cliente ao seu serviço, tornando assim mais fácil a migração e prevenindo o *vendor lock-in*. Em contrapartida, a virtualização baseada em contêineres ainda é um processo novo e em desenvolvimento nos provedores de nuvem. Ou seja, nem todas as ferramentas de gerenciamento de contêineres possuem suporte para a migração.

3) 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 rede virtualizada (enlaces, roteadores e *switches*) são conjuntamente migrados. É importante ressaltar que no cenário de computação em nuvem, a camada de virtualização pode ser MVs, contêineres sobre *hardware*, contêineres sobre SO ou contêineres dentro de MVs.

Ao mover uma IV, as configurações de rede e da aplicação hospedada são mantidas inalteradas. Para tanto, a Rede Virtual (RV) deve ser pré-configurada no provedor de serviço de destino de maneira que a configuração da mesma não seja alterada, por exemplo, endereços de IP privados. Além dos desafios de migração de contêineres e MVs, a migração da rede ainda é um desafio em aberto [8, 9, 10].

B. Discussão sobre Migração de Recursos Virtualizados

A corretagem em nuvem, foco deste trabalho, apoia-se na migração de IVs compostas por contêineres alocados em MVs. Entre os fatores favoráveis a esta abordagem estão: *(i)* a liberdade de gerência dada ao cliente por esta tecnologia (independência do provedor); e *(ii)* utilização de contêineres em serviços oferecidos atualmente pelos provedores de nuvens IaaS. A exclusão de uso de contêineres sobre SO e sobre *hardware* neste trabalho se dá pelo mesmo fator: a necessidade, por parte do cliente, de privilégios para o gerenciamento do contêiner tornando-o dependente do provedor. Portanto, caso o cliente possua acesso ao gerenciamento de seus contêineres, os benefícios da corretagem em nuvem disponibilizados por este trabalho podem ser estendidos a estas duas outras formas de conteinerização.

No contexto de nuvem computacional IaaS, a migração da aplicação não é suficiente para se ter sucesso pois, para se garantir a disponibilidade e a integridade do serviço, tem-se ainda que efetuar a migração da rede. Na rede interna, a manutenção do comportamento da aplicação pode ser garantida com a replicação do modelo de topologia e projeto de redes do provedor de origem para o provedor de destino. Considerando-se que o gerenciador de recursos do provedor de origem é compatível com o gerenciador de destino, a replicação do projeto de rede virtual no destino restringe-se à manutenção dos endereços IP, máscaras, configuração dos roteadores e *switches*, entre outros.

Por outro lado, o IP da rede externa da IV, após migrada ao provedor de destino, não pode ser mantido, pois o IP público associado a esta rede é propriedade do provedor de origem. Entretanto, com o intuito de manter a visibilidade e a disponibilidade do serviço, faz-se necessário a manutenção das conexões ativas no momento de início da migração. Uma solução para manter as comunicações abertas das IVs quando ela estava no provedor de origem após a migração ao provedor de destino é o redirecionamento do tráfego de rede proveniente da Internet. Embora não trivial, trabalhos de pesquisas na área relatam algumas soluções para este problema, tais como as apresentadas em [24] e [25], 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, encontra-se fora do escopo de pesquisa deste trabalho.

Finalmente, a migração de uma IV por um corretor de nuvem é transparente para o cliente, cabendo à ferramenta de corretagem a avaliação e execução das ações de migração, tanto no provedor de origem quanto no provedor de destino. Ao cliente resta a confirmação das etapas da migração, por meio de *Application Programming Interfaces* (APIs) que acessam os ambientes de ambos os provedores de nuvem.

III. Arquitetura para Migração de IVs

O corretor de nuvem proposto neste trabalho caracteriza-se por uma arquitetura constituída de três componentes: o **Gerenciador**, o **GeraVI** e o **RecriaVI**. Os componentes desta arquitetura e suas relações encontram-se representados na Figura 2. O **Gerenciador** é responsável pela cópia das aplicações, na nuvem de origem, e pelo transporte destas aplicações para a nuvem de destino. O MigraVI é independente do provedor, podendo executar na nuvem de origem, de destino ou em outro sítio externo. Para tal, ele exige somente as credenciais de acesso do cliente na nuvem de origem e de destino. O **GeraVI**, localizado na nuvem de origem, é responsável pela obtenção das informações atualizadas da RV e pelo *checkpoint* dos contêineres na origem. Na nuvem de destino, encontra-se o **RecriaVI**, que tem como função principal a recriação da RV e dos contêineres a partir do último *checkpoint* gerado pelo **GeraVI**.

