Além do OpenStack: Disponibilizando o Suporte para Funções Virtualizadas de Rede NFV-MANO no CloudStack

José Flauzino, Vinicius Fulber-Garcia, Giovanni Venâncio, Elias P. Duarte Jr.

Universidade Federal do Paraná (UFPR), Departamento de Informática,
Caixa Postal 19018 81531-990, Curitiba - PR

{jwvflauzino,vfgarcia,gvsouza,elias}@inf.ufpr.br

Abstract. *The NFV Management and Orchestration architecture (NFV-MANO) consists of specifications for the management and orchestration of Virtualized Network Functions (VNFs). Curiously, most NFV developments have been done on the OpenStack cloud platform and, in particular, the Tacker project. The CloudStack platform has been barely used in the context of NFV-MANO. In this work we propose Vines (Vines Is an NFV-MANO Extensible Solution), a new solution integrated to the CloudStack platform to deploy and manage VNFs. A VNF Manager (VNFM) was developed, that supports the execution of basic lifecycle operations (deployment, update, and removal) as well as automatic failure recovery and vertical scaling. Vines has an advantage when compared to Tacker, as it allows fine grained internal management of VNFs. Experimental results are presented comparing CloudStack/Vines to OpenStack/Tacker, showing the feasibility of the proposed solution.*

Resumo. *A arquitetura de referência NFV-MANO (NFV - Management and Orchestration) consiste de um conjunto de especificações relacionadas ao gerenciamento do ciclo de vida e orquestração de VNFs (Virtualized Network Functions). Curiosamente, boa parte dos esforços de desenvolvimento em NFV (Network Function Virtualization) são focados na plataforma de nuvem OpenStack e, em especial, no projeto Tacker. O CloudStack, porém, uma das principais plataformas de nuvens, tem sido superficialmente explorado no contexto do NFV-MANO. Este trabalho propõe o Vines (Vines Is an NFV-MANO Extensible Solution), uma solução integrada à plataforma CloudStack para implantação e gerenciamento do ciclo de vida de VNFs, o que inclui desde a instanciação, atualização e remoção de VNFs, até a recuperação automática de falhas e escalabilidade vertical. Para isso, um VNF Manager (VNFM) foi desenvolvido, disponibilizando um conjunto completo de operações de ciclo de vida de VNFs. Além disso, o Vines tem uma vantagem sobre o Tacker, pois suporta a gerência interna de VNFs. Resultados de experimentos comparativos entre o CloudStack/Vines e o OpenStack/Tacker, ressaltam a viabilidade da proposta.*

1. Introdução

A Virtualização de Funções de Rede (NFV - *Network Function Virtualization*) permite que funções tradicionalmente executadas em hardware proprietário sejam implementadas em software e instanciadas como VNFs (*Virtualized Network Functions*) em hardware de

propósito geral, utilizando técnicas de virtualização. Através do paradigma NFV é esperado o aumento de flexibilidade e a diminuição de custos operacionais (OPEX) e de capital (CAPEX) [Martins et al. 2014, Yousaf et al. 2017] para o gerenciamento e operação de redes. Com o objetivo de oferecer especificações que contribuam com o progresso e permitir interoperabilidade no paradigma NFV, o *European Telecommunications Standards Institute* (ETSI) vem propondo a arquitetura NFV-MANO (NFV - *Management and Orchestration*). A arquitetura define um conjunto de especificações de gerenciamento e orquestração de VNFs que servem como base para o desenvolvimento de soluções NFV [ETSI 2014].

Dentre outras especificações, o NFV-MANO define diversos blocos funcionais. Um desses blocos é o VNF *Manager* (VNFM), responsável, principalmente, pelo gerenciamento do ciclo de vida das VNFs. O ciclo de vida envolve ações como a instanciação, atualização e remoção de VNFs. Outras atividades importantes incluem o ajuste automático de recursos computacionais e a detecção e recuperação de VNFs falhas [Venâncio et al. 2019]. A arquitetura do NFV-MANO, e em especial o VNFM, é normalmente implementada sobre plataformas de computação em nuvem. Isso ocorre já que essas plataformas são elementos facilitadores para NFV, pois fornecem métodos de orquestração e gerenciamento de recursos virtualizados, além de permitirem a automatização de atividades de gerenciamento [Chiosi et al. 2012].

Diversas plataformas para NFV-MANO, como OSM (Open Source MANO) [ETSI 2020], OpenBaton [OpenBaton 2020] e OPNFV [OPNFV 2020a], são majoritariamente baseadas em OpenStack e, em particular o Tacker [Tacker 2020], que é um projeto do OpenStack. O Tacker, orquestrador e gerente de VNFs amplamente utilizado, não possui suporte nativo a VNF *Packages* - VNFP (que poderia ser traduzido como Pacotes VNF), utilizados para armazenar todos os arquivos necessários para a instanciação e gerenciamento de uma VNF (*e.g.*, VNF *Descriptor* - VNFD, scripts de gerenciamento do ciclo de vida da função de rede e imagens de software). Consequentemente, o Tacker realiza o gerenciamento apenas das instâncias virtuais, negligenciando operações como instalação, inicialização de parada da própria função de rede (inicialização de processos e aplicações) internamente a uma VNF. Essas limitações levam, em último caso, a problemas na implementação e adoção de NFV em infraestruturas de nuvem já existentes, gerando um entrave para a popularização do paradigma. A incapacidade de se comunicar e controlar uma VNF internamente (mesmo que existam facilitadores durante a configuração inicial) gera a necessidade de ações extras por parte dos operadores de rede para a conclusão da instanciação e gerenciamento de VNFs, tornando o processo apenas semi-automatizado.

