Orquestração Automatizada de Serviços de Rede em Ambientes Multi-nuvem

Laudelino A. dos Santos[1], Harrison F. S. Machado[1], Leandro C. Resendo[1], Maxwell E. Monteiro[1], Gilmar L. Vassoler[1] e Cristina K. Dominicini[1]

[1]Programa de Pós-graduação em Computação Aplicada (PPComp)
Campus Serra do Instituto Federal do Espírito Santo (IFES)

{laudelinojr,harrison.sanches}@gmail.com

{leandro,maxmonte,gilmarvassoler,cristina.dominicini}@ifes.edu.br

Abstract. *In a multi-cloud environment, the allocation of virtual services needs well defined criteria so that decision making is aligned with end-user needs and heterogeneous cloud constraints. However, there is a lack of works that explore mechanisms capable of dynamically providing inputs to the various optimization models developed to solve the multi-cloud allocation problem and automatically apply the output of these models to the clouds. This article proposes an orchestrator that is able to autonomously collect metrics from services and multi-cloud infrastructure, make them available to an optimizer, and apply dynamic allocation decisions using commercial cloud platforms.*

Resumo. *Em um ambiente multi-nuvem, a alocação de serviços virtuais precisa de critérios bem definidos para que as tomadas de decisões estejam alinhadas às necessidades do usuário final e às restrições de nuvens heterogêneas. Entretanto, existe uma lacuna de trabalhos que explorem os mecanismos capazes de fornecer dinamicamente as entradas para os diversos modelos de otimização desenvolvidos para resolver o problema de alocação multi-nuvem e aplicar a saída desses modelos nas nuvens de forma automatizada. Este artigo, propõe um orquestrador que é capaz de coletar autonomamente métricas dos serviços e da infraestrutura multi-nuvem, disponibilizá-los para um otimizador e aplicar as decisões de alocação dinâmica usando plataformas de nuvem comerciais.*

1. Introdução

O advento do paradigma de computação em nuvem permitiu a criação de ambientes de infraestrutura com recursos e aplicações suficientes para atender as necessidades atuais, e com possibilidades de expansão ou redução destes a qualquer tempo, de forma elástica [Qu et al. 2018]. Isto se deve às cinco características principais deste paradigma, que são: serviço sob demanda, amplo acesso à rede de dados, elasticidade rápida, medição de serviços e *pool* de recursos [Birje et al. 2017]. Todas estas características podem ser consideradas diferenciais positivos quando comparados ao uso da computação tradicional.

Existem vários tipos de nuvem, dentre os quais o modelo de nuvem híbrida, que tem sido bastante utilizada [Odun-Ayo et al. 2018], principalmente nas empresas que precisam manter atividades específicas na nuvem privada, muitas destas relacionadas à atividade fim da empresa. Nesta lógica, os serviços menos essenciais são enviados para a

nuvem pública. Neste mesmo contexto, muitas empresas também optam por ter muitas nuvens espalhadas em locais geograficamente distintos.

Atualmente, a maioria das plataformas de nuvem fornecem APIs (do inglês *Application Programming Interface*) para estabelecer comunicação entre elas, além de opção para interligação por meio de VPN (do inglês *Virtual Private Network*). Existe uma certa complexidade para a manipulação e gerenciamento das nuvens, mas que se bem exploradas, podem trazer grandes benefícios [de Sousa et al. 2019].

A utilização de NFV (*Network Function Virtualization*), que é responsável por separar funções de rede do hardware e oferecê-los por meio de serviços virtualizados, os chamados VNFs (do inglês *Virtual Network Function*), tem tornado desnecessário a aquisição de novos hardwares para a entrega de cada novo serviço. Desta forma, vários serviços são decompostos em vários VNFs [de Sousa et al. 2019]. Para tal, é necessário criar estas VNFs por meio de software, utilizando por exemplo o conceito de Redes Definidas por Software (do inglês SDN, *Software-Defined Networking*). Exemplos de VNF são: um *Firewall*, um servidor DHCP, um servidor virtual ou até mesmo um roteador.

A integração entre múltiplas nuvens de um ambiente híbrido para provimento de serviços virtualizados, de acordo com [Barhate and Dhore 2018], requer uma definição dos parâmetros de uso e recursos que devem ser gerenciados e compartilhados, à fim de evitar que o ambiente de nuvem se torne degradado, por vezes, sem condições de uso. Neste contexto, um Orquestrador é o responsável por orquestrar e coordenar a seleção e utilização de recursos de computação para atender aos requisitos desejados [de Sousa et al. 2019]. Além da concepção de um novo serviço em nuvem, este deve também se preocupar com a manutenção do ambiente, à fim de garantir a elasticidade dos recursos disponíveis para determinado serviço e, de preferência, de forma automática.

Uma parte central do funcionamento do orquestrador é o processo de otimização, que consiste em alocar as instâncias das funções de serviço requisitadas na infraestrutura para um conjunto de solicitações de serviço a fim de encontrar uma solução eficiente de acordo com uma função objetivo específica (por exemplo, maximizar o número de instâncias [Luizelli et al. 2015] e minimizar o custo operacional [Bari et al. 2015]). Vale destacar a importância da existência de uma lógica para efetuar esta alocação (do inglês, *placement*) dos serviços virtualizados de forma equilibrada entre as nuvens [Laghrissi and Taleb 2018]. Para tal pode ser inclusive levado em consideração a proximidade do usuário final a estes recursos, de forma a reduzir a latência fim-a-fim do serviço para o usuário [Cziva and Pezaros 2017].

Existem diversos modelos de otimização na literatura para resolver o problema de orquestração em nuvem [Bhamare et al. 2017, Herrera and Botero 2016], mas muito menos atenção tem sido dada aos mecanismos que permitiriam que esses modelos sejam de fato implementados em nuvens comerciais de forma dinâmica. Por exemplo, o orquestrador precisa contar com um processo de monitoramento contínuo que permita obter informações dinamicamente sobre o estado dos recursos da infraestrutura das nuvens, as métricas dos serviços e as requisições a serem instaladas, além de traduzir esses dados de acordo com os parâmetros de entrada do modelo de otimização adotado. É necessário ainda que os modelos de otimização utilizados possam ser configurados de maneira flexível de acordo com as políticas administrativas e acordos de nível de serviço

(*Service Level Agreements*, SLAs). Ao final, a saída do processo de otimização deve ser convertida em ações que irão efetivamente configurar os recursos em múltiplas nuvens com diferentes interfaces de comunicação para implantação do serviço requisitado.