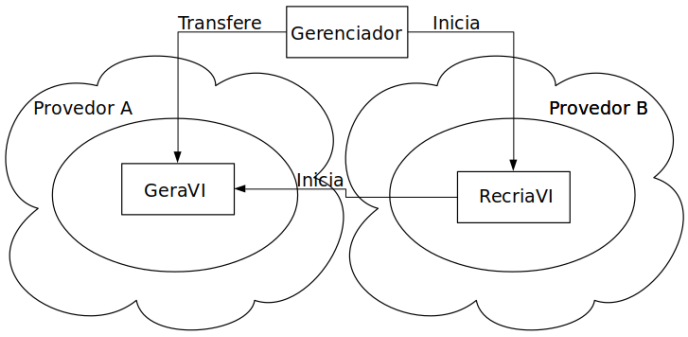

Figura 2. Arquitetura do MigraVI.

O corretor de nuvem MigraVI é o intermediador na realização da migração entre os diferentes provedores. Para realizar o processo de migração, a arquitetura necessita dos dados de autenticação e autorização das nuvens de origem e destino. Consequentemente, o cliente deverá ter total confiança no mecanismo, conforme é comum na configuração de corretores de nuvens já existentes [2, 5, 6, 26]. É importante ressaltar que MigraVI não cria nem gerencia as contas de clientes nos provedores de nuvem de origem e destino. Tal gerenciamento dos dados de autenticação e autorização

fica sob responsabilidade do cliente. Similarmente, questões relacionadas ao *Service Level Agreement* (SLA) e aos dados de cobrança estão fora do escopo deste trabalho.

O 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, o mecanismo proposto por este trabalho não considera no presente momento os cenários de *cloud bursting* [27].

A. Cenário de Migração de IVs

O cenário de execução é composto por duas nuvens e pelo corretor de nuvem MIGRAVI, responsável por realizar a migração da IV da nuvem de origem, localizada no *Provedor A*, para a nuvem de destino, localizada no *Provedor B*. 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 3 representa a composição da IV: um conjunto de hospedeiros (*hospedeiro1, hospedeiro2* e *hospedeiro3*) possuem alocados os contêineres ($c1$, $c2$, $c3$, $c4$, $c5$ e $c6$). Os contêineres são interconectados por enlaces virtualizados.

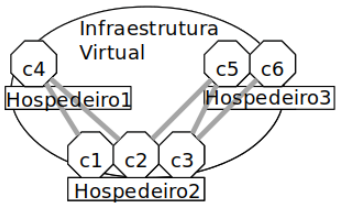

Figura 3. Exemplo de IV composta por contêineres hospedados em MVs conectadas por uma RV.

Conforme é possível observar na Figura 4, após MIGRAVI obter acesso às nuvens de origem e de destino, é realizada uma leitura da nuvem de origem executado pelo módulo **GeraVI**. Essa leitura tem como objetivo levantar as informações da rede e dos hospedeiros da nuvem de origem, tais como número de roteadores, *switches*, canais de comunicação, endereços de IP, tipo de hospedeiros e nomes das imagens. Na nuvem de destino, o **RecriaVI** recebe estas informações e inicia o preparo do ambiente de destino, instanciando as MVs vinculadas à IV a ser migrada. Assim que o ambiente de destino estiver instanciado (Figura 5), o **RecriaVI** passa a configuração da RV, recriando o projeto de rede de acordo com a topologia e endereços IPs descritos na origem. Finalmente, a última etapa do **RecriaVI** é a migração dos contêineres, seguindo o mapeamento contêiner/MV informado pelo **GeraVI** obtido na nuvem de origem. Neste ponto, caso os contêineres possuam volumes associados, estes também são migrados para a nuvem de destino. Assim que a migração dos contêineres e volumes forem finalizados, ou seja, transferidos e restaurados no destino(Figura 6), a migração da IV é finalizada.