O Apache CloudStack é uma plataforma em nuvem com posição de destaque, sendo a segunda plataforma de código aberto mais adotada no ano de 2019 para nuvens privadas, ocupando o sétimo lugar quando são consideradas tanto as plataformas de código aberto como proprietárias [Flexera 2019]. Sua lista de usuários conhecidos atualmente inclui importantes organizações que a utilizam para criar suas próprias nuvens privadas ou para integração de sistemas [ASF 2020a]. No Brasil, o CloudStack também possui grande relevância, sendo, por exemplo, adotado pela RNP (Rede Nacional de Ensino e Pesquisa), para prover serviços como o Compute@RNP [RNP 2020], que atende diversas instituições governamentais e universidades brasileiras.

Tendo em vista a inquestionável relevância do CloudStack, este trabalho tem como objetivo apresentar a arquitetura, implementação e avaliação do Vines (Vines *Is a* NFV-MANO *Extensible Solution*), um VNFM para CloudStack projetado de acordo com a arquitetura NFV-MANO. O Vines é capaz de gerenciar o ciclo de vida de VNFs, de modo a instanciar, atualizar e remover VNFs em nuvens CloudStack, bem como, instalar, inicializar e parar a função de rede propriamente dita, internamente à VNF. Além disso, a solução proposta é capaz de realizar processos de escala automática de recursos computacionais e recuperação automática de falhas de VNFs. É possível afirmar que o Vines tem uma vantagem sobre o Tacker, na medida em que suporta a gerência interna de VNFs. Resultados de experimentos comparativos entre o CloudStack/Vines e o OpenStack/Tacker, demonstram a viabilidade da solução proposta.

O restante do trabalho está organizado da seguinte forma. A Seção 2 apresenta uma breve fundamentação sobre NFV e discute os trabalhos relacionados. Na Seção 3 são descritos os detalhes do VNFM proposto, bem como os mecanismos de recuperação automática de falhas e escalabilidade de VNFs. A avaliação experimental é descrita na Seção 4. Por fim, a Seção 5 apresenta as conclusões e trabalhos futuros.

2. Fundamentação e Trabalhos Relacionados

Nesta seção são apresentados conceitos que fundamentam o paradigma NFV com enfoque na arquitetura de orquestração e gerência NFV-MANO e seus principais componentes. Além disso, também são apresentados e discutidos os principais trabalhos relacionados.

2.1. Introdução ao Paradigma NFV

O paradigma NFV, originalmente proposto pelo *European Telecommunications Standards Institute* (ETSI), oportuniza uma mudança no modelo de implementação de funções de rede, facilitando e flexibilizando seus processos de desenvolvimento, instanciação, provisionamento e gerenciamento [ETSI 2012]. Para isso, funções de rede tipicamente desenvolvidas em hardware através de equipamentos dedicados e proprietários são migradas para um plano de software, sendo estas executadas em ambientes virtualizados providos por servidores de propósito geral. Entre as vantagens da adoção do paradigma NFV estão: redução de CAPEX e OPEX; suporte dinâmico a mudanças de requisitos de usuários e operadores de rede; facilidade de escala e migração; e redução do tempo necessário para a criação, depuração e divulgação de novas funções e protocolos.

A organização arquitetural do paradigma NFV é composta por três blocos principais: Infraestrutura NFV (*NFV Infrastrucutre* - NFVI), Funções Virtualizadas de Rede (*Virtualized Network Functions* - VNF) e Gerência e Orquestração NFV (NFV *Management and Orquestration* - NFV MANO). O NFVI é formado pelos recursos de hardware (*i.e.*, computação, armazenamento e rede) juntos à camada de virtualização que torna possível o uso dos mesmos para a instanciação de VNFs. Já as instâncias de funções de rede implementadas e executadas em um plano de software constituem o bloco de VNFs, nele ocorre o processamento e o encaminhamento de pacotes, quadros e/ou fluxos. Finalmente, o MANO realiza processos de orquestração e gerência de ciclo de vida tanto dos recursos físicos e virtuais, quanto das instâncias de VNFs, atuando diretamente nos demais blocos.

Em especial, o bloco de gerência de funções de rede e orquestração de serviços, o NFV-MANO, é subdividido em três componentes: (I) Gerenciador de Infraestrutura

Virtualizada (*Virtualized Infrastructure Manager* - VIM), que realiza o gerenciamento e a orquestração de recursos virtualizados pertencentes a uma ou mais NFVIs; (II) Gerenciador de VNF (VNF *Manager* - VNFM), encarregado de gerenciar o ciclo de vida das VNFs; e (III) Orquestrador NFV (NFV *Orchestrator* - NFVO), responsável por orquestrar os recursos de um ou mais VIMs e gerenciar o ciclo de vida dos serviços de rede. Outro importante elemento especificado no NFV-MANO é o chamado Catálogo de VNF (VNF *Catalog*). Esse elemento, por sua vez, consiste de um conjunto de repositórios de VNFs denominados Pacotes de VNF (VNF *Packages* - VNFP), utilizados para armazenar todos os arquivos necessários para a instanciação e gerenciamento de uma VNF (*e.g.*, VNFD, *scripts* de gerenciamento do ciclo de vida da função de rede e imagens de software).

A passagem do núcleo dos sistemas de telecomunicações para soluções implementadas inteiramente em software é uma tendência crescente. Esse fenômeno ocorre para acompanhar a dinamicidade e a grande quantidade de tráfego gerada pelas novas tecnologias, como as redes 5G, *Internet of Things* (IoT) e *Big Data*. As características dessas tecnologias demandam serviços flexíveis, escaláveis e inteligentes para a realização das comunicações [Blanco et al. 2017]. O paradigma NFV é apontado como um dos pilares dessa nova era das redes, mas outros paradigmas, como Redes Definidas por software (*Software Defined Network* - SDN [Kreutz et al. 2015]), computação na borda (*i.e.*, *Edge Computing* [Shi and Dustdar 2016] e *Fog Computing* [Vaquero and Rodero-Merino 2014]) também irão prover importantes recursos para desenvolvimento e implantação dessa nova estrutura, atuando de forma conjunta ao NFV.

