Proposta de Vim *On-Demand* para fatiamento de Nuvem

Eduardo H. R. Zanela[1], Cesar A. C. Marcondes[2], Fábio L. Verdi[1]
Alex M. G. de Almeida[3], Emerson R. A. Barea[1]

[1]Departamento de Computação – UFSCar

[2]Instituto Tecnológico da Aeronáutica – ITA

[3]Faculdade de Tecnologia de Ourinhos – FATEC

eduardo-hnq@hotmail.com

{marcondes,verdi,emerson.barea}@ufscar.br

alex.marino@fatecourinhos.edu.br

Abstract. *This paper presents a proposal for a Virtualized Infrastructure Manager On-Demand for cloud slicing that allows bare metal control and network services with focus in on-demand creation. For this, a management system was implemented using Openstack in conjunction with virtualization tools allowing the instantiation of a physical machine. In this paper, we present 3 options of installation: complete instantiation bare metal, complete instantiation with containers and complete instantiation with preloaded image. Our results indicate that the proposed, implemented, orchestrated and tested architecture meets the requirements of VIM On-Demand, and points out the main prevailing points in VIM.*

Resumo. *Este trabalho apresenta uma proposta de VIM Sob-Demanda para fatiamento de nuvens que permita controle bare metal e de serviços de redes, com enfoque na sua criação sob demanda. Para isto, foi implementado um sistema de gerenciamento utilizando o Openstack em conjunto com ferramentas de virtualização, permitindo a instanciação de uma máquina física. Neste artigo, apresentamos 3 opções de instalação: instanciação completa bare metal, instanciação completa com containers e instanciação completa com imagem pré-carregada. Nossos resultados indicam que a arquitetura proposta, implementada, orquestrada e testada atende os requisitos do VIM On-Demand, e apontam os principais pontos preponderantes no VIM.*

1. Introdução

Um dos maiores desafios enfrentados no desenvolvimento da rede 5G é atender às diversas demandas de consumidores por novos serviços, modificando a forma de implantação de redes de tamanhos fixos para tamanhos variáveis, possibilitando atender aos requisitos de desempenho específicos quanto à latência, escalabilidade, disponibilidade e confiabilidade, impostos por cada caso de uso.

Para tal, é necessária a introdução do fatiamento de rede baseado em demandas, que permitirá que cada fatia de rede, denominado como *slice*, seja alocada conforme a demanda e requisitos do locatário em função de suas aplicações. O ponto central da

implantação do fatiamento é dependente da orquestração de atividades que possibilitem a instalação, preparação do ambiente sucedida de suas configurações e instalação dos serviços requeridos. A conjuminação destas tarefas visa, por fim, a elaboração de um aparato de gerenciamento de fatias de redes virtualizadas, as quais adotam anglicismo *Virtualized Infrastructure Manager* (VIM).

A definição de *slice* de rede pode ser dado como redes lógicas executadas em rede física ou virtual, mutuamente isoladas, dotadas de controle e gerenciamento independentes, que podem ser criados sob demanda [Clayman 2017].

Uma das alternativas que vem sendo utilizada para demonstração do conceito apresentado é a virtualização da rede, que possibilita transformar redes reais em redes virtuais, utilizando soluções tecnológicas baseadas, principalmente, em software tais como: *Software Defined Networking* (SDN), *Network Functions Virtualization* (NFV) e *Virtualized Network Function* (VNF). Estas tecnologias concedem capacidade programática à solução, flexibilidade e modularidade para o desenvolvimento de diferentes redes virtuais para diferentes casos de uso.

Dentre os aspectos importantes correlacionados ao desenvolvimento do VIM, são as verificações dos pontos preponderantes relativos à escalabilidade, aferições pertinentes à resiliência do ambiente e constatações dos tempos gastos durante o processamento de cada etapa da instalação. A proposta apresentada neste trabalho aponta para o desafio de um VIM de fatiamento de rede que criem *slices* sob demanda. Espera-se que os *slices* caracterizem leveza, praticidade, facilidade de instalação e configuração, somando-se ainda à capacidade de gerenciamento dos componentes de infraestruturas virtualizadas. Ao término deste trabalho, espera-se responder duas questões: (*i*) Os recursos computacionais, tais como memória principal e quantidade de núcleos são determinantes para a eficiência do processo? (*ii*) É possível aferir vantagem em desempenho da abordagem VIM *on-demand* em relação à tradicional?

No melhor dos nossos esforços, este trabalho apresenta uma proposta de desenvolvimento de um VIM sob demanda (VIM *on-Demand*), no qual será apresentado o desenvolvimento de forma virtualizada e dividido em 3 opções de instalações: Instanciação Completa *Bare Metal*; Instanciação Completa com *Containers* e Instanciação Completa com Imagem Pré-Carregada. Atribui-se como aspecto inovador desta proposta (*i*) o uso de equipamentos de baixo custo, (*ii*) uso exclusivo de tecnologias de código aberto, (*iii*) orquestração total da implementação e (*iv*) instanciação de diferentes casos de uso. Ressalta-se a escolha da utilização do Openstack em função dos seus serviços aderentes ao propósito deste trabalho, que são a existência do *bare metal* e serviços de controle de computação.

O restante desse trabalho está organizado da seguinte maneira: a Seção 2 relaciona os trabalhos correlatos a este. Na Seção 3 são apresentados os principais conceitos envolvidos no trabalho, o aparato tecnológico e a metodologia empregada na execução dos experimento. Na Seção 4 são discutidos os resultados obtidos no rol de experimentos realizados e, por fim, na Seção 5 conclui este trabalho e apresenta sugestões de trabalhos futuros.

2. Trabalhos Relacionados

Apesar do conceito de 5G ser considerado novo, existem diversos trabalhos acadêmicos que colaboram com o seu desenvolvimento, porém o único que utiliza o conceito de *Slice as a Service*, fator central para o provisionamento VIM *on-demand*, é o projeto NECOS[1], no qual este trabalho faz parte [Silva et al. 2018].