Com esta motivação, este trabalho propõe implementar uma orquestração automatizada de serviços de rede em ambientes multi-nuvem, levando-se em consideração métricas como latência entre nuvens, utilização de recursos computacionais das nuvens e métricas de serviço com a finalidade de gerir os recursos de forma eficiente e atender aos requisitos dos serviços. Em especial, nossa proposta permite que tanto os parâmetros do modelo de otimização quanto os parâmetros de monitoramento sejam configurados de forma flexível e dinâmica e que os resultados desses processos sejam automaticamente traduzidos em comandos de configuração para múltiplas nuvens.

Foi implementado um protótipo com tecnologias utilizadas em ambientes reais de nuvem, como os frameworks Open Source Mano (OSM) e OpenStack. Os resultados obtidos mostram que é possível mudar a decisão de orquestração nas nuvens de forma automatizada com base em métricas coletadas do ambiente de infraestrutura e configurações de políticas de alto nível estabelecidas pelo usuário final para definição de modelos de otimização, de forma a entregar a este um ambiente mais adequado às suas necessidades.

Por fim, o artigo está organizado da seguinte forma: a Seção 2 discute os trabalhos relacionados e destaca as principais contribuições deste artigo. A Seção 3 apresenta a proposta com uma visão de alto nível, onde a respectiva implementação será apresentada na Seção 4, com uma arquitetura detalhada de todo o processo proposto. A seção 5 avalia a proposta por meio de testes realizados no protótipo, testes estes divididos em diversos cenários. Ao final, a Seção 6 apresenta a conclusão e possíveis trabalhos futuros.

2. Trabalhos Relacionados

Observa-se que vários trabalhos importantes já abordaram este problema de orquestração em ambientes híbridos sob o ponto de vista de um problema de otimização. Pode-se citar [Bhamare et al. 2017], cujo foco é diminuir o fluxo entre datacenters através do posicionamento eficiente das cadeias de funções. As cadeias de funções aqui citadas são conjuntos de VNF. Segundo [Rosa et al. 2014], de forma prática, um VNF pode ser por exemplo, um servidor DHCP ou um Firewall, também chamados de *Network Function*, que é uma Função de Rede.

O trabalho em [Sun et al. 2019], também trata do problema do posicionamento usando Programação Linear Inteira (ILP, ou Integer Linear Programming) e otimiza o consumo de energia com foco no atendimento de requisitos dos encadeamentos. Já [Dietrich et al. 2017] resolve o problema do roteamento, através de otimização por ILP, propondo o particionamento de cadeias em múltiplos domínios conectados por gateways virtuais com o posicionamento estratégico das funções nestes domínios. De acordo com [Gupta et al. 2017], este também otimiza o posicionamento de funções em ambientes híbridos usando um algoritmo preditivo com o objetivo de minimizar a latência das cadeias de funções.

Embora algumas soluções de otimização para o problema de orquestração de serviços híbridos tenham sido propostas, poucos trabalhos focam no desenvolvimento dos mecanismos de orquestração para permitir a implantação dessas soluções de alocação de recursos nas infraestruturas reais de rede de datacenter.

Muito progresso tem sido feito e muitas soluções comerciais foram propostas na área de orquestração [Mijumbi et al. 2016], tais como: OPNFV, Open Baton, Open Source MANO (OSM) e Cloudify. O trabalho de [de Sousa et al. 2019] também detalha várias iniciativas acadêmicas, tais como: T-NOVA, Unify, e SONATA. Neste contexto, um dos principais desafios é como unificar as tecnologias dos diferentes domínios de forma que seja possível configurar os recursos que compõe um serviço fim-a-fim [de Sousa et al. 2019]. Além disso, essas soluções não fornecem mecanismos que permitam alterar os mecanismos internos de tomada de decisão de forma dinâmica, considerando-se a configuração de diferentes modelos de otimização.

Outro desafio importante é como integrar os diferentes planos de serviço, gerenciamento e controle para atender aos requisitos das aplicações e SLAs [Guerzoni et al. 2017]. Neste sentido, há uma escassez de trabalhos até o momento, que se refira a propostas de soluções de orquestração para lidar com esses desafios, tornando viável a implementação integrada de serviços virtualizados em ambientes híbridos.

Sendo assim, considerando o estado da arte e a relevância dos trabalhos relacionados citados, é possível confirmar a relevância do problema abordado neste artigo para fornecer um solução de orquestração de serviços que integre as etapas de monitoramento, otimização e configuração automatizada em um ambiente multi-nuvem de produção.

3. Proposta

Para resolver o problema descrito, a proposta deste artigo é desenvolver um orquestrador, denominado PLAO (*PLAcement Orchestrator*), capaz de provisionar e gerenciar recursos de forma integrada e automatizada entre múltiplas nuvens, levando-se em consideração os SLAs e as políticas definidas pelos usuários. A Figura 1 mostra a visão de alto nível da arquitetura de orquestração proposta. As interfaces de *northbound* do orquestrador (em amarelo) interagem com o módulo de Monitoramento e com o módulo *Dashboard*. O primeiro fornece informações de monitoramento sobre o estado dos recursos de nuvem e rede, que fundamentam a tomada de decisão para atendimento das demandas de serviço com base na visão centralizada de toda a infraestrutura. O segundo permite que o usuário defina requisições, políticas e SLAs para serviços de acordo com requisitos de desempenho e escalabilidade.