2.2. Trabalhos Relacionados

Existem diversas soluções que implementam funcionalidades dos módulos de gerenciamento e orquestração de VNFs proposto na arquitetura NFV-MANO. Essas soluções disponíveis atualmente são providas em um modelo de plataforma (OpenStack/Tacker e OpenBaton) ou como projetos NFV que incluem módulos de gerenciamento de funções de rede (*Open Platform for NFV* e *Open Source MANO*). A seguir são discutidas as principais características de cada uma dessas soluções.

O Tacker [Tacker 2020] é uma plataforma projetada especificamente para OpenStack, sendo capaz de executar funções de VNFM e NFVO. As principais funcionalidades implementadas em relação ao VNFM são o gerenciamento do ciclo de vida básico (*e.g.*, instanciação, atualização e remoção), monitoramento, escalonamento e simplificação da configuração inicial de VNFs. Uma segunda plataforma, chamada OpenBaton [OpenBaton 2020], também implementa os componentes do NFV-MANO, dentre eles um VNFM genérico capaz de gerenciar o ciclo de vida de VNFs com base em descritores contidos em VNF *Packages*, possuindo também adaptadores para os VNFMs Juju e Docker. Através de *drivers* referente ao (VIM), o OpenBatom também oferece suporte a utilização de OpenStack, Amazon AWS e Docker Swarm.

O projeto *Open Platform for NFV* (OPNFV) [OPNFV 2020a] agrega um conjunto de tecnologias relacionadas ao paradigma NFV disponíveis exclusivamente em código aberto, buscando integrar, testar, medir e melhorar cada um dos componentes. O OPNFV oferece compatibilidade com OpenStack e o orquestrador de contêineres Kubernetes [OPNFV 2020b], além de utilizar o Tacker como VNFM, herdando todas suas características. Já o *Open Source MANO* (OSM) [ETSI 2020] é um projeto ETSI destinado a prover uma pilha de software que implementa os componentes do NFV-MANO.

Seu VNFM é estruturado para operar de forma genérica, possibilitando a integração com outros VNFMs específicos. Entretanto, essa integração de VNFMs exige modificações em seu orquestrador de serviços para garantir compatibilidade total.

3. Disponibilizando Suporte para NFV em Nuvem CloudStack

Embora atualmente seja possível instanciar e gerenciar manualmente máquinas virtuais (VMs - *Virtual Machines*) que executem funções de rede no CloudStack ou, até mesmo, criar agentes externos particulares que se comuniquem com sua interface nativa para prover alguns dos componentes da arquitetura NFV-MANO, ainda não há soluções nativamente implementadas no CloudStack para desempenhar o papel dos blocos funcionais NFV-MANO. Dessa forma, este trabalho propõem a modelagem e implementação da solução Vines, um VNFM nativo e integrado à plataforma de nuvem CloudStack. A seguir são apresentados os requisitos de implementação, a arquitetura proposta e demais detalhes operacionais da solução desenvolvida.

3.1. Arquitetura

A solução Vines proposta neste trabalho implementa um VNFM capaz de gerenciar o ciclo de vida de VNFs, executando-as em NFVIs orquestradas pelo CloudStack. A Figura 1 apresenta uma visão geral do CloudStack já com o VNFM proposto. Os módulos tradicionais do CloudStack, a própria plataforma de nuvem, podem ser compreendidos como o bloco operacional VIM do NFV-MANO, destinado a gerenciar os recursos computacionais das NFVIs disponíveis. Assim, o Vines estende as capacidades da plataforma CloudStack, que passa também a atuar como um VNFM, abstraindo detalhes de virtualização e provendo uma interface de alto nível para o gerenciamento de funções de rede.

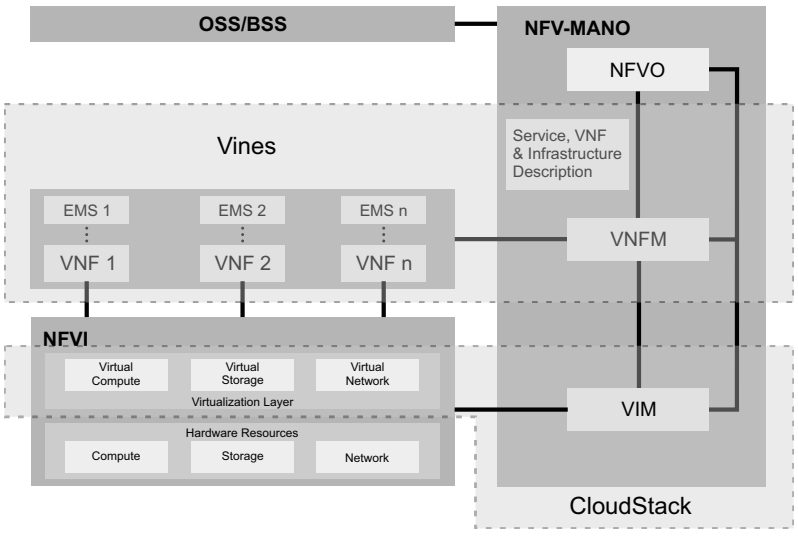

Figura 1. O CloudStack/Vines e os componentes do NFV-MANO.