Como outros projetos e trabalhos acadêmicos, destacam o 3GPP, o qual apresentou a proposta de padronização do sistema de rede de acesso 5G e do núcleo de sistema de rede 5G ao mesmo tempo [Kim et al. 2017]; o 5G NORMA, que propôs-se uma arquitetura capaz de lidar eficientemente com diversos requisitos e flutuações na procura de tráfego resultantes de conjunto de serviços heterogêneos e suspensos [Rost et al. 2017]; e o SONATA abordou os desafios significativos associados ao desenvolvimento e à implantação dos serviços complexos previstos para as redes 5G [Dräxler et al. 2017].

O trabalho [Freitas et al. 2018], apresenta um sistema capaz de implantar e fornecer VIMs operacionais completos através de um Data Center (DC) *slice Controller*, com base em uma arquitetura em que as fatias do DC são criadas sobre recursos transformáveis (computação e armazenamento). O presente trabalho tem como diferenças do trabalho de [Freitas et al. 2018], a instalação total do ambiente de controle sob demanda analisando diferentes abordagens para carregamento do ambiente. Além disso, entende-se que ao realizar a instalação do ambiente ao invés de importar imagens prontas, mais flexibilidade e opções de customização de tal ambiente podem ser exploradas.

Existem diversos trabalhos voltados à indústria, e para o presente trabalho foram analisados o Kayobe[2], que é um projeto Openstack com finalidade de automatizar a implantação do Openstack em um conjunto de servidores *bare metal*, utilizando *containers*; e o TripleO (Openstack On Openstack)[3] que é um projeto do Openstack que permite implantar duas nuvens, com a finalidade de instalar, atualizar e operar nuvens Openstack utilizando as próprias instalações de nuvem do Openstack como base.

Em relação à característica inovadora nessa criação de *slice*, comparado com a utilização de recursos no OpenStack Nova, compreende-se que o mesmo permite assimilar uma mini-nuvem para o locatário.

3. Materiais e Métodos

Nesta seção tem-se descritos os materiais e métodos utilizados durante o desenvolvimento da proposta; na Subseção 3.1 são apresentados os principais conceitos e definições envolvidos neste trabalho; na Subseção 3.3 o aparato computacional e tecnológico utilizado para realização deste trabalho; na Subseção 3.2 os procedimentos adotados que possibilitaram perfazer as atividades e, por fim, na Subseção 3.4 apresentamos a descrição dos experimentos.

3.1. Conceitos e Definições

Para efetivar o desenvolvimento da proposta de VIM *on-demand* nas condições propostas, alguns requisitos devem ser previamente elencados e discutidos. Os requisitos que

[1]http://www.h2020-necos.eu/.

[2]https://kayobe.readthedocs.io.

[3]https://wiki.openstack.org/wiki/TripleO.

nortearam o processo de implementação, configuração e instanciação do ambiente e experimentos, seguem descritos:

3.1.1. Virtualização

Permite criar uma versão similar de algo, um software adaptável como sistemas operacionais, dispositivos, hardware ou recursos de redes. A computação em nuvem, faz intensa utilização do conceito da virtualização, uma vez que a mesma proporciona uma redução de custo e melhor controle de seu ambiente, segregando a aplicação e sistema operacional dos componentes físicos de origem.

A virtualização dos serviços de rede implica na transformação de suas funções de redes estáticas em funções virtuais. Para [Han et al. 2015], a virtualização das funções de redes propõem melhorar o provisionamento dos serviços de redes. A transformação de redes reais em redes virtuais pressupõe a aplicação de tecnologias baseadas em software, conferindo capacidade de controle, flexibilidade e programação, além de colaborar com o desenvolvimento de diferentes redes virtuais para diferentes casos de uso.

3.1.2. Fatiamento de Rede

Trata-se de uma técnica específica de virtualização, a qual proporciona a implementação de várias redes lógicas sob uma infraestrutura de rede física compartilhada [Mademann 2017]. As redes lógicas dos *slices* atuam de ponta a ponta e são formadas por funções de rede, armazenamento e computação [Freitas et al. 2018].

No fatiamento, a infraestrutura de uma rede é subdividida em partes denominadas *slices*. A *slice* é reservada à uma aplicação de um usuário ou locatário e é considerada como um conjunto de recursos que, combinados com a orquestração, atendem a todos os requisitos de um caso de uso específico [Clayman et al. 2018].

A segregação da infraestrutura em múltiplas partes confere benefícios econômicos, tais como: redução de gastos, remoção das limitações de infraestrutura e tecnologias, expansão do cenário para vários clientes diferentes e menor transmissão de dados na rede, isto porque o tráfego de dados móveis é crescente [Silva et al. 2018].

Um caso de uso que tem como alvo a utilização de nuvens para oferecer serviços e infraestrutura para propósito geral, é composto por três tipos de agentes:

- **Provedor de infraestrutura**: é o provedor que possui e gerencia uma determinada rede física e recursos computacionais, armazenamento e comunicação;
- **Locatário**: aluga os serviços de infraestrutura de rede de um ou mais provedores, sob a forma de rede virtual;
- **Usuário final**: aluga os serviços tanto de infraestrutura de redes ofertados pelos provedores, quanto os serviços fornecidos pelos locatários, porém não consegue fornecer recurso para outros usuários.

Nos *slices*, os recursos são unidades gerenciáveis e são divididos em VNFs e recursos de infraestrutura. Para cada *slice* é necessário que exista um VIM gestor do controle dos recursos, suportando diferentes demandas, denominado como VIM *on-Demand* [Silva et al. 2018].

3.1.3. Slice as a Service

Refere-se a um novo conceito de serviço de rede, cujos recursos são acoplados internamente à *slice* e gerenciados como um todo, podendo ainda conter componentes de serviços independentes de outros *slices* [Ordonez-Lucena et al. 2017].