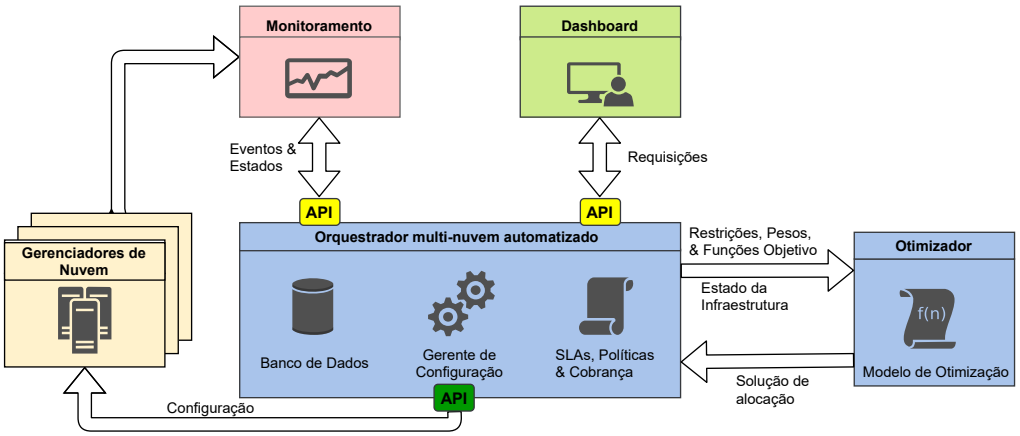


Figura 1. Visão de alto nível da arquitetura de orquestração.

A decisão inicial de orquestração define onde as instâncias de cada função que compõe o serviço serão alocadas considerando os recursos de cada um dos domínios e os requisitos do serviço. Depois, ao longo do ciclo de vida do serviço é necessário monitorar as métricas definidas, tais como: custo dos recursos alocados, utilização de CPU e memória de VMs, utilização de recursos de redes e latência de comunicação do serviço. Com base nesse monitoramento, pode ser necessário tomar decisões de orquestração (e.g., dimensionamento e migração) para assegurar o atendimento dos SLAs.

Um dos componentes principais do processo de orquestração é o otimizador. A ideia central da nossa proposta é permitir a configuração dinâmica dos parâmetros dos modelos de otimização. Dessa forma, pode-se alterar restrições, pesos e funções objetivo a depender dos requisitos a serem atendidos. Por fim, para implementar o serviço e aplicar as decisões de orquestração, o Orquestrador precisa ter interfaces de *southbound* (em verde na figura 1) com os Gerenciadores de Nuvens, uma vez que são estes elementos que de fato controlam os recursos a serem alocados.

4. Implementação de prova de conceito

Como prova de conceito da proposta da seção anterior, foi desenvolvido um protótipo utilizando-se tecnologias largamente utilizadas no mercado, com o objetivo de demonstrar o mecanismo para automatizar o processo de orquestração de serviços virtuais em um ambiente multi-nuvem. No escopo deste trabalho, as plataformas OpenStack e Open Source Mano (OSM) serão adotados como gerenciadores de nuvem privada, onde o OSM será utilizado para gerenciar o ciclo de vida dos serviços virtuais, e o OpenStack para prover os recursos para hospedá-los.

4.1. Tecnologias habilitadoras e limitações

Para disponibilizar o ambiente de nuvem foi utilizado o OpenStack, que é um software de código aberto para nuvem capaz de disponibilizar e gerenciar uma gama de recursos como: recursos de armazenamento, rede e máquinas virtuais, todos integrados entre si; possibilitando assim que qualquer empresa possa criar um ambiente próprio de nuvem, poderoso e integrável com outras soluções de nuvem como, por exemplo, AWS e Google. É a ferramenta de código aberto mais utilizada[1] atualmente em nuvens privadas, o que nos motivou a usá-la. Foi utilizado a versão Victoria do OpenStack.

Para o gerenciamento de ciclo de vida e alocação dos serviços virtuais, foi utilizado o OSM (do inglês, *Open Source Mano*), projeto alinhado às especificações da European Telecommunications Standards Institute (ETSI) Industry Specification Group (ISG) NFV, que criou um framework de referência para orquestração dos serviços de rede e gerenciamento de ciclo e vida, mais conhecido como NFV MANO. O OSM foi escolhido por suportar a integração com diferentes nuvens, como AWS, OpenStack e VMware, além de ter grandes patrocinadores da área de telecomunicações e um grande grupo de colaboradores [Mamushiane et al. 2019]. Desta forma, os serviços virtuais serão inseridos no ambiente de nuvem via OSM, que por sua vez irá processar as decisões de orquestração e enviar as requisições para o OpenStack. Foi utilizada a versão 8 com containers do OSM.

O OSM possui um módulo de Placement, denominado PLA[2], responsável pela

[1] `https://resources.flexera.com/web/pdf/report-state-of-the-cloud-2020.pdf`

[2] `https://osm.etsi.org/gitweb/?p=osm/PLA.git`

alocação de serviços virtuais, onde as tomadas de decisão usam por referência os custos fixos de cada VNFD (Descritor de VNF) em relação a cada nuvem presente no ambiente. Vale ressaltar que no OSM, um VNFD precisa estar sempre associado a pelo menos um NS (do inglês, *Network Service*), que representa a composição de um conjunto de VNFs para provisionamento de um serviço. Baseado nesses custos, o OSM toma a decisão de alocar as VNFs de um NS de forma a minimizar o custo total.

Os custos para cada VNF são arbitrados em um arquivo de configuração e são inseridos de forma manual, assim como os custos dos links entre as nuvens e os valores de latência e jitter. As Figuras 2 e 3 mostram exemplos de arquivos de configuração para as VNFs e os links, respectivamente. Observe na Figura 2 que o custo de cada nuvem em relação ao VNFD é representado pelo atributo *price*. As URLs para acesso às nuvens e os respectivos nomes utilizados de acordo com o cadastro das VIMs (*Virtual Infrastructure Managers*) são representados pelos atributos *vim_url* e *vim_name*. Conforme Figura 3, o arquivo de configuração para o Link possui o atributo *pil_price*, que representa o custo entre o par de Links listados no atributo *pil_endpoints*. Além deste, existem outros atributos, que são: *pil_description*, *pil_latency* e *pil_jitter*, que representam respectivamente a descrição do link, a latência e o jitter.

```
- vnfd: VNFA
  prices:
  - vim_url: http://10.172.200.6:5000/v3
    vim_name: openstack1
    price: 10
  - vim_url: http://10.172.200.12:5000/v3
    vim_name: openstack2
    price: 30
```

Figura 2. Exemplo de arquivo de configuração do OSM para VNFDs.

```
- pil:
  - pil_description: Link between OpenStack1 and OpenStack2
    pil_price: 5
    pil_latency: 5
    pil_jitter: 2
    pil_endpoints:
      - openstack1
      - openstack2
```

Figura 3. Exemplo de arquivo de configuração do OSM para links.

