UNIVERSIDADE DO ESTADO DE SANTA CATARINA - UDESC
CENTRO DE CIÊNCIAS TECNOLÓGICAS - CCT
MESTRADO EM COMPUTAÇÃO APLICADA

ADNEI WILLIAN DONATTI

CARACTERIZAÇÃO DO TRÁFEGO DE REDE NO DOMÍNIO DE CONTROLE DO OPENSTACK PRODUZIDO PELA MUDANÇA DE ESTADO DE MÁQUINAS VIRTUAIS

JOINVILLE

2020

ADNEI WILLIAN DONATTI

CARACTERIZAÇÃO DO TRÁFEGO DE REDE NO DOMÍNIO DE CONTROLE DO OPENSTACK PRODUZIDO PELA MUDANÇA DE ESTADO DE MÁQUINAS VIRTUAIS

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

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

JOINVILLE

2020

Ficha catalográfica elaborada pelo programa de geração automática da

Biblioteca Setorial do CCT/UDESC,

com os dados fornecidos pelo(a) autor(a)

Donatti, Adnei Willian
Caracterização do tráfego de rede no domínio de controle do OpenStack produzido pela mudança de estado de máquinas virtuais / Adnei Willian Donatti. -- 2020.
90 p.

Orientador: Charles Christian Miers
Coorientador: Guilherme Piêgas Koslovski
Dissertação (mestrado) -- Universidade do Estado de Santa Catarina, Centro de Ciências Tecnológicas, Programa de Pós-Graduação em Computação Aplicada, Joinville, 2020.

1. Computação em nuvem. 2. OpenStack. 3. Tráfego de rede. 4. Caracterização. 5. Máquina Virtual. I. Miers, Charles Christian. II. Koslovski, Guilherme Piêgas. III. Universidade do Estado de Santa Catarina, Centro de Ciências Tecnológicas, Programa de Pós-Graduação em Computação Aplicada. IV. Titulo.

Adnei Willian Donatti

Caracterização do tráfego de rede no domínio de controle do OpenStack produzido pela mudança de estado de máquinas virtuais

Esta dissertação foi julgada adequada para a obtenção do título de **Mestre em Computação Aplicada** área de de concentração em "Sistemas de Computação", e aprovada em sua forma final pelo Curso de Mestrado em Computação Aplicada do Centro de Ciências Tecnológicas da Universidade do Estado de Santa Catarina.

Banca Examinadora:

Dr. Charles Christian Miers
Orientador

Dr. Guilherme Piêgas Koslovski
Coorientador

Dr. Maurício Aronne Pillon
Convidado 1

LD. Tereza Cristina Melo de Brito Carvalho
Convidado 2

Dr. Fernando Frota Redígolo
Convidado 3

Joinville, 30 de Novembro de 2020

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

AGRADECIMENTOS

Em primeiro lugar, agradeço à minha mãe e ao meu pai (dona Carla e seu Valdecir), por terem me criado com muita simplicidade, honestidade e humildade. Certamente, sem o apoio (tanto emocional quanto financeiro) de meus pais, jamais teria chegado até aqui, então, gostaria de deixar clara minha eterna gratidão. À minha família, saibam que minha ausência não foi em vão, e que dei meu máximo para valorizar o que fizeram por mim.

Também quero agradecer imensamente aos meus professores orientadores, Prof. Dr. Charles Christian Miers e Prof. Dr. Guilherme Piêgas Koslovski. Principalmente ao professor Charles, que me apoiou e acreditou em mim quando nem mesmo eu acreditava. Admiro muito o trabalho dos meus "chefes", e espero um dia conseguir passar adiante todos os ensinamentos que me foram passados, e que ainda estão sendo passados. Creio que essa seja a melhor forma de retribuir e mostrar o quanto toda essa jornada significa pra mim. Além disso, agradeço aos demais professores do PPGCA e colaboradores do LabP2D, que, diretamente, participaram (e ainda participam) do meu processo de evolução acadêmico. Agradeço também aos membros da banca avaliadora e aos demais leitores deste documento, que reservam um pouco do seu tempo para dar atenção às ideias aqui descritas.