O *slice* deve ser criada diretamente em ambiente de nuvens, o que proporcionará menor tempo de resposta nas alterações das demandas de serviços. O *Slice as a Service* possui o conceito de princípio leve, o qual exaure ao limite as configurações dos *slices*, recursos físicos e capacidade de gerenciamento de recursos e desempenho por software.

Em uma situação onde os recursos de infraestrutura ficam inutilizados devido ao excesso de recursos, o conceito do *Slice as a Service* torna possível eliminar a ociosidade da rede, criando novos *slices* conforme requisitados pelos clientes, proporcionando diferentes VIMs e funções virtualizadas de rede. Para [Freitas et al. 2018], os *slices* permitem uma visão geral dos mecanismos, componentes, e abstrações que podem ser utilizados para fornecer um modelo de *Slice as a Service* baseado na instanciação VIM dinâmica.

O problema da alocação dos slices possui um forte componente de otimização. Em um sistema com escala maior e com caminhos de rede disjuntos (Figura 1), a escolha de enlaces para serem alocados para cada slice pode definir a quantidade de clientes suportados pela rede, bem como o desempenho da rede para tais clientes. O VIM é um dos principais componentes de uma estrutura virtualizada, cabe-lhe a responsabilidade de manter o controle e gerenciamento dos recursos de computação, armazenamento e infraestrutura de rede [Freitas et al. 2018].

Oposto às tradicionais configurações de um único VIM para toda infraestrutura da computação em nuvem, o VIM *on-Demand* é implantado dinamicamente para cada *slice*, cabendo a ele permitir que a *slice* de rede esteja apta ao atendimento de diferentes casos de uso, independentemente da necessidade de execução de qualquer tipo de tecnologia.

3.2. Procedimentos e Técnicas

Nesta subseção será apresentado um conjunto de tarefas necessários à implantação de um VIM *on-Demand*. A Figura 1 permite visualizar este conjunto de tarefas e suas dependências entre si:

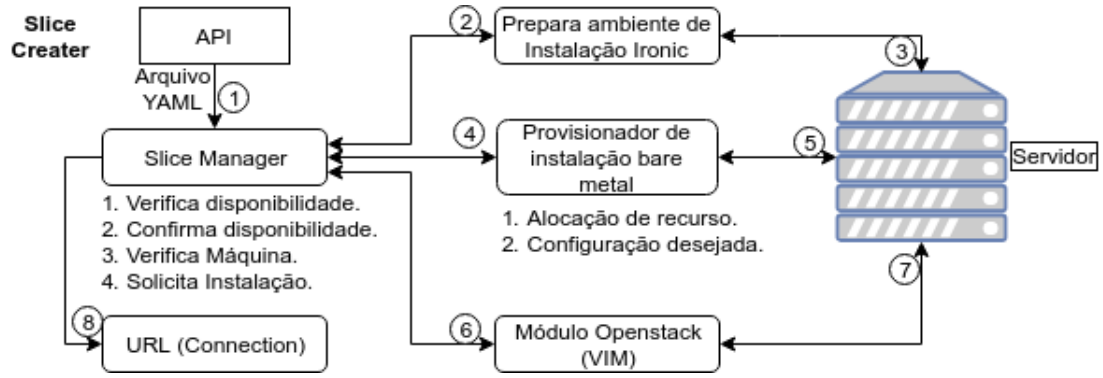


Figura 1. Infraestrutura VIM *On-Demand*.

A seguir serão apresentadas três opções de instalação VIM *on-Demand* proposta

nesse trabalho: Instanciação Completa *Bare Metal*, Instanciação Completa com Imagem Pré-Carregada e Instanciação Completa com *Containers*.

3.2.1. Instanciação Completa *Bare Metal*

Uma completa instanciação de um VIM *on-demand* pode ser considerada uma instalação completa (Figura 1), exigindo que seja realizado todo processo de instalação, sem procedimentos que reduzam o tempo de instalação, necessitando da execução de todos os passos.

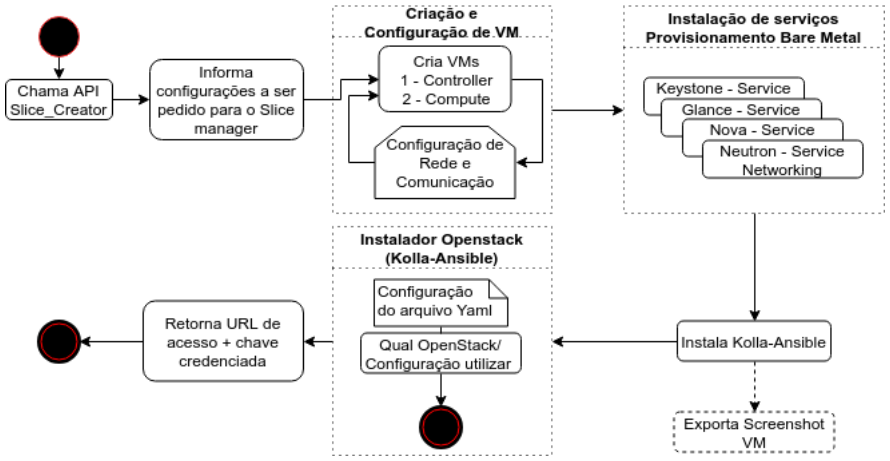


Figura 2. VIM *On-Demand Full Deployment*.

As etapas de instalação são divididas em: Criação de Virtual Machines (VMs), Configuração de rede, Comunicação entre VMs; Instalação Openstack; Instalação do serviço *bare metal* (Ironic); Provisionamento da máquina *bare metal*; Instalação Kolla-Ansible; Provisionamento de serviço Openstack e retorno da URL de acesso.

3.2.2. Instanciação Completa com Imagem pré-carregada

A instanciação de um VIM *on-Demand* com uma imagem pré-carregada, é uma instanciação, na qual parte das etapas encontram-se pré-executadas ou provisionadas da instanciação completa, porém com a máquina em um *snapshot*, necessitando ser importado e reconfigurado. O diagrama representado na Figura 3 traz à luz o arcabouço de tarefas compreendidas.