A partir desses arquivos de configuração, o módulo de placement do OSM, o PLA, toma a decisões de alocação de serviço, considerando o mínimo da somatória de todos os custos de VNFs que compõe o NS e o custo dos links entre nuvens, quando aplicável. Além da definição dos custos e links, o modelo de otimização do OSM também permite a especificação de restrições. A Figura 4 mostra um exemplo de comando para instanciação de um NS com restrição em relação ao link de dados. Neste exemplo, a latência não pode ultrapassar o valor de 2 e o jitter não pode ultrapassar o valor de 1.

Entretanto, essa abordagem do OSM possui duas limitações principais: (i) não é possível configurar os custos de forma dinâmica; e (ii) os custos são arbitrados como valores fixos sem relação com os estados dos recursos da infraestrutura ou os requisitos

```
osm ns-create --nsd_name teste_artigo  --ns_name test2ArtigoPLA --vim_account
↪  openstack1 --config '{placement-engine: PLA, placement-constraints:
↪  {vld-constraints: [{id: ns_vl_2mlm, link-constraints: {latency: 2, jitter: 1}}]},
↪  wim_account: False}'
```

Figura 4. Exemplo de comando OSM para execução de NS com restrição.

de serviço. Para resolver este problema e permitir a implementação da nossa proposta, este trabalho propõe a implementação da solução PLAO, que visa coletar as métricas necessárias para subsidiar a alocação de recursos e criar novos pontos de controle que permitirão a configuração dos custos do modelo de otimização do OSM de forma automatizada de acordo com as políticas definidas para os serviços e o estado da infraestrutura.

4.2. Implementação da arquitetura PLAO

Na arquitetura PLAO, conforme Figura 5, são identificadas três camadas principais, que são: camada de usuário, onde está contido o *dashboard* de interação com o usuário; a camada de nuvens, que abriga toda a infraestrutura envolvida; e o PLAO, que é responsável por gerenciar e processar as requisições originadas da camada de usuário e nuvens.

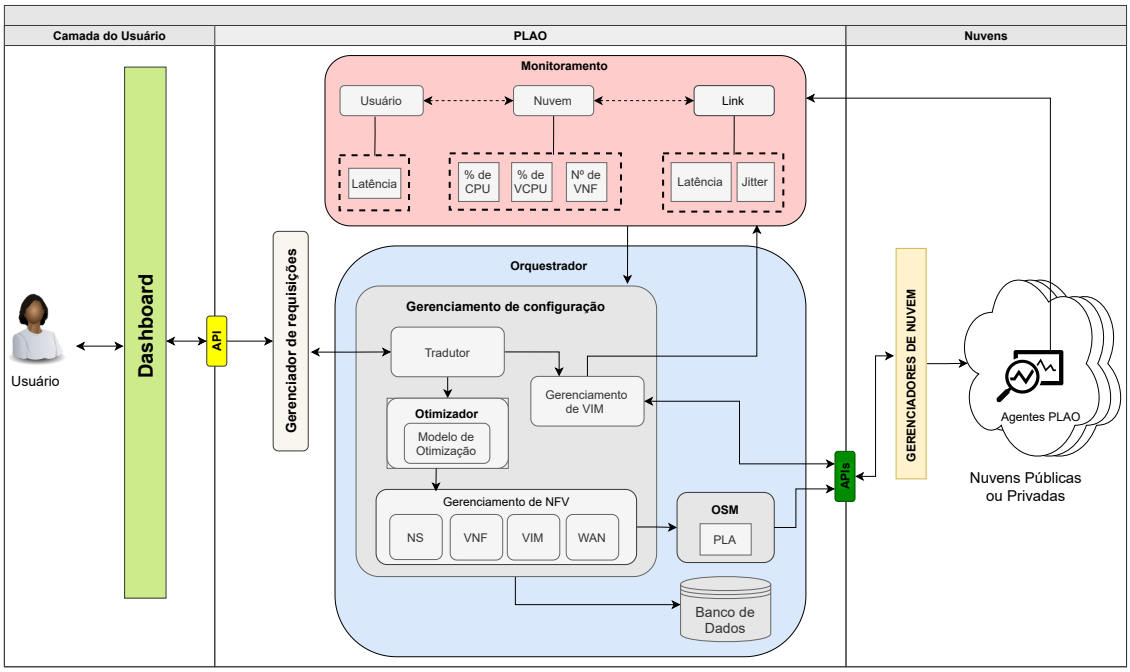


Figura 5. Arquitetura do PLAO

O *dashboard* fornece ao usuário um ambiente onde ele possa criar e configurar serviços conforme seus requisitos. Após as escolhas do usuário, o *dashboard* repassa essas configurações do usuário para o PLAO, onde elas serão traduzidas e processadas.

A camada PLAO é composta pelos componentes Gerenciador de Requisições, Monitoramento e Orquestrador. O Gerenciador de Requisições irá processar e encaminhar as requisições enviadas pelo usuário. O Orquestrador conta com 3 módulos principais, que são: Gerenciamento de Configuração, Banco de Dados e OSM. No banco de dados

são armazenados dados para autenticação de serviços e usuários, além de armazenar logs, como horário de criação e exclusão de VM's, projetos criados e deletados, entre outros.

O Gerenciamento de Configuração tem a responsabilidade de traduzir, gerenciar e orquestrar as requisições recebidas pelo Gerenciador de requisições. Para isto, ele possui os módulos Tradutor, Otimizador, Gerenciamento de VIM e Gerenciamento de NFV. O módulo de Gerenciamento de NFV se comunica com o OSM via requisições HTTP. Este tem a responsabilidade de repassar as traduções sobre a VIM e VNFs para o OSM, que fará o provisionamento do NS de acordo com as nuvens e politicas escolhidas pelo usuário. É importante ressaltar que antes que seja possível criar um NS no OSM, é necessário traduzir a requisição do usuário e convertê-la em um arquivo de configuração que será utilizado para alocação. O primeiro passo é garantir que este usuário tenha as credenciais de acesso necessárias, bem como os projetos que irão hospedar o NS com as respectivas políticas de segurança, já criadas nas nuvens que serão utilizadas.