O CloudStack possui diversos arranjos da sua arquitetura que podem ser escolhidos durante a implantação do mesmo, de acordo com as demandas do usuário [ASF 2020b]. A Figura 2 apresenta os principais elementos do CloudStack utilizados para implantar o ambiente NFV proposto em um cenário simples. Esse cenário consiste

de um nodo de gerenciamento único e múltiplos nodos de computação. A solução Vines está contida dentro da principal aplicação do CloudStack, chamada de CloudStack *Management*, que é executado por uma máquina física denominada *Management Server*. Assim, o acesso ao Vines, por parte dos clientes, é realizado através da mesma interface nativa da plataforma CloudStack. Junto ao *Management Server*, a solução Vines mantém um VNF *Catalog* composto por um conjunto de VNFPs. Os VNFPs agrupam descritores contendo requisitos de implantação de VNFs e *scripts* relacionados às ações de gerenciamento do seu ciclo de vida.

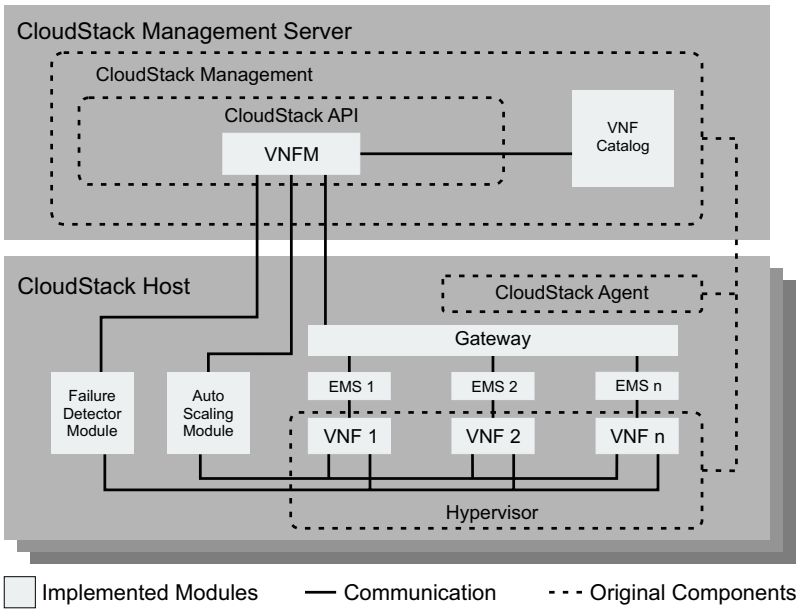

Figura 2. Arquitetura do VNF Manager para CloudStack.

O gerenciamento do ciclo de vida básico das VNFs (*i.e.*, instanciação, atualização e remoção) é realizado pelo VNFM através da comunicação pré estabelecida entre as aplicações CloudStack *Management* e CloudStack *Agent*. As demais ações de gerenciamento do ciclo de vida das VNFs são efetuadas através da comunicação entre o VNFM e o *Gateway*. Assim, ao instanciar uma VM destinada a executar uma VNF, o VNFM encaminha (*push*) o VNFP da função de rede escolhida para a VM hospedeira, passando a ser capaz de executar ações como: instalação (aplicação); configuração; inicialização; e paralisação.

Opcionalmente, através da configuração de uma política de monitoramento no VNFD presente no VNFP, o VNFM pode acionar o módulo de detecção de falhas para monitoramento do estado da VNF. Desse modo, no caso de uma ocorrência de falha por parada, um processo de recuperação automática da VNF (*i.e.*, recriá-la) é iniciado. Um processo semelhante é realizado para o acionamento do módulo de escalabilidade vertical. Entretanto, nesse caso em particular, o VNFM realiza o monitoramento de métricas relacionadas ao consumo de recursos computacionais da VNF (*e.g.*, CPU e memória) e, de acordo com políticas pré definidas, executa a atualização de recursos computacionais disponibilizados à VNF.

3.1.1. Modelo de VNFP

Um requisito fundamental da arquitetura NFV-MANO é que o VNFM seja capaz de gerenciar VNFs heterogêneas e com requisitos específicos, independentemente de sua implementação [ETSI 2014]. Assim, o VNFP é responsável por agrupar todos os elementos necessários para tal gerenciamento, além de mapeá-los de forma padronizada, de modo que o VNFM reconheça cada elemento e possa utilizá-los de forma adequada em cada operação.

O NFV-MANO estabelece o uso da especificação TOSCA (*Topology and Orchestration Specification for Cloud Applications*) para a descrição dos elementos NFV e as conexões entre eles [ETSI 2014]. Desta forma, o modelo padrão de VNFP para CloudStack foi definido através de um TOSCA YAML *Cloud Service Archive* (CSAR) [ETSI 2018]. O VNFP adotado contém um diretório *TOSCA-Metadata* com o arquivo *TOSCA.meta*, no qual estão contidas definições usadas como entrada para analisar o restante do conteúdo do arquivo CSAR (subdiretórios, VNFD, *scripts*, binários, entre outros).

Através do modelo de VNFP adotado, o VNFM é capaz de efetuar operações básicas de gerência do ciclo de vida de VNFs e outras mais específicas como instalação, inicialização e paralisação da função de rede (aplicação) internamente à VNF. Além disso, o VNFM implementado suporta a inclusão de VNFPs constituídos de um arquivo compactado em formato *zip* ou repositórios Git, desde que estejam de acordo com a especificação CSAR.

3.1.2. Gerenciamento Interno de VNFs

Para realizar operações internamente às VNFs, o Vines conta com um EMS presente em cada VNF instanciada. Este EMS, projetado e implementado especificamente para operar com o Vines, é capaz de interpretar requisições de operações de gerenciamento emitidas pelo VNFM e executá-las em sua respectiva VNF. Para cada ação, o EMS executa um *script* (preparado pelo desenvolvedor da função de rede) que está incluso no VNFP armazenado na VNF. Assim, cada *script* executa uma sequência de instruções, realizando atividades como compilação de código fonte, instalação de aplicações, inicialização de aplicações, dentre outras.