Ao invés de criar uma nova VM para instalar os controladores de cada *slice*, realizamos o *import* do *snapshot* da VM, que é uma tarefa menos custosa computacionalmente. Com isso, o oneroso trabalho de criação e configuração de VMs, instalação e configuração de serviços de provisionamento *bare metal* é evitado, restando apenas a configuração do serviço de rede da nova VM e a configuração para as próximas máquinas *bare metal* a serem criadas. As etapas restantes para conclusão de uma instalação completa de uma *slice* com seus serviços são a instalação do kolla-ansible e a instalação e configuração do controlador Openstack que serão utilizados.

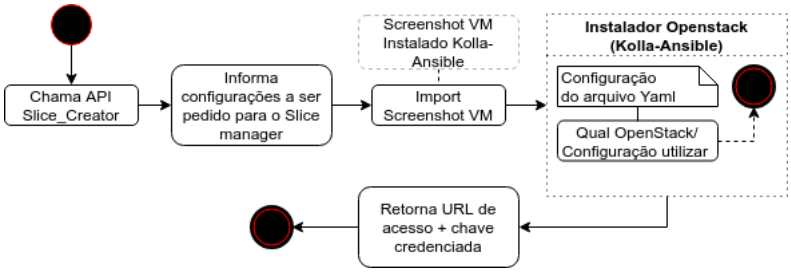


Figura 3. VIM *on-Demand* pré-carregada.

A Figura 3 apresenta passos necessários para a finalização da instalação. A eliminação do grupo de Criação e Configuração de VM e *Install services for bare metal Provisioners* são tarefas que diferenciam os grupos de atividades visualizadas na Figura 2, restando o grupo de instalação Openstack diretamente no Kolla-Ansible.

3.2.3. Instanciação Completa com *containers*

A última instanciação elimina todos os processos de instalação dos subgrupos anteriores: Criação de VM base para os serviços, instalação do Openstack Ironic (*Bare Metal*) e instalação do Openstack Kolla-ansible (Controlador Openstack).

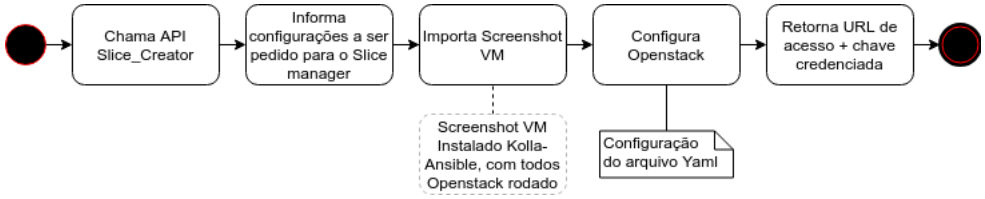


Figura 4. VIM *on-Demand* com *containers*.

Embora as etapas anteriores que pudessem tornar mais confusa e onerosa durante a instalação, esta instanciação simplificada não envolve tarefas de instalação de pacotes ou serviços. Neste método é necessário a importação do *snapshot* pronto, conforme demonstra a Figura 4, e configuração.

3.3. Aparato Computacional

Para realizar a implantação do ambiente foi utilizado o software de virtualização Virtualbox em conjunto com Vagrant, os quais são softwares com finalidade de criação e manutenção de ambientes de desenvolvimento virtualizados . A fim de controlar as máquinas virtuais em nível físico foi utilizado o Virtual BMC.

Com o objetivo de gerenciamento e provisionamento das máquinas físicas foi utilizando o Openstack Ironic. Para instalação de controladores que permitam operar nuvens utilizamos o Openstack Kolla-Ansible, pois este é composto por *containers* prontos para produção e ferramentas de implantação de Openstacks. Para a implementação do VIM *on-Demand* utilizamos o servidor com as especificações descritos na Tabela 1.

Para criação e controle das máquinas virtuais foi utilizado o software Vagrant, versão 2.2.3. Para exportação e importação do ambiente virtualizado foi utilizado o software

Tabela 1. Quadro de configurações do servidor utilizado.

Componente	Dimensão
Processador	i5 3.30GHz 12 núcleos
Disco Rígido	1 terabytes
Memória RAM:	32 Gigabytes
Sistema Operacional	Linux Ubuntu Server 64 bit

Virtualbox, versão 5.2.18. A utilização do software de virtualização Vagrant deve-se às suas características de fácil controle das máquinas, tais como a criação, o controle e a remoção do ambiente virtualizado. O Virtual BMC servi-se para gerenciar e monitorar as VMs, permitindo executar comandos a nível de máquinas *bare metal* reais, utilizando protocolo IPMI. Com o aparato computacional preparado o processo de criação do VIM *on-Demand* pode ser dividido em três partes: *bare metal*, Kolla-Ansible e a Versão do *Controller* Openstack.

3.4. Descrição dos experimentos

Para atender aos requisitos do ambiente proposto, definiu-se um modelo de arquitetura conforme apresentada na Figura 5, onde é possível visualizar as etapas de execução da implantação das tecnologias utilizadas em conjunto com as tarefas de importação e exportação de imagens. É possível observar que as atividades de instalações e configurações foram divididas em 3 ambientes, sejam eles:

- ***Machine Server*:** Máquina Física que recebe a instalação de um sistema e dos softwares responsáveis por criar e controlar um ambiente virtualizado;
- ***Machine Bare Metal*:** Máquina Virtual composta pelos serviços Openstack, e responsável pela instanciação da Máquina Física *Bare Metal*;
- ***Machine Full*:** Máquina Virtual anterior composta também com o Kolla-Ansible e responsável pela instalação dos serviços Openstack na Máquina Física *Bare Metal*.