O módulo de Gerenciamento de VIM, gerencia os usuários e projetos de cada serviço que utiliza o PLAO. Esse módulo contém funções como criar, deletar e atualizar dados de usuário e de projetos que estão sendo utilizados em um determinado serviço, além de funcionalidades de criação de redes e de grupos de segurança com politicas estabelecidas pelo usuário. Para isto são utilizadas APIs fornecidas por cada nuvem para garantir que essas configurações estejam prontas previamente à utilização do OSM. Na implementação do protótipo, o módulo utilizou o *openstackSDK*[3], uma biblioteca para construir aplicativos para trabalhar com nuvens OpenStack, mas outras nuvens, como o AWS, oferecem funcionalidades semelhantes. O módulo de Gerenciamento de VIM também permite utilizar as APIs de coletas de métricas e monitoramento de cada nuvem, que servirão de entrada para o módulo de Monitoramento.

O Otimizador otimiza as configurações da nuvem de modo a prover uma melhor alocação de serviço para o usuário. Este módulo, com base nas traduções e dados recebidos do Gerenciamento de Configuração, faz a otimização dos custos de cada VNF por Nuvem. Além das configurações do usuário, outras variáveis podem influenciar no custo da VNF, como por exemplo a extrapolação do limite de uso de CPU da Nuvem. O módulo tem a responsabilidade de alterar todos os parâmetros de custo, tanto de VNFs quanto de links, e repassa-las ao Gerenciamento de NFV. O módulo de Monitoramento tem a responsabilidade coletar dados do ambiente de infraestrutura e que irão subsidiar a tomada de decisões do Orquestrador. São coletados dados das Nuvens, do link entre as nuvens e do usuário. Mais detalhes na seção 4.3.

A camada Nuvens é onde estão localizadas a infraestrutura. Ela contém gerenciadores de nuvem, privada ou pública, que ao receberem uma requisição, gerenciam e configuram a nuvem na qual a requisição foi destinada. Na implementação deste trabalho, foram utilizados nuvens com o Openstack, conforme já mencionado.

4.3. Coleta de dados e cálculo de custos

Nesta implementação foram utilizadas as APIs do Openstack *Keystoneauth*[4], que efetua a autenticação nas nuvens, e *Nova Client*[5], que fornece uma interface para acesso a

[3] `https://pypi.org/project/openstacksdk`
[4] `https://docs.openstack.org/keystoneauth/latest/`
[5] `https://docs.openstack.org/python-novaclient/latest/`

informações do Nova. São coletados dados de percentual utilizado de vCPU, memória, disco, e a quantidade de VNF's instanciadas. Em nenhuma destas APIs é possível coletar informações do Sistema Operacional, nem parâmetros de rede como Latência e Jitter entre as nuvens, ou entre as nuvens e o usuário final, sendo necessário a criação do PLAO.

Conforme Figura 6, o processo de coleta de métricas será feito através de uma comunicação contínua entre um agente e um servidor da arquitetura PLAO. Todos os dados, após coletados no cliente, serão enviados para o PLAO, onde serão processados de acordo com as políticas e SLAs definidas pelo usuário final. Após este processo, os arquivos de configuração do *placement* serão ajustados e enviados para o módulo de *placement* do OSM e utilizados na próxima instanciação de serviços.

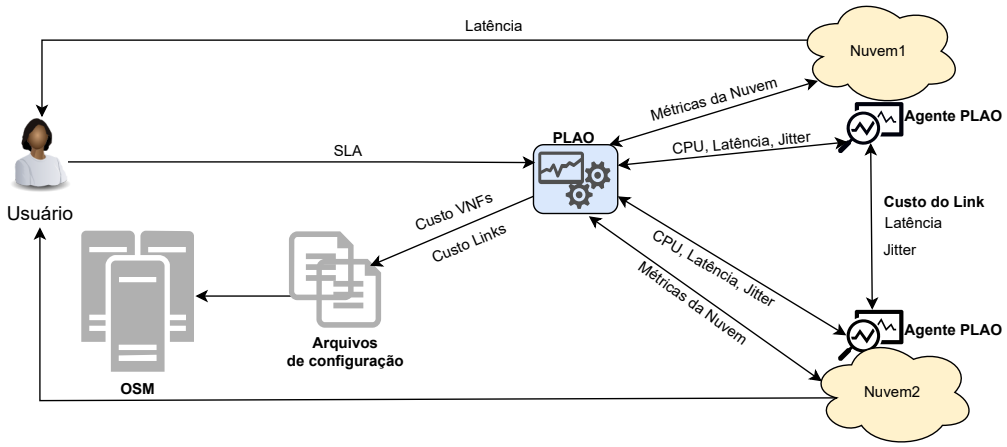


Figura 6. Esquema de coleta de dados do PLAO

Para que seja possível atribuir custos aos links entre as nuvens, o agente PLAO utiliza as ferramentas `iperf` e `ping`, respectivamente, para coletar os dados de jitter e latência entre cada nuvem. Para a coleta de jitter, serão enviados pacotes UDP durante 5 segundos. Para a coleta da latência, o `ping` enviará 5 pacotes para efetuar a medição e enviará ao PLAO caso haja uma variação em módulo maior que 2ms em relação à última coleta. Futuramente esta coleta será realizada utilizando uma média de "x" intervalos anteriores a fim de evitar instabilidades devido a picos isolados.

Este trabalho também propõe que seja possível utilizar métricas relativas ao serviço. Na nossa prova de conceito, medimos a latência entre o usuário final e as nuvens como componente do cálculo do custo de uma VNF, com o objetivo de melhorar a experiência do usuário, visto que provavelmente o mesmo fará uso da nuvem com a menor latência. Diferente das outras métricas, esta coleta será por demanda em cada nuvem após o usuário final requisitar um novo serviço. Para a coleta da latência entre cada nuvem e o usuário final também foi utilizada o envio de 5 pacotes com a ferramenta `ping`.