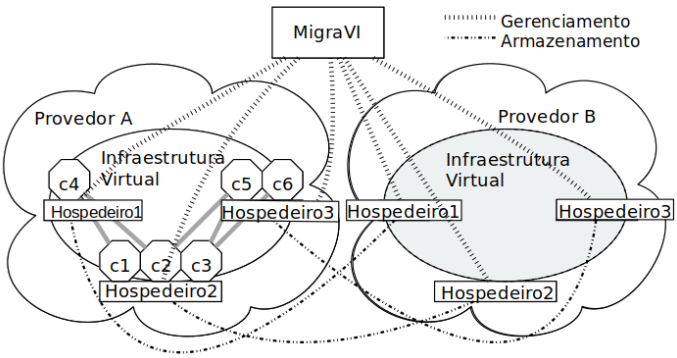

Figura 4. Preparação do cenário de migração.

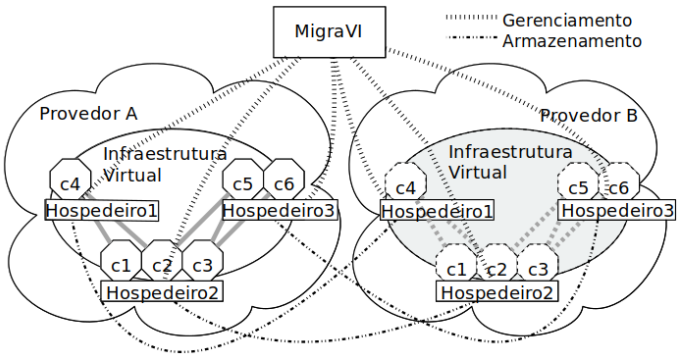

Figura 5. Migração de contêineres e volumes.

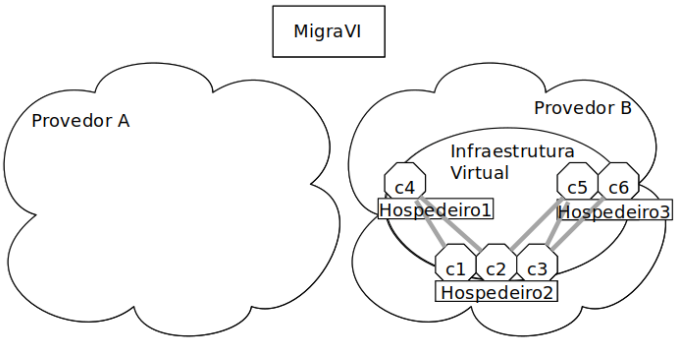

Figura 6. Finalização do processo de migração da IV.

B. Algoritmos de Gerenciamento (MIGRAVI)

O pseudocódigo do **Gerenciador** é apresentado no Algoritmo 1. 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.