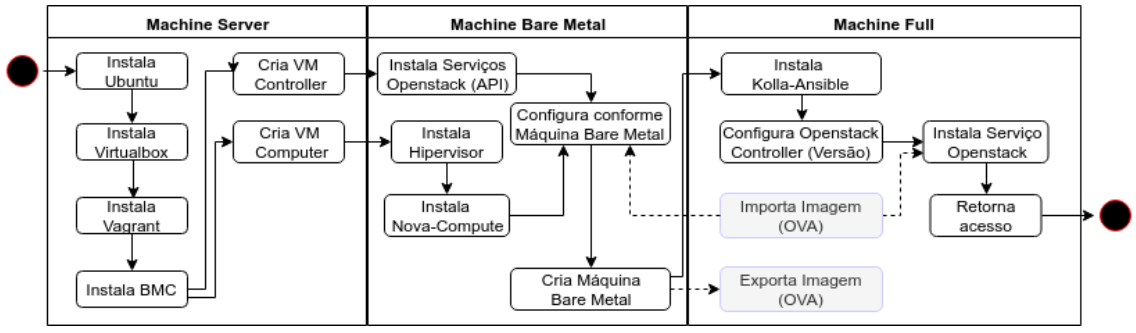


Figura 5. Diagrama de atividades em conjunto com as tecnologias utilizadas em cada VM.

Para a exportação e importação das imagens durante a instanciação com imagem pré-carregada e instanciação completa com *containers*, utilizou-se o comando VBoxManager, o qual é uma interface de linha de comando utilizada para controle do VirtualBox, que permite acessar todos os recursos presentes na interface gráfica do usuário.

A execução do experimentos contemplaram 27 repetições para cada aplicação de VIM *on-Demand* proposto, utilizando cálculos de tomada de tempo para verificação de desempenhos durante sua instanciação. Cada execução completa foi dividida em partes e os seus tempos de execução foram capturados. A execução completa para criação de um VIM *on-Demand* foi subdividida em:

- ***BM***: Representa o tempo dos procedimentos de criação e configuração de VMs (Controller e Computer), instalação do Openstack Ironic, configuração e instanciação da máquina *bare metal*;
- ***KA*** (Kolla-Ansible): Representa o tempo dos procedimentos inicializados após a finalização da criação da máquina *bare metal*, a instalação do Openstack Kolla-Ansible e configuração de ambiente para instalação do controlador Openstack na máquina *bare metal*;
- ***OS*** (Openstack): Representa o tempo do processo de configuração e instanciação da máquina para instalação do serviço Openstack;
- ***TOTAL***: Combinação dos períodos do *Bare Metal* (BAREMETAL) + Kolla-ansible (KA) + Openstack (OS);
- ***ExpVMBM*** e ***ExpVMKA*** (Export VM): Tempo correspondente a exportação de máquina virtual;
- ***ImpVMBM*** e ***ImpVMKA*** (Import VM): Tempo correspondente a importação da máquina virtual;

Tabela 2. Configuração das Máquinas Virtuais

CPU/Mem	**4 GB**	**9 GB**	**14 GB**
2 Núcleos	2N4G	2N9G	2N14G
4 Núcleos	4N4G	4N9G	4N14G
6 Núcleos	6N4G	6N9G	6N14G

Uma execução completa pode ser considerada como a execução de todos os procedimentos de instalação e a configuração até o resultado final de criação da fatia com o serviço Openstack Tacker. Foram realizados três testes (execuções completas) para cada etapa da criação do *slice*. As configurações das VMs foram separadas por núcleos e memórias, sendo criadas conforme a Tabela 2.

4. Discussão dos Resultados

Após a captura do tempo de cada etapa durante o processamento, visando realizar cálculos de tomada de tempo para verificação de desempenhos, extraiu-se os menores valores de tempo compilados na Tabela 4. Observando os valores dos tempos para cada configuração de máquina virtual, conclui-se que o melhor computo de tempo foi 3,65 minutos, se refere ao teste ***OS*** seguido do teste ***KA*** que computou 9,52 minutos. A Tabela 3 apresenta a média das execuções de cada configuração de máquina, onde é possível observar novamente que as etapas ***OS*** e ***KA*** computaram os menores tempos médios, respectivamente 4,10 e 11,50 minutos. É possível observar também que as etapas ***KA*** e ***OS*** sucedem em ordem de tempo, tanto para a tomada de menor tempo quanto para a de tempo médio.

A etapa ***BM*** apresentou os maiores valores de tempos para todas configurações de máquina virtual quando comparado com as etapas ***KA*** e ***OS***, tendo como menor tempo o

Tabela 3. Média dos Tempos (Min)

VM	BM	KA	OS	TOTAL
2N 4G	67,84	20,10	5,19	94,27
2N 9G	59,31	14,85	4,33	78,95
2N 14G	47,83	10,35	3,99	62,51
4N 4G	50,93	18,29	4,77	73,80
4N 9G	42,82	11,50	4,10	58,51
4N 14G	40,97	10,35	3,74	55,46
6N 4G	46,07	14,62	4,46	64,84
6N 9G	40,84	10,87	3,99	55,38
6N 14G	39,27	9,52	3,65	52,62
Média	46,07	11,50	4,10	62,51

Tabela 4. Menores Tempos (Min)

VM	BM	KA	OS	TOTAL
2N 4G	58,89	20,10	5,19	94,27
2N 9G	56,61	14,85	4,33	78,95
2N 14G	42,68	10,35	3,99	62,51
4N 4G	48,28	18,29	4,77	73,80
4N 9G	41,83	11,50	4,10	58,51
4N 14G	39,34	10,35	3,74	55,46
6N 4G	43,66	14,62	4,46	64,84
6N 9G	39,80	10,87	3,99	55,38
6N 14G	38,44	9,52	3,65	52,62
Menor	38,44	9,52	3,65	52,62