Ainda, para evitar que apenas uma variável seja levada em consideração na composição do custo dos experimentos da prova de conceito, uma métrica coletada da nuvem de percentual de uso de CPU será utilizada em uma média ponderada juntamente com a métrica de latência para cálculo do custo da VNF. Para a coleta do dado de CPU é utilizado a bilioteca *psutil* [6], que fornece informações sobre os processos em execução.

[6] `https://github.com/giampaolo/psutil`

Ao solicitar a instanciação das VNFs, o usuário deverá informar um peso entre 0 e 9 para cada uma destas métricas citadas, de forma que a soma destes seja 10. Desta forma, esta ponderação entre latência e percentual de uso de CPU permitirá ao demandante do serviço configurar estes pesos, por exemplo, de acordo com as características da aplicação, num cenário onde cada VNFD as represente. Outra funcionalidade desta implementação é a inserção de um custo adicional à nuvem que tiver uma métrica com valores além do limite pré-estabelecido dentro da lógica do PLAO, para indicar, por exemplo, que a nuvem se encontra em um estado de degradação. Para esta implementação, faremos uso da métrica de percentual de uso de CPU e o valor a ser acrescido será 10.

Além das ponderações indicadas pelo usuário para cada VNFD, o usuário também pode informar algumas regras de nível de serviço ou SLAs (do inglês, *Service Level Agreement*) para uso do link de dados entre as nuvens. Esta regra é útil quando o usuário for instanciar ao mesmo tempo duas ou mais VNFDs e exista a possibilidade de cada uma ir para uma nuvem distinta. Exemplo: Não fazer uso do link de dados entre as nuvens nas situações em que os dados ultrapassem os valores de latência e *jitter* pré-estabelecidos pelo usuario no momento da instanciação.

5. Avaliação da Proposta

Para avaliar a aplicabilidade da arquitetura proposta e realizar os experimentos, foi criado um ambiente de teste com 5 servidores virtuais e uma máquina de usuário, dos quais 4 servidores compõem duas nuvens OpenStack (N1 e N2) com instalação do tipo multi-nó (2 servidores para cada nuvem). Por fim, o quinto servidor foi destinado para a instalação do PLAO, que contém, entre outros componentes (vide Figura 5), o OSM. As configurações de cada servidor estão descritas na Tabela 1.

Tabela 1. Configuração do ambiente de testes

Nuvem	Tipo de Nó	qntd	vcpus	vmemory (GB)	vdisk (GB)
N1	OpenStack Controller	1	2	8	102
N1	OpenStack Compute	1	4	8	102
N2	OpenStack Controller	1	2	8	102
N2	OpenStack Compute	1	4	8	102
-	PLAO/OSM	1	1	4	60
-	Máquina do Usuário	1	-	-	-

O experimento foi dividido em 5 cenários, com os seguintes passos executados em cada um deles: **i)** Inicialização do serviço PLAO com espera de 10 segundos; **ii)** Inicialização do agente PLAO nos servidores OpenStack Compute com espera de 30 segundos e início de coleta dos dados da nuvem; **iii)** Solicitação de instanciação de um NS ao PLAO. **iv)** Intervalo de 180 segundos enquanto os dados são coletados; **v)** Finalização do Agente PLAO nos servidores Openstack Compute; **vi)** Finalização do serviço PLAO; **vii)** Intervalo sem testes de 60 segundos até o início do próximo cenário de testes. Com exceção da transição para o Cenário 5, em que ocorreu uma espera de 80 segundos, pois dependia da instanciação de novos NSs.

Em todos os cenários do experimento, o NS é composto por 2 VNFs, VNFA e VNFB, cujos pesos para cada tipo de métrica são: VNFA, peso 7 para latência e peso 3 para CPU, e VNFB, com peso 1 para latência e peso 9 para CPU. Neste contexto, cada

VNFD representa uma VM e a latência é entre o usuário e a nuvem. Ao ultrapassar o limite de 90% de uso de CPU, a nuvem será considerada degradada. Para atingir o objetivo do experimento em testar a dinâmica da orquestração, foram analisados parâmetros diferentes em alguns dos cenários, modificando condições da infraestrutura (e.g., aumento de latência entre nuvens ou requisições concorrentes) ou da requisição (e.g., restrições de latência aceitável), onde: **i)** Cenário 1 - Referência; **ii)** Cenário 2 - Aumento de latência para 15ms entre as nuvens 1 e 2 com o uso do aplicativo TC[7] (*Traffic Control*), um utilitário para manipular controle de tráfego; **iii)** Cenário 3 - Diminuição da latência de volta para as condições do Cenário 1; **iv)** Cenário 4 - Aplicação de uma restrição imposta pelo usuário para o cumprimento de SLA de custo de link de no máximo 2; **v)** Cenário 5 - Instanciação de 2 NSs extras para representar requisições concorrentes e forçar que percentual de uso de CPU da nuvem ultrapassasse o limite pré-estabelecido na aplicação. Também foi mantida a restrição imposta pelo usuário para o cumprimento de SLA de custo de link de no máximo 2.