Através do modelo de VNFP adotado e o EMS desenvolvido, o VNFM do Vines é capaz de ir além das operações básicas de gerência do ciclo de vida de VNFs, podendo efetuar ações mais específicas como instalação, inicialização e paralisação da função de rede (aplicação) internamente à VNF.

Todas as funções de gerenciamento interno das VNFs são realizadas pelo VNFM em conjunto com o EMS de cada VNF. Por isso, independentemente de como a VNF foi desenvolvida (linguagem de programação usada e outras especificidades), desde que seja disponibilizada corretamente em um modelo de VNFP CSAR como foi definido na Seção 3.1.1, o Vines deverá ser capaz de gerenciá-la de forma satisfatória. Isso porque, todos os pré-requisitos devem estar contidos no VNFP.

Para exemplificar, suponha que uma função de rede hipotética foi implementada em linguagem Java e disponibilizada a partir do código fonte (não compilado). Neste

caso, o VNFP deve conter todo o código fonte da aplicação e o *script* de instalação deve possuir instruções como: (I) instalar a JVM (*Java Virtual Machine*); (II) instalar um compilador Java; (III) instalar possíveis dependências; (IV) compilar o código fonte; e (V) instalar a aplicação principal (criar diretórios, mover arquivos, etc.). Todavia, situações como essa devem ser evitadas, pois pode gerar uma sobrecarga desnecessária na VNF e maior tempo de execução das operações de gerenciamento. Por isso, é recomendado que estes passos sejam simplificados. No caso do exemplo anterior, se a aplicação tivesse sido disponibilizada através de binários (já compilada), dois passos seriam eliminados (obrigatoriamente II e IV, em alguns casos até o passo III).

3.1.3. Detecção e Recuperação de Falhas de VNFs

A recuperação automatizada de falhas é importante para garantir a disponibilidade do serviço em ambientes NFV. Além disso é possível argumentar que simplifica a operação e manutenção da infraestrutura virtualizada, contribuindo com a redução de OPEX [Niwa et al. 2015]. Nesse sentido, a solução Vines implementa um mecanismo capaz de recuperar automaticamente VNFs que apresentem falha por parada (*crash*). Assim, o operador de rede é capaz de, ao instanciar uma VNF, requisitar ao VNFM que a VNF seja monitorada pelo detector de falhas após instanciada.

O módulo de detecção de falhas é composto por três componentes principais: *Failure Detector Agent* (FDA); *Recovery Agent* (RA); e uma base de dados compartilhada entre estes componentes que armazena informações de estado das VNFs. Essa base separa as VNFs monitoradas em três conjuntos distintos de acordo com os estados recuperados durante o monitoramento, sendo eles *alive*, *suspected* e *recovering*. A Figura 3 apresenta a arquitetura do detector de falhas proposto.

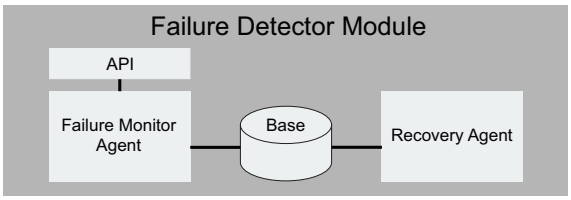

Figura 3. Arquitetura módulo de detecção e recuperação de falhas de VNFs.

O componente FDA é responsável por monitorar VNFs de modo a detectar falhas por parada. Esse componente possui uma API utilizada pelo VNFM para comunicar-se com o FDA, indicando ações de início ou parada do monitoramento de uma determinada VNF. Já o RA tem a tarefa específica de recuperar VNFs que foram apontadas como falhas pelo FDA.

O fluxo de operação do detector de falhas inicia-se após a conclusão do processo de instanciação de uma VNF que deverá ser monitorada pelo mesmo. O VNFM comunica-se com o FDA através de sua API, acionando-o para monitorar o estado da VNF recém instanciada. Em seguida, o FDA adiciona a nova VNF no conjunto *alive* e passa a monitorar seu estado através da solicitação e recebimento de mensagens de estado. Em caso de falha por parada de uma VNF em *alive*, o FDA a move para o conjunto *suspected*. O RA, por sua vez, inspeciona o conjunto *suspected* periodicamente em busca

de novas VNFs consideradas falhas pelo FDA. Quando encontra uma nova VNF neste conjunto, o RA inicia o processo de recuperação, movendo a VNF para o conjunto *recovering*. Dessa forma, uma VNF em estado de recuperação é mantida em *suspected* e *recovering* simultaneamente até o instante em que o RA identifica que a mesma foi recuperada com sucesso. Nesse momento, a VNF recuperada é retirada de *suspected* e *recovering* e é adicionada ao conjunto *alive* novamente, tornando a ser monitorada pelo FDA.

Tanto o FDA quanto o RA permitem a customização dos intervalos de tempo entre a execução de suas atividades através de parâmetros que podem ser alterados em um arquivo de configuração unificado. No caso do FDA, é possível ajustar o intervalo de monitoramento das VNFs, o tempo entre solicitações de mensagens de estado às VNFs, bem como o atraso máximo tolerado para obtenção de resposta das VNFs. Isso permite, respectivamente, a variação do tempo de detecção de falhas, do uso de banda para trocas de mensagens e da precisão na detecção. Em relação ao RA, é possível customizar o tempo entre cada verificação durante o processo de recuperação de uma VNF falha. Essas configurações possibilitam a adaptação dos mecanismos de detecção e recuperação de acordo com a demanda de cada ambiente de rede.

Atualmente o módulo de monitoramento do Vines é capaz de recuperar VNFs através da reinstanciação (*respawn*). Além disso, as VNFs são consideradas *stateless*, portanto, informações de estado não são mantidas.