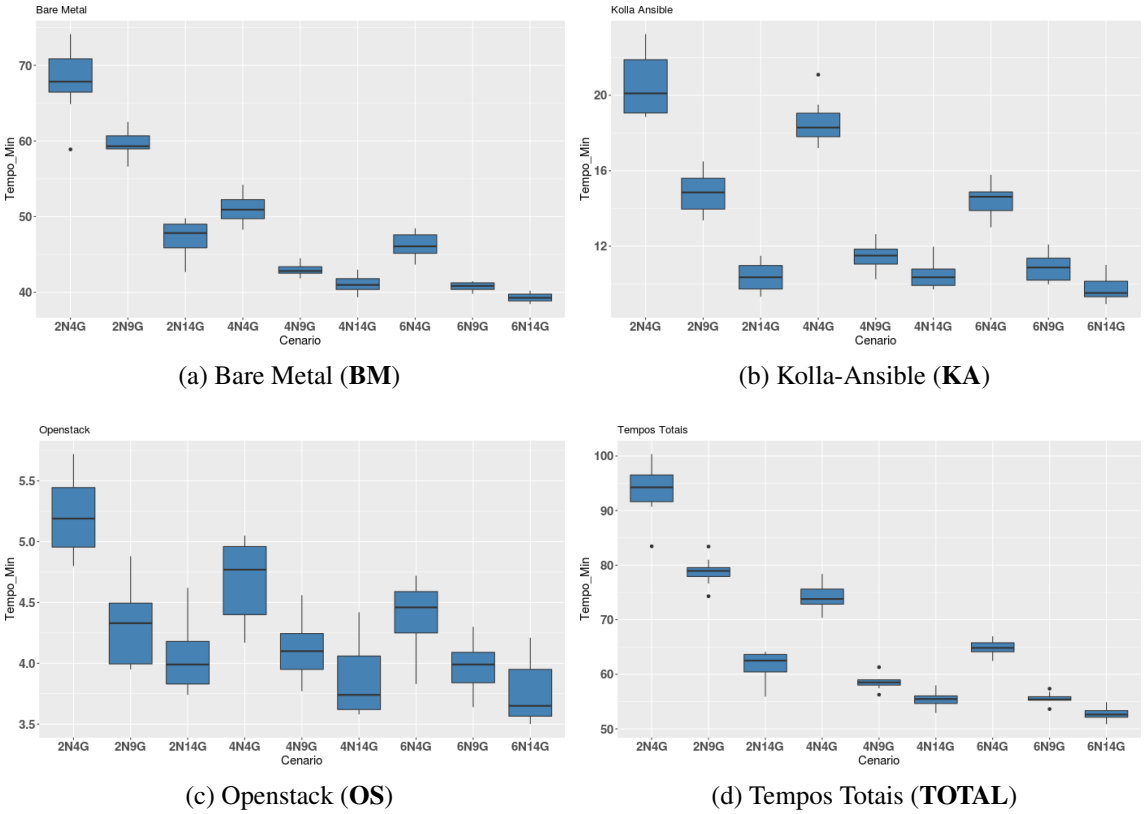
valor de 38,44 minutos. Ao observar mais atentamente, é possível notar que independentemente das configurações dos números de núcleos e memória, a ordem de relevância em função da tomada de tempo obedece sempre a mesma ordem, ou seja, na sequência ***BM***, ***KA*** e ***OS***. Para a representatividade de uma instalação completa, realiza-se a computação de todas as etapas para cada configuração de máquinas virtuais, tendo como o valor médio 62,51 minutos e como menor valor o tempo de 52,62 minutos.

De modo a observar o quão relevante se faz a capacidade computacional diante das tarefas propostas, observa-se que na Figura 6a, para o cenário do passo ***BM*** a quantidade de núcleos e memória interferem positivamente na tomada de tempo. A execução combinando 6 núcleos de processamento, com 14 e 9 gigabytes de memória RAM, aferiram desempenhos similares. Entretanto, com 14 gigabytes há uma distribuição de tempo mais estável, podendo ser observado pela menor distância entre o sétimo e o nono quartil. Desempenho semelhante ocorre para configuração com 4 núcleos e com 14 e 9 gigabytes de memória.

As tomadas de tempo para o passo ***KA***, observadas na Figura 6b, assemelham-se aos cômputos do passo ***OS***, exceto pela escala de tempo na qual a ordem de tempo do ***OS*** é menor. A similaridade do comportamento repete-se inclusive entre os *boxplots* das configurações de 6 núcleos combinados com 14 e 9 gigabytes, sinalizando homogeneidade das tomadas de tempo.

A Figura 6d apresenta a somatória dos resultados das etapas ***BM***, ***KA*** e ***OS***, onde demonstram a representatividade total da execução durante os testes, permitindo visualizar alguns pontos flutuantes.

Dos experimentos realizados com ***OS***, observa-se comportamento normal, o que pode ser visualizado na Figura 6c. Ao observar as tomadas por núcleos, analisamos que os grupos com 2, 4 e 6 núcleos, são decrescentes e para cada grupo há também uma variação decrescente conforme aumenta a quantidade de memória. Entretanto, ao se observar o comportamento entre os experimentos com 14 gigabytes de memória combinados com 6 e 4 núcleos, nota-se resultados similares em termos de estabilidade, ou seja, o conjunto de tomada de tempos possui igualdade na distribuição normal, mesmo advindo da configuração com menor número de núcleos, sugerindo que neste caso, a quantidade de memória é preponderante.



(a) Bare Metal (**BM**)

(b) Kolla-Ansible (**KA**)