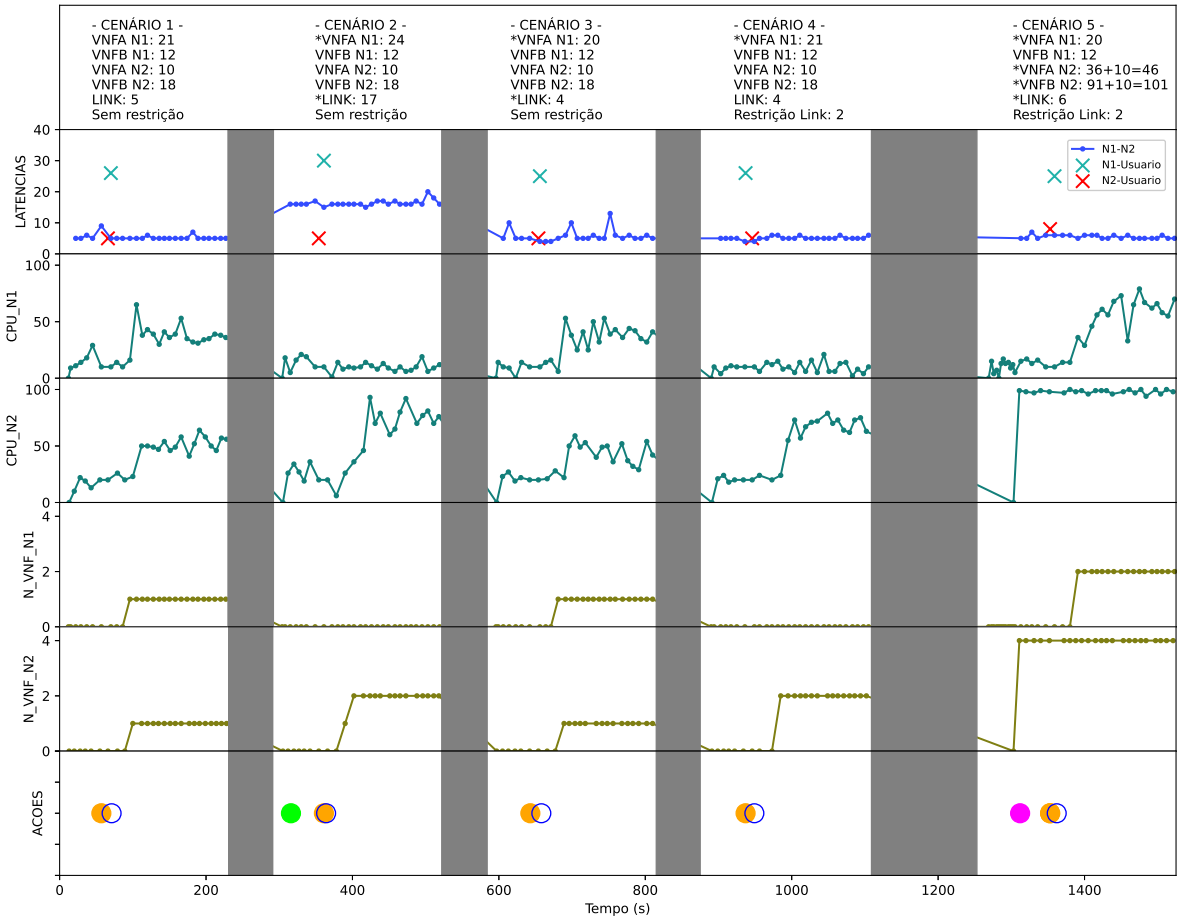
Tabela 2. Detalhamento da lógica alocação do Cenário 1

VNF	Nuvem	Dado	Peso	Valor	Custo
VNFA	N1	Latencia para Usuario (ms) CPU(%)	7 3	26 10	$\lfloor 0.7 * 26 + 0.3 * 10 \rceil = 21$
VNFB	N1	Latencia para Usuario (ms) CPU(%)	1 9	26 10	$\lfloor 0.1 * 26 + 0.9 * 10 \rceil = 12$
VNFA	N2	Latencia para Usuario (ms) CPU(%)	7 3	5 20	$\lfloor 0.7 * 5 + 0.3 * 20 \rceil = 10$
VNFB	N2	Latencia para Usuario (ms) CPU(%)	1 9	5 20	$\lfloor 0.1 * 5 + 0.9 * 20 \rceil = 18$

Opção	Custo	Resultado
1	VNFA N1 + VNFB N1 + Link = 21+12+0	33
2	VNFA N1 + VNFB N2 + Link =21+18+5	44
3	**VNFA N2 + VNFB N1 + Link =10+12+5**	**27**
4	VNFA N2 + VNFB N2 + Link = 10+18+0	28

Para ilustrar melhor a lógica utilizada pelo PLAO, vamos tomar como exemplo o Cenário 1, em que a orquestração automatizada decidiu instanciar uma VNF em cada nuvem para o NS composto por VNFA e VNFB, o que demonstra que neste momento esta era a melhor opção de custo total. A formação do custo de cada VNF por nuvem pode ser observado na Tabela 2, onde constam os valores que são recebidos das nuvens e processados com os devidos pesos escolhidos pelo usuário final no momento da requisição. Para o cálculo do custo total deve-se considerar o custo do link entre as nuvens nos casos em que o NS possui VNFs alocadas em nuvens diferentes. A Tabela 2 mostra também o cálculo do custo total para todas as combinações de alocação (VNFA e VNFB em uma única nuvem ou em nuvens diferentes) e destaca a opção escolhida de menor custo. Esta mesma lógica foi utilizada para o cálculo do custo total e consequente decisão de alocação para todos os cenários do experimento.

[7] https://access.redhat.com/documentation/en-us/red_hat_enterprise_linux/8/html/configuring_and_managing_networking/linux-traffic-control_configuring-and-managing-networking



Cor	Ação
○	Instanciação de NS.
●	Alteração do custo do link, considerando latência e jitter entre nuvens.
●	Alteração do custo das VNFDs de acordo com a latência para o usuário final e percentual de uso de CPU da nuvem.
●	Aumento dos custos de todos os VNFDs da nuvem após degradação do uso de CPU.

Figura 7. Experimento com 5 cenários

O gráfico da Figura 7 demonstra os resultados obtidos no experimento e exibe uma tabela com os custos calculados dinamicamente para os VNFDs em cada cenário, além de gráficos informando a latência entre as nuvens N1 e N2 e também entre nuvem e usuário final, o percentual de uso de CPU das nuvens N1 e N2, a quantidade de VNFs instanciadas em N1 e N2, além de uma relação gráfica das ações executadas pelo orquestrador.

Conforme já detalhado, no Cenário 1 o cálculo do custo total definiu que a melhor opção de alocação para as condições apresentadas foi alocar a VNFA na nuvem N1 e a VNFB na nuvem N2. No Cenário 2, após aumento da latência entre as nuvens, se tornou mais vantajoso instanciar todas as VNFs na nuvem N2. No Cenário 3, a latência voltou ao nível do Cenário 1 e se manteve o mesmo comportamento inicial. No Cenário 4, após a presença do SLA restringindo o valor da latência do link em 2 ms, se tornou mais vantajoso instanciar todas as VNFs na nuvem 2, pois o link em questão era 4 e não atendia ao SLA imposto. No cenário 5, após degradação da nuvem N2 com um aumento do percentual de uso da CPU devido a requisições concorrentes, o custo foi elevado para todos os VNFDs da nuvem N2, conforme lógica do PLAO. Desta forma, diferente do

Cenário 4, se tornou mais vantajoso instanciar todas as VNFs na nuvem N1 no Cenário 5. Vale ressaltar que neste quinto cenário, levamos em consideração que outra requisição, concorrente a este experimento, tenha sido instanciada antes de iniciar o experimento, levando assim a um aumento de consumo de CPU em N2.