3.1.4. Escalabilidade Automática de VNFs

Um importante benefício esperado através do paradigma NFV consiste da redução de CAPEX. Parte dessa redução pode ser atribuída à agilidade oferecida para a realizar a escala de recursos computacionais atribuídos às VNFs, proporcionando assim melhor aproveitamento da infraestrutura, de forma a garantir também que as VNFs tenham os recursos que necessitam. A escala pode ser feita tanto de forma horizontal quanto vertical. A escala horizontal consiste da criação de réplicas da VNF e o balanceamento da carga de trabalho entre elas. A escala vertical prevê a adição de recursos computacionais (*e.g.*, CPU, memória, disco, etc.) à VNF [Yazdanov and Fetzer 2012].

A solução Vines inclui um módulo nativo para a realização automática de escala vertical. A Figura 4 apresenta a arquitetura do módulo de escala automática, sendo este composto por três componentes: *Metrics Collector & Policy Analyser* (MCPA); *Scaling Agent* (SA); e uma base de dados compartilhada entre MCPA e SA.

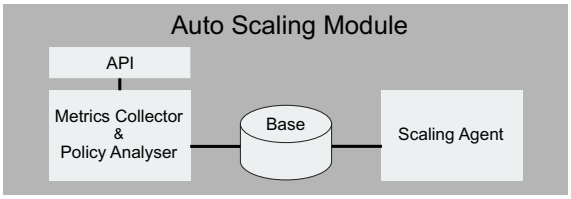

Figura 4. Arquitetura do módulo de escalabilidade automática de VNFs.

O MCPA é o módulo responsável por monitorar o uso de recursos computacionais das VNFs e analisar se a utilização está de acordo com as políticas pré definidas.

Já o módulo SA é responsável por escalar VNFs que ultrapassaram o limite aceitável de utilização de recursos durante um determinado período de tempo (estipulado pelas políticas). O funcionamento do módulo de escala automática ocorre da seguinte maneira: após instanciar uma VNF que contém políticas de elasticidade em seu descritor, o VNFM informa ao MCPA, através de sua API, que essa VNF deve ser adicionada em sua lista de monitoramento, denominada *monitoring*. Assim, o MCPA verifica periodicamente o consumo de recursos computacionais das VNFs contidas em *monitoring*, comparando as métricas atuais de cada VNF às políticas pré estabelecidas. Ao detectar a necessidade de escala, o MCPA move a VNF do conjunto *monitoring* para o conjunto *to_scale*. O SA, por sua vez, verifica regularmente se há uma novidade no conjunto *to_scale* e, quando identifica uma nova VNF, a adiciona no conjunto *scaling* e inicia o processo de escala de recursos. Após a conclusão da escala, o SA remove a VNF dos conjuntos *to_scale* e *scaling* e a insere novamente em *monitoring*.

Entre as customizações possíveis neste módulo estão a modificação do intervalo de tempo de coleta e análise das métricas pelo MCPA, bem como o intervalo que o SA busca por VNFs a serem escaladas. A variação desses valores influencia diretamente no tempo de detecção da necessidade de escala, atraso do início do processo de escala e uso de banda para atividades de monitoramento das métricas. Assim, ajustes podem ser realizados para atender a requisitos específicos de cada ambiente e usuário.

4. Avaliação Experimental

Nesta seção são descritos os cenários de teste e apresentados os resultados comparativos entre as plataformas CloudStack/Vines e OpenStack/Tacker. Cada plataforma dispõe de uma máquina física com processador Intel(R) Core(TM) i7-6700 @ 3.40 GHz com 4 núcleos e 8 GB RAM DDR3L 1600MHz. Em ambos os casos, o sistema operacional subjacente consiste de um Ubuntu 16.04 Server, além de KVM como virtualizador padrão. Os experimentos foram configurados e executados através de *scripts* Python. Esses *scripts* se comunicam com a API de gerenciamento de ambas as plataformas para avaliar operações de ciclo de vida (instanciação e remoção) e gerenciamento de VNFs (recuperação de falhas). Todos os experimentos foram executados 30 vezes e os resultados são apresentados como médias com um intervalo de confiança de 95%

O primeiro experimento tem por objetivo avaliar o tempo de execução de duas operações do ciclo de vida básico de VNFs: instanciação e remoção. Uma VNF padrão foi configurada para instanciação em ambas as plataformas (512 MB RAM, 1 núcleo virtual, 2 interfaces de rede virtual). O tempo de instanciação apresentado representa a duração em segundos entre o instante em que a chamada de instanciação é executada, até o momento no qual a VNF responde pela primeira vez com uma mensagem de estado. Já o tempo de remoção é composto pela duração entre o instante da execução da chamada até o fim do processo de remoção da VNF. Os resultados dos testes descritos são apresentados na Figura 5.

Os resultados apontam que, para o experimento realizado, o CloudStack/Vines é capaz de instanciar uma VNF em um tempo médio de 31,92 segundos, sendo, em média, 13,76 segundos mais rápido que o OpenStack/Tacker, que realiza a instanciação de uma VNF com um tempo médio de 45,68 segundos. Em relação a operação de remoção, os resultados indicam um melhor desempenho do OpenStack/Tacker, que leva em média

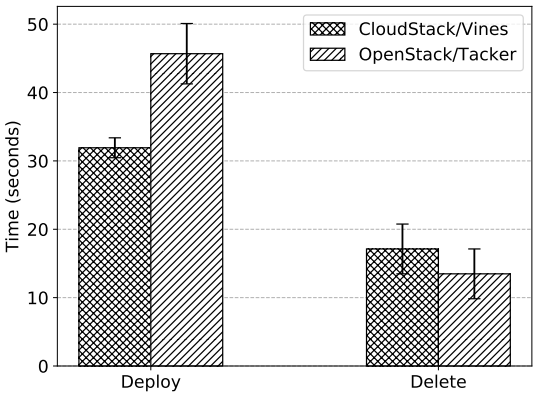

Figura 5. Tempo de instanciação e remoção.