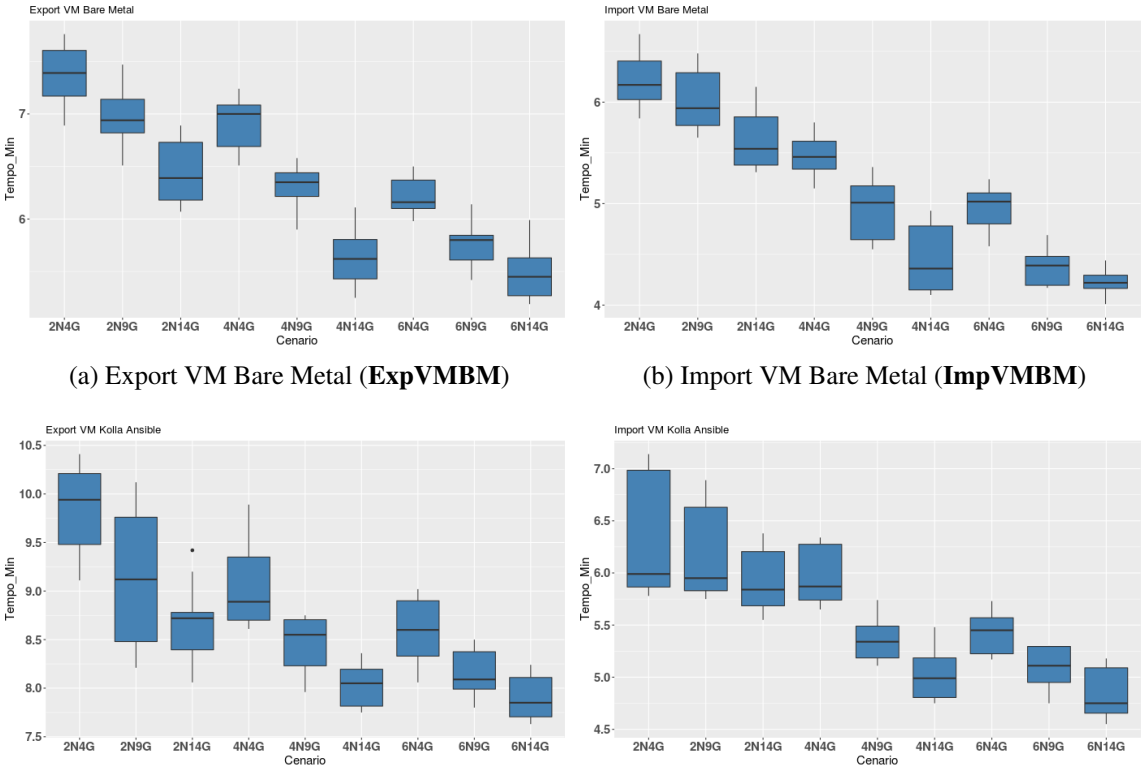
(c) Openstack (**OS**)

(d) Tempos Totais (**TOTAL**)

Figura 6. Execuções das etapas por configurações de máquinas virtuais.

Considerando as possíveis opções de instalações: Instanciação Completa com Imagem pré-carregada (Subseção 3.2.2) e Instanciação Completa com *containers* (Subseção 3.2.3), averiguou-se os tempos para exportação e importação de cada *snapshot*, conforme a Figura 7. As Figuras 7a e 7b representam, respectivamente a exportação e importação do *snapshot* de uma máquina virtual nas quais já foram realizadas as configurações das suas interfaces, instalação e configuração de serviços de provisionamento *bare metal*; enquanto as Figuras 7c e 7d reproduzem, respectivamente, a exportação e importação do *snapshot* de uma máquina virtual onde já foram realizadas a instalação e configuração do Openstack Ironic (*bare metal*) e do Openstack Kolla-ansible (Controlador Openstack).

Os tempos apresentados pelas Figuras 7c e 7d, foram maiores que os das Figuras 7a e 7b já que os *snapshots* utilizados possuem maior tamanho correspondente a quantidade de mais instalações realizadas no mesmo. Apesar deste fato, pode-se perceber que durante a realização das exportações para ambas situações (*bare metal* e kolla-ansible), suas realizações foram similares, assim como as execuções das importações. O principal fator de diferencial entre as exportações e importações foram os altos tempos de processamento apresentados durante as importações dos *snapshots* com o cenário de 2 núcleos. Quando comparado com todos os testes realizados em etapas anteriores, mesmo a situação com a menor quantidade de núcleos (2) e com a maior quantidade de memórias (14GB), acabou sendo tão eficientes quanto a uma situação de quantidade de 4 núcleos e com 4GB de memória.



(a) Export VM Bare Metal (**ExpVMBM**)

(b) Import VM Bare Metal (**ImpVMBM**)

(c) Export VM Kolla-Ansible (**ExpVMKA**)

(d) Import VM Kolla-Ansible (**ImpVMKA**)

Figura 7. Exportação e importação de máquinas virtuais.

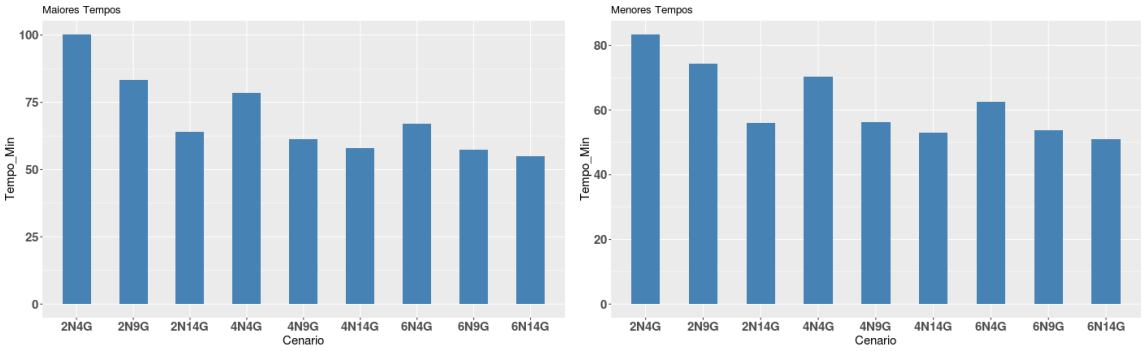
Corroborando para a constatação da prevalência da quantidade de memória em detrimento do número de núcleos, notamos nas Figuras 8a e 8b, uma ligeira superioridade quanto a estabilidade, ou seja, as barras relativas às configurações com 14 gigabytes de memória sobre as configurações com 6 núcleos.

A Figura 8a apresenta os maiores tempos para cada configuração de máquina virtual durante o processo de execução dos testes ***BM***, ***KA*** e ***OS***, enquanto a Figura 8b, ao contrário da anterior, apresenta os menores tempos durante o processo de execução dos testes. Ambas apresentam a variação de período durante a execução para cada cenário, permitindo visualizar, embora timidamente, uma relevante diferença entre cada cenário de configurações das máquinas virtuais.