Observe na Tabela 3 um resumo das instanciações das VNFs nos 5 cenários, onde cada "X" corresponde a uma VNF instanciada. Os resultados obtidos com o protótipo demonstram que o PLAO consegue monitorar diferentes métricas da infraestrutura e do serviço em um ambiente multi-nuvem e reagir a variações dessas métricas de acordo com as políticas de alto nível definidas pelo usuário na requisição de serviços, traduzindo essas políticas em parâmetros do modelo utilizado por um processo de otimização e alterando de forma dinâmica e automatizada as tomadas de decisão do orquestrador.

Tabela 3. Alocação de VNFs por Nuvem

Nuvens	Cenário 1	Cenário 2	Cenário 3	Cenário 4	Cenário 5
Nuvem 1	X		X		XX
Nuvem 2	X	XX	X	XX	

6. Conclusão e Trabalhos Futuros

Este trabalho propôs, implementou e avaliou, de modo experimental com tecnologias de produção, uma solução para automatizar a orquestração multi-nuvem, considerando políticas de alto nível definidas pelo usuário para métricas da infraestrutura e do serviço. Com isto, conseguimos provar que é possível monitorar essas métricas e utilizá-las na configuração do modelo de otimização que define a tomada de decisão. Conseguimos também alterar os custos das VNFs assim que uma das nuvens se tornar degradada ou o parâmetro selecionado passar do limite pré-estabelecido. Também é importante ressaltar que esta automação possibilita ao usuário ponderar custos que estejam relacionados a características das suas aplicações e influenciar o processo de orquestração.

Como trabalhos futuros destaca-se a necessidade da inclusão de testes com nuvens públicas (e.g., AWS), além da inserção de novas métricas para composição dos custos das nuvens e avaliação da escalabilidade dos métodos de coleta de latência. Pretendemos também explorar cenários dinâmicos em que o orquestrador modifica incrementalmente as decisões de alocação já realizadas quando as condições da nuvem variam, além de estender os modelos para cobrir o encadeamento de funções de rede. Por fim, estamos investigando como representar modelos de otimização mais complexos no OSM.

Agradecimentos

Agradecemos ao IFES pelo apoio financeiro via Programa Institucional de Apoio à Pós-graduação Stricto Sensu (Propós) e Programa Pibic, e à FAPES pelo suporte ao laboratório CIDIG do Centro de Pesquisa Inovação e Desenvolvimento do Espírito Santo. Este trabalho faz parte do projeto SAWI - Edital PIPE/FAPESP, CGI.br e MCTIC.

Referências

Barhate, S. M. and Dhore, M. (2018). Hybrid cloud: A solution to cloud interoperability. In *2018 Second International Conference on Inventive Communication and Computational Technologies (ICICCT)*, pages 1242–1247. IEEE.

Bari, M. F. et al. (2015). On orchestrating virtual network functions. In *Network and Service Management (CNSM), 11th International Conference on*, pages 50–56. IEEE.

Bhamare, D. et al. (2017). Optimal virtual network function placement in multi-cloud service function chaining architecture. *Computer Communications*, 102:1–16.

Birje, M. N. et al. (2017). Cloud computing review: concepts, technology, challenges and security. *International Journal of Cloud Computing*, 6(1):32–57.

Cziva, R. and Pezaros, D. P. (2017). On the latency benefits of edge nfv. In *2017 ACM/IEEE Symposium on Architectures for Networking and Communications Systems (ANCS)*, pages 105–106. IEEE.

de Sousa, N. F. S. et al. (2019). Network service orchestration: A survey. *Computer Communications*, 142:69–94.

Dietrich, D. et al. (2017). Multi-provider service chain embedding with nestor. *IEEE Transactions on Network and Service Management*, 14(1):91–105.

Guerzoni, R. et al. (2017). Analysis of end-to-end multi-domain management and orchestration frameworks for software defined infrastructures: an architectural survey. *Transactions on Emerging Telecommunications Technologies*, 28(4):e3103.

Gupta, L. et al. (2017). Colap: A predictive framework for service function chain placement in a multi-cloud environment. In *2017 IEEE 7th Annual Computing and Communication Workshop and Conference (CCWC)*, pages 1–9. IEEE.

Herrera, J. G. and Botero, J. F. (2016). Resource allocation in nfv: A comprehensive survey. *IEEE Transactions on Network and Service Management*, 13(3):518–532.

Laghrissi, A. and Taleb, T. (2018). A survey on the placement of virtual resources and virtual network functions. *IEEE Communications Surveys & Tutorials*, 21(2):1409–1434.

Luizelli, M. C. et al. (2015). Piecing together the NFV provisioning puzzle: Efficient placement and chaining of virtual network functions. In *2015 IFIP/IEEE International Symposium on Integrated Network Management*, pages 98–106, Ottawa.

Mamushiane, L. et al. (2019). Overview of 9 open-source resource orchestrating etsi mano compliant implementations: A brief survey. In *2019 IEEE 2nd Wireless Africa Conference (WAC)*, pages 1–7. IEEE.

Mijumbi, R. et al. (2016). Management and orchestration challenges in network functions virtualization. *IEEE Communications Magazine*, 54(1):98–105.

Odun-Ayo, I. et al. (2018). Cloud computing architecture: A critical analysis. In *2018 18th International Conference on Computational Science and Applications (ICCSA)*, pages 1–7. IEEE.

Qu, C., Calheiros, R. N., and Buyya, R. (2018). Auto-scaling web applications in clouds: A taxonomy and survey. *ACM Computing Surveys (CSUR)*, 51(4):1–33.

Rosa, R. et al. (2014). Network function virtualization: Perspectivas, realidades e desafios. *Minicursos SBRC*.

Sun, G. et al. (2019). Energy-efficient and traffic-aware service function chaining orchestration in multi-domain networks. *Future Generation Computer Systems*, 91:347–360.