13,48 segundos, enquanto o CloudStack/Vines executa a operação em 17,12 segundos. Os principais fatores de influência nos resultados de instanciação e remoção de VNFs são, respectivamente, o tempo de instanciação e de remoção das VMs, sendo estes tempos inerentes às plataformas de nuvem, não aos VNFMs.

Uma métrica importante relacionada à eficiência de mecanismos de recuperação de falhas é o MTTR (*Mean Time To Recover*), que representa o tempo médio de recuperação [Grover and Sack 2007, Candea et al. 2002]. Nesse sentido, o segundo experimento realizado visa medir o MTTR do módulo de detecção e recuperação de VNFs falhas das plataformas CloudStack/Vines e OpenStack/Tacker. Neste experimento, VNFs com as mesmas características são instanciadas em ambas plataformas com políticas de monitoramento equivalentes, as quais definem um intervalo de monitoramento de 5 segundos e *timeout* de 2 segundos para requisições de mensagens de estado. Falhas por parada foram simuladas através da desativação da interface de rede de gerenciamento da VNF, fazendo com que a mesma não responda às requisições de mensagens de estado. Os resultados são apresentados na Figura 6.

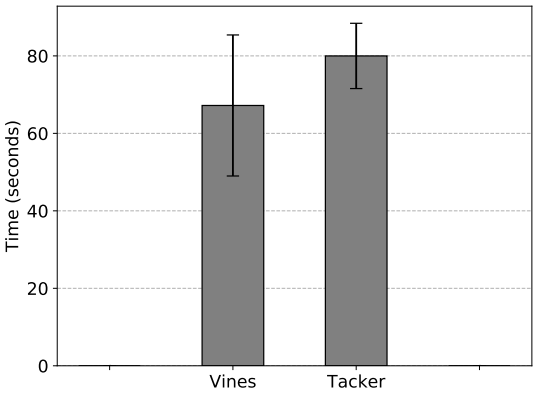

Figura 6. Tempo de recuperação.

O primeiro resultado importante a ser destacado é que em todos os experimentos com o detector de falhas, ambas as plataformas foram capazes de detectar e recuperar a VNF que apresentou a falha por parada. Os resultados demonstram que o CloudS-

tack/Vines foi capaz de recuperar a VNF falha em um tempo médio de 67,20 segundos, enquanto que o OpenStack/Tacker levou um tempo maior, sendo, em média, 79,99 segundos. A recuperação em menor tempo feita pelo Vines é decorrente do tempo mais curto de instanciação como foi apresentado na Figura 5, tendo em vista que para recuperar uma VNF ambos VNFMs precisam removê-la e em seguida instanciá-la novamente.

Como mencionado na seção anterior, o Vines possui um módulo de elasticidade vertical. A abordagem vertical elimina alguns desafios encontrados na escala horizontal, como por exemplo, o balanceamento de carga entre as instâncias que representam a mesma VNF. No entanto, o Tacker oferece suporte apenas a escalabilidade horizontal, o que inviabiliza uma comparação direta com o Vines neste quesito.

Um módulo de escalabilidade automática realiza seu monitoramento baseado em uma política que define diversos parâmetros, como por exemplo, limites máximos e mínimos de uso de recursos computacionais (CPU, memória, disco, etc.) durante determinado período. Assim, o tempo decorrido para detectar a necessidade de realizar uma escala, varia de acordo com as especificações de cada política de monitoramento. Por outro lado, o tempo médio de execução da escala em si tende a ser mantido independentemente da política aplicada. Por conta disso, a avaliação do módulo de elasticidade do Vines foi realizada através da análise do tempo médio para a execução da escala.

No experimento conduzido, uma VNF foi instanciada com a indicação através dos parâmetros adequados que necessitava de monitoramento do uso de recursos. Em seguida era provocado um uso excessivo de memória através da ferramenta *stress*, de modo a exceder os limites definidos na política de monitoramento. Assim, após ocorrer o uso excessivo de recursos de forma contínua durante o período definido na política, o mecanismo de elasticidade do Vines inicia o procedimento de escala da VNF. Para executar a operação de escala, o Vines faz a parada da VNF, altera os recursos virtuais disponíveis à VNF (no CloudStack isso é tratado como oferta de serviço) e inicializa a VNF.

Como resultado, o Vines demonstrou ser capaz de escalar todas as VNFs que excederam o consumo de recursos definidos nas políticas, levando em um tempo médio de 58,91 segundos para cada escala. Isso representa uma duração razoável para a operação de escala de recursos, tendo em vista que, para recuperar uma VNF falha, o módulo de recuperação de falhas do Vines leva em média 8,29 segundos a mais de tempo do que a escala.

5. Conclusão

Este trabalho apresentou uma proposta de disponibilização na plataforma Apache CloudStack do paradigma NFV que a ETSI vêm propondo através da arquitetura NFV-MANO. O Vines, um VNFM, foi projetado e implementado de forma integrada ao CloudStack, sendo capaz de gerenciar o ciclo de vida completo de VNFs, realizando desde operações básicas como instanciação, atualização e remoção de VNFs, até atividades como o controle básico da função de rede em cada VNF, recuperação de falhas e elasticidade automática. Para isso, o Vines conta com o suporte a VNF *Packages* CSAR, que contém os elementos necessários para o gerenciamento de VNFs, tais como binários da função de rede, *scripts* e descritores que definem as características da VNF, bem como políticas de monitoramento de falhas e de uso de recursos computacionais. Devido a esta carac-

terística, a granularidade das operações de gerência do Vines é mais fina que, por exemplo, a do OpenStack/Tacker.