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)
```

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. 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 arquivo 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.

C. Sequência de Eventos para Migração de uma IV

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 7. 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.

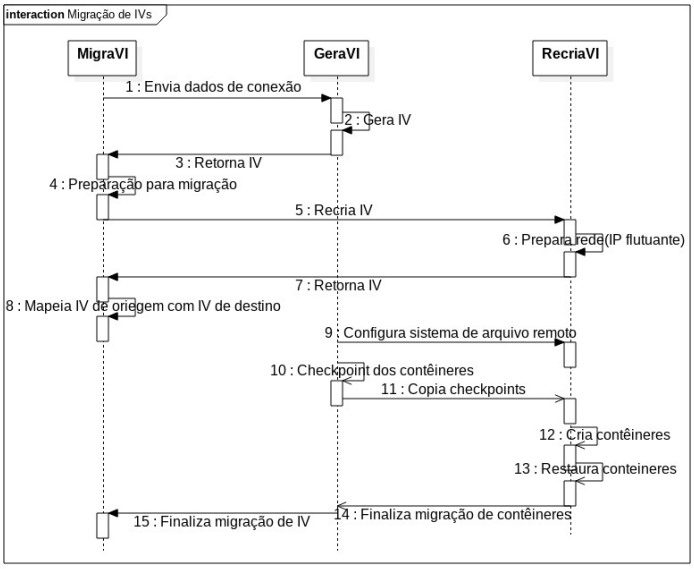

Figura 7. Diagrama de sequência para realizar a migração de IVs entre provedores distintos.

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 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).

IV. Análise Experimental

A presente seção resume a implementação e análise experimental de um protótipo baseado em OpenStack, denominado MigraVI-OpenStack. 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 III-C.

A. MigraVI-OpenStack: *Ferramentas e Tecnologias*

Para a implementação do protótipo MigraVI-OpenStack, diversas ferramentas e tecnologias são utilizadas:

1) OpenStack: O OpenStack é um conjunto de projetos modelado para gerenciar nuvens computacionais, sejam elas públicas ou privadas [28]. Dessa forma, pode-se dizer que o OpenStack é responsável por gerenciar e controlar uma grande quantidade de recursos, sendo eles computacionais, de armazenamento e de rede. Os recursos podem ser provisionados e acessados pelos clientes através de um painel de controle gráfico, bem como por intermédio de APIs.

2) Flame: No OpenStack, a orquestração de IVs pode ser realizada através do projeto HEAT, o qual descreve a IV e seus recursos através de modelos denominados *Heat Orchestration Template* (HOT). Uma vez a IV criada, a recuperação automática do modelo HOT desta IV é possível através do Flame. Este reconstrói o modelo HOT consultando o OpenStack diretamente pela API. Dessa maneira, com a utilização do projeto HEAT e do Flame, o MigraVI-OpenStack está habilitado a recriar a IV na nuvem de destino, mantendo as mesmas características do projeto e topologia de redes da nuvem de origem.

3) Docker e Convoy: O Docker é uma ferramenta para a entrega e gerenciamento de contêineres em ambientes locais e em nuvens híbridas [29]. Ferramentas de orquestração de contêineres como Kubernetes, não possuem suporte nativo para migração de contêineres juntamente com os dados. Dessa forma, faz-se necessário a utilização de um volume o qual deve ser acoplado ao contêiner, caso o cliente deseje persistir os dados. Assim, mesmo que o contêiner seja destruído, os dados permanecem disponíveis nos volumes e podem ser acessados por outros contêineres. Entretanto, os volumes do Docker são atrelados ao hospedeiro no qual foram inicialmente criados, ou seja, caso o contêiner seja migrado para outro hospedeiro os volumes devem ser disponibilizados no destino.

Para lidar com esse problema existem *drivers* para gerenciar os volumes do Docker e assim obter a portabilidade. O MigraVI-OpenStack faz uso do Convoy, 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. Durante o processo de migração, o MigraVI-OpenStack precisa que os dados dos *snapshots* gerados pelo **GeraVI**, na nuvem de origem, estejam acessível a **RecriaVI**, na nuvem de destino. No MigraVI-OpenStack, a disponibilização dos dados entre a nuvem de origem e de destino foi feita via *Network File System* (NFS), pois o Convoy já possui suporte nativo a essa tecnologia

B. Cenários de Migração