Considerando o comportamento homogêneo para os cenários apresentados até o momento é possível afirmar à questão *(i)* através dos recursos computacionais que são preponderantes na eficiência do processo e dadas nossas observações a memória principal, prevalece sobre a quantidade de núcleos.

Com relação às médias das tomadas de tempo para os experimentos, a Figura 6d exibe as médias e respectivas marcações de desvio padrão. É possível observar que obstante a onerosidade do tempo de instalação do *bare metal* (Figura 6a), os tempos dos passos de Exportação e Importação de VMs (Figura 7) são extremamente vantajosos em relação aos tempos de instalação do *Kolla-Ansible* (Figura 6b) e *Openstack* (Figura 6c).

É notória a diferença estatística das atividades de não instalação, ou seja, importa-



(a) Maiores Tempos (b) Menores Tempos

Figura 8. Tempo MAX e MIN

ção de configurações de VMs sobre os tempos de instalação, pois os tempos das mesmas são inferiores ao tempo médio de ambas e subtraídas do desvio padrão. Deste modo, pode-se afirmar que a resposta para a questão *(ii)* aponta no sentido de que a abordagem VIM *on-demad* é fortemente vantajosa, implicando na eficiência de utilização de recursos computacionais sob esta abordagem.

5. Considerações finais

Este trabalho apresentou um modelo de VIM *on-demand*, a qual possibilita instanciar máquinas a nível *bare metal* e serviços de computação, proporcionando, ao final, o seu controle e gerenciamento. Além disso, também foi possivel dividir o processo de instalação, configuração e instanciação em 3 etapas e exportar um *snapshot* para cada ambiente: (*I*) Instanciação Completa *Bare Metal*, (*II*) Instanciação Completa com *Containers* e (*III*) Instanciação Completa com Imagem Pré-Carregada.

Ao concluir os experimentos, observa-se que os melhores resultados durante os experimentos foram os que usaram memória RAM com 14 GB (G14), a maior quantidade de memória utilizada durante os testes. Diante disso, conclui-se que a quantidade de memória é aspecto preponderante no VIM *on-Demand*, pois os valores que apresentaram melhor desempenho foram em situações de maior quantidade de memória, e não alteração de outros fatores como quantidade de núcleos. Adicionalmente à preponderância da quantidade de memória, foi possível constatar que a abordagem VIM *on-demand* confere ganho computacional no tocante a eficiência de uso dos recursos.

Como trabalhos futuros, pretende-se incluir outros controladores, como o VLSP e Kubernets, no VIM *on-Demand*, e realizar novos experimentos analisando consumo de energia, desempenho de memória e processamento com casos de usos que necessitem de uma sobrecarga.

Agradecimentos

Esta pesquisa é financiada pela Comissão Europeia e Ministério da Ciência, Tecnologia, Inovação e Comunicação (MCTIC) através da RNP e CTIC no contexto da 4º chamada conjunta EU-BR, acordo #777067 (*NECOS - Novel Enablers for Cloud Slicing*). Este estudo foi também parcialmente financiado pela CAPES e CNPq.

Referências

Clayman, S. (2017). Network slicing supported by dynamic vim instantatiation.

Clayman, S., Tusa, F., and Galis, A. (2018). Extending slices into data centers: the vim on-demand model. In *2018 9th International Conference on the Network of the Future (NOF)*, pages 31–38. IEEE.

Dräxler, S., Karl, H., Peuster, M., Kouchaksaraei, H. R., Bredel, M., Lessmann, J., Soenen, T., Tavernier, W., Mendel-Brin, S., and Xilouris, G. (2017). Sonata: Service programming and orchestration for virtualized software networks. In *2017 IEEE International Conference on Communications Workshops (ICC Workshops)*, pages 973–978. IEEE.

Freitas, L. A., Braga, V. G., Corrêa, S. L., Mamatas, L., Rothenberg, C. E., Clayman, S., and Cardoso, K. V. (2018). Slicing and allocation of transformable resources for the deployment of multiple virtualized infrastructure managers (vims). In *2018 4th IEEE Conference on Network Softwarization and Workshops (NetSoft)*, pages 424–432. IEEE.

Han, B., Gopalakrishnan, V., Ji, L., and Lee, S. (2015). Network function virtualization: Challenges and opportunities for innovations. *IEEE Communications Magazine*, 53(2):90–97.

Kim, J., Kim, D., and Choi, S. (2017). 3gpp sa2 architecture and functions for 5g mobile communication system. *ICT Express*, 3(1):1–8.

Mademann, F. (2017). System architecture milestone of 5g phase 1 is achieved. *Online verfügbar unter: http://www.3gpp.org/NEWS-EVENTS/3GPP-NEWS/1930-SYS_ARCHITECTURE, letzter Zugriffam*, 1:2018.

Ordonez-Lucena, J., Ameigeiras, P., Lopez, D., Ramos-Munoz, J. J., Lorca, J., and Folgueira, J. (2017). Network slicing for 5g with sdn/nfv: concepts, architectures and challenges. *arXiv preprint arXiv:1703.04676*.

Rost, P., Mannweiler, C., Michalopoulos, D. S., Sartori, C., Sciancalepore, V., Sastry, N., Holland, O., Tayade, S., Han, B., Bega, D., et al. (2017). Network slicing to enable scalability and flexibility in 5g mobile networks. *IEEE Communications magazine*, 55(5):72–79.

Silva, F. S. D., Lemos, M. O., Medeiros, A., Neto, A. V., Pasquini, R., Moura, D., Rothenberg, C., Mamatas, L., Correa, S. L., Cardoso, K. V., et al. (2018). Necos project: Towards lightweight slicing of cloud federated infrastructures. In *2018 4th IEEE Conference on Network Softwarization and Workshops (NetSoft)*, pages 406–414. IEEE.