Com a introdução do CloudStack ao contexto NFV-MANO, surge um conjunto de novas oportunidades a serem exploradas. Em relação a alocação eficiente de recursos, pretende-se como trabalhos futuros investigar o suporte a escala vertical em tempo de execução através do uso de outros virtualizadores, bem como a elasticidade horizontal. Outra importante linha de investigação futura é a evolução do Vines para operar também como NFVO, de modo a realizar a composição de serviços de rede através do encadeamento das funções virtualizadas de rede (*Service Function Chaining* - SFC).

Na página[1] do projeto Vines estão disponibilizados código fonte, instruções de instalação, termos de licença, dentre outros recursos.

Referências

ASF (2020a). Apache cloudstack users. Technical report, Apache CloudStack: Open Source Cloud Computing. `http://cloudstack.apache.org/users.html` Acessado: Mar. 2020.

ASF (2020b). Choosing a deployment architecture. Technical report, Apache CloudStack: Open Source Cloud Computing. `http://docs.cloudstack.apache.org/en/latest/conceptsandterminology/choosing_deployment_architecture.html` Acessado: Mar. 2020.

Blanco, B., Fajardo, J. O., Giannoulakis, I., Kafetzakis, E., Peng, S., Pérez-Romero, J., Trajkovska, I., Khodashenas, P. S., Goratti, L., Paolino, M., et al. (2017). Technology pillars in the architecture of future 5g mobile networks: Nfv, mec and sdn. *Computer Standards & Interfaces*, 54:216–228.

Candea, G., Cutler, J., Fox, A., Doshi, R., Garg, P., and Gowda, R. (2002). Reducing recovery time in a small recursively restartable system. In *Proceedings international conference on dependable systems and networks*, pages 605–614. IEEE.

Chiosi, M., Clarke, D., Willis, P., Reid, A., Feger, J., Bugenhagen, M., Khan, W., Fargano, M., Cui, C., Deng, H., et al. (2012). Network functions virtualisation: An introduction, benefits, enablers, challenges & call for action. Technical report, European Telecommunications Standards Institute.

ETSI (2012). Network functions virtualization: White paper. Technical report, European Telecommunications Standards Institute.

ETSI (2014). Network functions virtualisation (nfv): Management and orchestration. Technical report, European Telecommunications Standards Institute.

ETSI (2018). Network functions virtualisation (nfv) release 2; protocols and data models; vnf package specification. Technical report, European Telecommunications Standards Institute.

ETSI (2020). Osm virtual infrastructure managers. Technical report, European Telecommunications Standards Institute. `https://osm.etsi.org/wikipub/index.php/VIMs` Acessado: Mar. 2020.

[1]Disponível em http://www.inf.ufpr.br/jwvflauzino/vines

Flexera (2019). State of the cloud report from flexera. Technical report, Flexera: IT Management Software, Optimization & Solutions.

Grover, W. D. and Sack, A. (2007). High availability survivable networks: When is reducing mttr better than adding protection capacity? In *2007 6th International Workshop on Design and Reliable Communication Networks*, pages 1–7. IEEE.

Kreutz, D., Ramos, F. M. V., Veríssimo, P. E., Rothenberg, C. E., Azodolmolky, S., and Uhlig, S. (2015). Software-defined networking: A comprehensive survey. *Proceedings of the IEEE*, 103(1):14–76.

Martins, J., Ahmed, M., Raiciu, C., Olteanu, V., Honda, M., Bifulco, R., and Huici, F. (2014). Clickos and the art of network function virtualization. In *Symposium on Networked Systems Design and Implementation*, pages 459–473. USENIX.

Niwa, T., Miyazawa, M., Hayashi, M., and Stadler, R. (2015). Universal fault detection for nfv using som-based clustering. In *Asia-Pacific Network Operations and Management Symposium*, pages 315–320. IEEE.

OpenBaton (2020). Vim drivers. Technical report, 5G Berlin Project. `https://openbaton.github.io/documentation/vim-driver` Acessado: Mar. 2020.

OPNFV (2020a). Open platform for nfv. Technical report, The Linux Foundation. `https://www.opnfv.org` Acessado: Mar. 2020.

OPNFV (2020b). Virtual infrastructure management. Technical report, The Linux Foundation. `https://docs.opnfv.org/en/stable-hunter/release/overview.html#virtual-infrastructure-management` Acessado: Mar. 2020.

RNP (2020). Compute. Technical report, Rede Nacional de Ensino e Pesquisa. `https://www.rnp.br/servicos/gestores-de-ti/hospedagem-e-armazenamento/compute` Acessado: Mar. 2020.

Shi, W. and Dustdar, S. (2016). The promise of edge computing. *Computer*, 49(5):78–81.

Tacker (2020). Tacker - openstack nfv orchestration. Technical report, The OpenStack Project. `https://wiki.openstack.org/wiki/Tacker` Acessado: Mar. 2020.

Vaquero, L. M. and Rodero-Merino, L. (2014). Finding your way in the fog: Towards a comprehensive definition of fog computing. *SIGCOMM Comput. Commun. Rev.*, 44(5):27–32.

Venâncio, G., Garcia, V. F., da Cruz Marcuzzo, L., Tavares, T. N., Franco, M. F., Bondan, L., Schaeffer-Filho, A. E., Paula dos Santos, C. R., Granville, L. Z., and P. Duarte Jr, E. (2019). Beyond vnfm: Filling the gaps of the etsi vnf manager to fully support vnf life cycle operations. *International Journal of Network Management*.

Yazdanov, L. and Fetzer, C. (2012). Vertical scaling for prioritized vms provisioning. In *International Conference on Cloud and Green Computing*, pages 118–125. IEEE.

Yousaf, F. Z., Bredel, M., Schaller, S., and Schneider, F. (2017). Nfv and sdn—key technology enablers for 5g networks. *IEEE Journal on Selected Areas in Communications*, 35(11):2468–2478.