Foram definidos dois cenários que para proporcionar uma comparação entre a complexidade de migrar uma IV e migrar apenas um contêiner. O primeiro cenário consiste em realizar a migração do contêiner A, originalmente instanciado na MV_1, hospedada na nuvem origem, para a MV_2, hospedada na nuvem de destino. Este cenário tem como objetivo quantificar o tempo mínimo de migração de um contêiner, seguindo todas as etapas sequencialmente. O cenário de migração baseia-se em uma *Wide Area Network* (WAN). Com as mesmas nuvens de origem e destino, o segundo cenário efetua a migração da IV completa. Neste cenário, a IV a ser migrada é constituída por um *switch*, duas MVs (MV_1 e MV_2) e 20 contêineres (10 em cada MV).

C. Ambiente de Experimentação

A infraestrutura de pesquisa em computação em nuvem utilizada para a realização destes experimentos foi o CloudLab, constituída, atualmente, por três sítios distribuídos geograficamente nos Estados Unidos da América [30]. As nuvens escolhidas para realização dos testes foram: a localizada no estado de Utah, como origem, e 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 WAN. A latência observada

[1]Código disponível em: https://bitbucket.org/micovick/migracao-de-ivs.git.

entre os dois sítios foi de aproximadamente $9187ms$ com um RTT médio de 51.853. Além disso, o protótipo MIGRAVI-OPENSTACK é baseado em um conjunto de ferramentas, linguagens e configurações, especificamente: Docker versão $18.03.0$-*Community Edition*-ce; Convoy versão $0.5.0$; Python versão $2.7.6$; MV que executam Ubuntu 14.04 Trusty, com uma vCPU e 2 GB de RAM; e OpenStack na versão Pike. Adicionalmente, para que MIGRAVI-OPENSTACK tenha acesso as MVs e assim consiga realizar a migração dos contêineres, ele apoia-se no *Secure Shell* (SSH), cujo as chaves de acesso às IVs devem ser fornecidas pelo cliente.

D. Métricas para Análise

O protótipo do MIGRAVI-OPENSTACK foi avaliado sob a ótica de duas métricas principais: (i) tempo total de migração de uma IV; e (ii) 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.

E. Discussão dos Resultados

Em todos os cenários, a amostra utilizada foi de tamanho 10 e os valores descritos nos gráficos são os resultados da média destas execuções. No primeiro cenário constatou-se que a média das execuções da migração dos contêineres ficou em $3,41$ segundos. A execução desta etapa restringe-se a transferência dos contêineres da origem para o destino, sem adequações na configuração das IVs. As duas outras etapas, realizadas pelo **GeraVI** e **RecriaVI**, são analisadas individualmente, nas Figuras 8 e 9, 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.

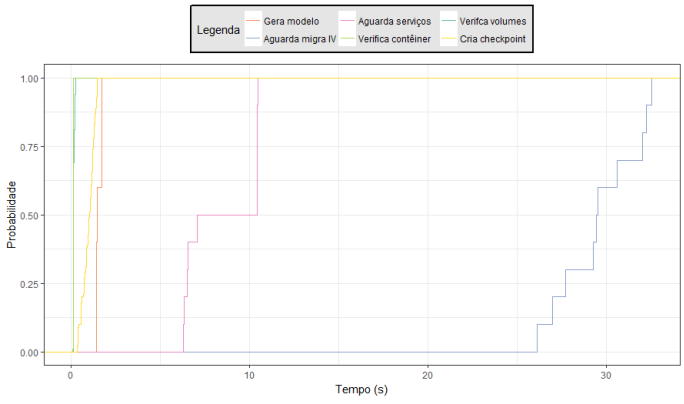

Figura 8. Tempo de execução das tarefas efetuadas no provedor de origem pelo módulo **GeraVI**.

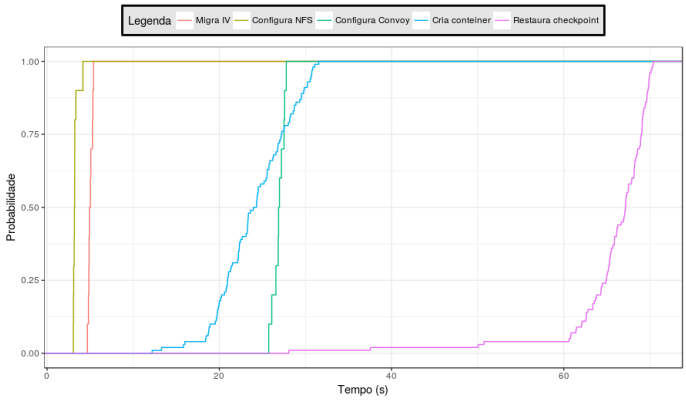

Figura 9. Tempo de execução das tarefas efetuadas no provedor de destino pelo módulo **RecriaVI**.

Como resultado da análise da Figura 8, 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, espera a estabilização dos serviços na nuvem de destino, *e.g.*, o SSH.

A Figura 9 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. 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 a 60 segundos. O maior grau de probabilidade indica o tempo desta atividade mais próximo a 60 segundos. 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 60 segundos. Dessa forma, o tempo necessário para manter dois provedores ativos durante a migração varia de acordo com a quantidade de dados que devem ser sincronizados entre eles. 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.

As análises das Figuras 8 e 9 basearam-se na probalidade de ocorrência de cada atividade. Os próximos resultados focaram na quantificação do uso de rede e do tempo associado a cada atividade. Na Figura 10, tem-se o tempo consumido por cada atividade durante a migração de uma IV. Importante ressaltar que na Figura 10 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, permitem destacar dois comportamentos distintos: (i) as atividades *Restaura contêiner*, *Cria Contêiner*, *Configura Convoy*, *Aguarda migra IV* são as maiores consumidoras de tempo, (ii) as demais atividades possuem valores inferiores a $5,1\%$.

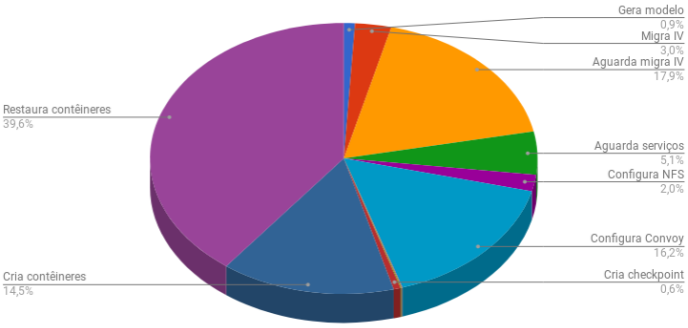

Figura 10. Percentual dos tempos de execuções das tarefas durante a migração de uma IV.

A Figura 11 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. Dessa forma, é possível observar o quanto cada atividade pode impactar no uso da rede durante a migração. Como é possível observar na Figura 11, a atividade que mais consume rede é a atividade de restauração dos contêineres (*restaura checkpoint*). 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). Este consumo é decorrente da transferência dos *checkpoint*, via NFS.

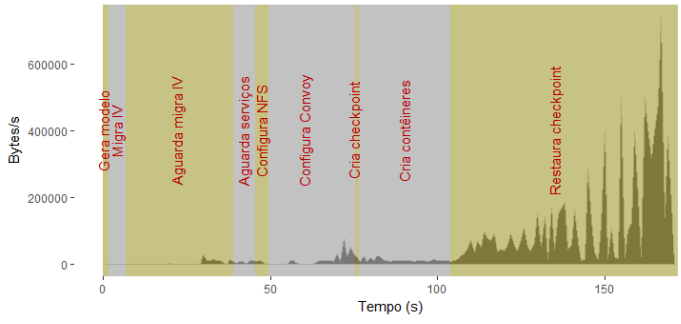

Figura 11. Utilização da rede durante a migração de uma IV.

V. Trabalhos Relacionados

Para discussão dos trabalhos relacionados, foi selecionado ao menos um trabalho que aborde a migração de cada unidade de migração definida na Seção II-A. Iniciando pela migração de aplicações em computação em nuvem, o trabalho apresentado por [31] propõe um algoritmo de orquestração baseado em *live migration* de aplicações em ambientes de nuvens computacionais, 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. O processo de migração de aplicações também é abordado em [32]. Neste artigo os autores descrevem o P-TOSCA, um modelo utilizado especificamente em nuvens *Platform as a Service* (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 *Topology and Orchestration Specification for Cloud Applications* (TOSCA) e que são necessários para implantar uma aplicação [33]. 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. 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 [34]. 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 *Software as a Service* (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 específica 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 [35] 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 [36]. 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 Qualidade de Serviço (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 [23] 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 [37] é desenvolvido um mecanismo para a migração de IVs utilizando a tecnologia de *Software Defined Networks* (SDN). 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 nos *switches*.

Como pode-se observar, a migração em computação em nuvem é um tema atual e ainda em aberto. Entretanto, nota-se pouca pesquisa para tratar sobre migração de IVs. Nota-se que o problema é realizar a migração da rede juntamente com as aplicações. Logo, o presente trabalho introduziu uma arquitetura para um corretor permitindo que o cliente realize a migração de sua IV sem depender dos provedores de origem e destino.

VI. Considerações & Trabalhos Futuros

Em um ambiente de computação em nuvem heterogênico, do ponto de vista do cliente, a migração e realocação de IV entre provedores é uma tarefa complexa. Neste contexto, este trabalho propôs o corretor de nuvem MigraVI, que permite a migração de dados e redes virtualizadas de clientes de forma agnóstica aos provedores. O corretor de nuvem MigraVI executa com privilégios do cliente da nuvem e é constituído por três componentes: **Gerenciador, GeraIV** e **RecriaIV**. Um protótipo operacional em OpenStack deste corretor é utilizado para se obter resultados preliminares em ambiente real, permitindo discutir a solução de migração de IV proposta neste trabalho.

Os resultados obtidos permitem concluir que a atividade que gera maior impacto no tempo total de migração é a restauração dos contêineres. Com uma análise fina do consumo de rede, observou-se que o atraso na execução da atividade de restauração do contêineres é decorrente do intenso uso de rede. Aproximadamente 40% do tempo total de migração é gasto na transferência de arquivos entre nuvem de origem e destino.

Como trabalho futuro a curto prazo, tem-se a definição e testes de outros cenários que variem número de contêineres, de MVs e de sub-redes virtuais. A médio prazo, busca-se otimizar algumas etapas e, principalmente, testar outras tecnologias de gerenciamento de contêineres e de transferência de arquivos entre os provedores, certamente um gargalo de desempenho na migração de IV.

Agradecimentos

Os autores agradecem o apoio do Laboratório de Processamento Paralelo e Distribuído (LabP2D) no Centro de Ciências Tecnológicas (CCT) da Universidade do Estado de Santa Catarina (UDESC). Este trabalho foi realizado utilizando recursos fornecidos pela FAPESC.

Referências

[1] F. Liu *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.

[2] Z. Zhang *et al.*, "A survey on cloud interoperability: Taxonomies, standards, and practice," *SIGMETRICS Perform. Eval. Rev.*, vol. 40, no. 4, pp. 13–22, Apr. 2013.

[3] K. Kaur *et al.*, "Interoperability and portability approaches in inter-connected clouds: A review," *ACM Comput. Surv.*, vol. 50, no. 4, pp. 49:1–49:40, Oct. 2017.

[4] P. Hofmann and D. Woods, "Cloud computing: The limits of public clouds for business applications," *IEEE Internet Computing*, vol. 14, no. 6, pp. 90–93, Nov 2010.

[5] N. Grozev and R. Buyya, "Inter-cloud architectures and application brokering: taxonomy and survey," *Software: Practice and Experience*, vol. 44, no. 3, pp. 369–390, 2014.

[6] A. N. Toosi *et al.*, "Interconnected cloud computing environments: Challenges, taxonomy, and survey," *ACM Comput. Surv.*, vol. 47, no. 1, pp. 7:1–7:47, May 2014.

[7] A. López García *et al.*, "Standards for enabling heterogeneous iaas cloud federations," *Comput. Stand. Interfaces*, vol. 47, no. C, pp. 19–23, Aug. 2016.

[8] J.-F. Zhao and J.-T. Zhou, "Strategies and methods for cloud migration," *international Journal of Automation and Computing*, vol. 11, no. 2, pp. 143–152, 2014.

[9] P. Jamshidi *et al.*, "Cloud migration research: a systematic review," *IEEE Transactions on Cloud Computing*, vol. 1, no. 2, pp. 142–157, 2013.

[10] V. Medina and J. M. García, "A Survey of Migration Mechanisms of Virtual Machines," *ACM Comput. Surv.*, vol. 46, no. 3, pp. 30:1–30:33, jan 2014.

[11] F. Liu *et al.*, "Nist cloud computing reference architecture," *NIST special publication*, vol. 500, no. 2011, p. 292, 2011.

[12] F. Anhalt *et al.*, "Specifying and provisioning virtual infrastructures with hipernet," *International Journal of Network Management*, vol. 20, no. 3, pp. 129–148, 2010.

[13] P. M. Mell and T. Grance, "Sp 800-145. the nist definition of cloud computing," NIST, Gaithersburg, MD, United States, Tech. Rep., 2011.

[14] S. Bhardwaj *et al.*, "Cloud computing: A study of infrastructure as a service (iaas)," *International Journal of engineering and information Technology*, vol. 2, no. 1, pp. 60–63, 2010.

[15] RDO, "Difference between Floating IP and private IP," https://www.rdoproject.org/networking/difference-between-floating-ip-and-private-ip/, 2017, Acessado em: 01-11-2017.

[16] E. C. Junior *et al.*, "Uma taxonomia para corretagem de nuvens iaas baseada na migração de infraestruturas virtuais," *Revista Brasileira de Computação Aplicada*, vol. 9, no. 1, pp. 15–30, 2017.

[17] D. Ruck *et al.*, "Eavira: Energy-aware virtual infrastructure reallocation algorithm," in *2017 VII Brazilian Symposium on Computing Systems Engineering (SBESC)*, Nov 2017.

[18] S. Akoush *et al.*, "Predicting the performance of virtual machine migration," in *2010 IEEE International Symposium on Modeling, Analysis and Simulation of Computer and Telecommunication Systems*, Aug 2010, pp. 37–46.

[19] U. Deshpande and K. Keahey, "Traffic-sensitive live migration of virtual machines," *Future Generation Computer Systems*, vol. 72, pp. 118 – 128, 2017.

[20] C. Clark *et al.*, "Live migration of virtual machines," in *Proceedings of the 2nd Conference on Symposium on Networked Systems Design & Implementation-Volume 2*. USENIX Association, 2005, pp. 273–286.

[21] A. J. Mashtizadeh *et al.*, "Xvmotion: Unified virtual machine migration over long distance," in *2014 USENIX Annual Technical Conference (USENIX ATC 14)*. Philadelphia, PA: USENIX Association, 2014, pp. 97–108.

[22] M. R. Hines *et al.*, "Post-copy live migration of virtual machines," *SIGOPS Oper. Syst. Rev.*, vol. 43, no. 3, pp. 14–26, jul 2009.

[23] F. Zhang *et al.*, "CBase: A New Paradigm for Fast Virtual Machine Migration across Data Centers," in *Proceedings of the 17th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing*. IEEE Press, 2017, pp. 284–293.

[24] S. Rajagopalan *et al.*, "SecondSite: disaster tolerance as a service," *ACM SIGPLAN Notices*, vol. 47, no. 7, pp. 97–108, 2012.

[25] T. Wood *et al.*, "CloudNet: dynamic pooling of cloud resources by live WAN migration of virtual machines," in *ACM Sigplan Notices*, vol. 46, no. 7. ACM, 2011, pp. 121–132.

[26] I. Mansour *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*, ser. ICC '16. New York, NY, USA: ACM, 2016, pp. 62:1–62:7.

[27] S. K. Nair *et al.*, "Towards secure cloud bursting, brokerage and aggregation," in *Web services (ecows), 2010 ieee 8th european conference on*. IEEE, 2010, pp. 189–196.

[28] OpenStack, "Openstack," https://www.openstack.org/, 2017, Acessado em: 10-11-2017.

[29] Docker, "Docker - build, ship, and run any app, anywhere," Docker:https://www.docker.com/, 2017, Acessado em: 10-11-2017.

[30] "Cloudlab," https://www.cloudlab.us/index.php, 2018, Acessado em: 14-104-2018.

[31] J. Carrasco *et al.*, "Component-wise application migration in bidimensional cross-cloud environments," 2017.

[32] S. Ristov *et al.*, "P-tosca portability demo case," in *Cloud Networking (CloudNet), 2014 IEEE 3rd International Conference on*. IEEE, 2014, pp. 261–263.

[33] O. Standard, "Topology and orchestration specification for cloud applications version 1.0," 2013.

[34] H. Ali *et al.*, "A cloud interoperability broker (cib) for data migration in saas," *Future Computing and Informatics Journal*, 2017.

[35] M. Duggan *et al.*, "A network aware approach for the scheduling of virtual machine migration during peak loads," *Cluster Computing*, pp. 1–12, 2017.

[36] K. Tsakalozos *et al.*, "Live vm migration under time-constraints in share-nothing iaas-clouds," *IEEE Transactions on Parallel and Distributed Systems*, vol. 28, no. 8, pp. 2285–2298, 2017.

[37] Y. Zhao *et al.*, "Virtual network migration on the geni wide-area sdn-enabled infrastructure," *arXiv preprint arXiv:1701.01702*, 2017.