Não posso deixar de mencionar a minha gratidão a todos os meus amigos, mesmo os mais distantes, que me motivaram, direta ou indiretamente, a não desistir. Principalmente os colegas do LabP2D, que durante essa jornada tornaram-se mais que meus colegas. Alguns tornaram-se amigos tão próximos a ponto de estarem presentes até mesmo nos momentos mais complicados, estendendo a mão e oferecendo apoio. Espero tê-los sempre em minha vida. Além disso, ficará na memória os momentos que passei com amigos e familiares no Opa Bier, River Falls e no "Bar Secreto", que mesmo em momentos difíceis, me proporcionaram alegria e distração.

Por fim, mas não menos importante (não mesmo!), gostaria de dedicar este trecho final ao meu "irmãozinho". Meu fiel companheiro e cão de guarda, Zombie. Zombie sempre esteve disposto a me acompanhar aonde quer que eu fosse. Eu estando feliz ou não. Eu o dando a devida atenção e carinho ou não.

“Educação é uma descoberta progressiva de nossa própria ignorância.”

Voltaire

RESUMO

A adoção de nuvens computacionais privadas, de pequeno a grande porte, é uma opção para otimização no uso dos recursos computacionais de uma organização. Embora os benefícios do uso já sejam conhecidos faz algum tempo, ainda há questões de como planejar nuvens para evitar problemas básicos de desempenho. Neste sentido, nuvens privadas usando a solução de código aberto OpenStack tem sido preferidas em detrimento a demais soluções nesta categoria. O OpenStack distribui o tráfego de rede entre diversas interfaces e redes virtualizadas, que conectam *hosts* com serviços de nuvem, divididos em domínio de: controle, público ou convidado. O presente trabalho tem como objetivo caracterizar o tráfego de rede no domínio de controle do OpenStack produzido pela mudança de estado de máquinas virtuais (MVs). Pouco se trata sobre o cenário de infraestrutura de redes de nuvens computacionais, as pesquisas nesta área voltam-se, geralmente, a operações no domínio público ou de convidados. Neste sentido, é realizada uma caracterização na rede administrativa do OpenStack, no sentido de planejar e conduzir uma caracterização/classificação do tráfego nesta rede. A fim de realizar a caracterização sobre a rede do OpenStack, são utilizados métodos de experimentação que buscam identificar os serviços operantes, bem como aferir o tráfego de rede no domínio de controle do OpenStack durante um conjunto de operações comuns em MVs. A caracterização realizada permitiu definir o comportamento, tanto de tráfego de rede como de chamadas de serviço, no domínio de gerenciamento do OpenStack. Além disso, foram identificadas que as operações envolvidas na criação e engavetamento de instâncias são as principias consumidoras de recursos.

Palavras-chaves: Computação em nuvem, OpenStack, tráfego de rede, caracterização, máquina virtual.

ABSTRACT

The adoption of private computing clouds, from small to large, is an option for optimizing the use of an organization's computing resources. Although the benefits of using clouds have been known for some time, there are still questions about of how to plan for clouds in order to avoid basic performance issues. In this context, private clouds using the open source OpenStack solution have been preferred over other solutions in this category. OpenStack distributes network traffic among several interfaces and virtualized networks, which connect hosts the cloud services, divided into the domain of: control, public, and guest. This work characterize the network traffic from the OpenStack management domain produced by *Virtual Machine* (VM) state changes. The cloud provider network architecture is not a topic frequently discussed. Researches in this area usually focus on the guest or public domains. Thus, a characterization is performed in the OpenStack administrative network, in order to classify and better understand the behavior of network traffic in the management domain. In order to characterize the OpenStack network, experimentation methods are used to identify the operating services, as well as to measure the network traffic in the OpenStack control domain during a set of common operations in VMs. The characterization made it possible to define the behavior of both network traffic and service calls in the OpenStack management domain. Moreover, we identified the operations involved in the creation and shelving of VM instances are the main resource consumers.

Keywords: Cloud computing, OpenStack, network traffic, characterization, virtual machine

LISTA DE ILUSTRAÇÕES

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

ACL *Access Control List*
AMQP *Advanced Message Queuing Protocol*
API *Application Programming Interface*
AWS *Amazon Web Services*
CaaS *Container as a Service*
CLI *Command Line Interface*
CSA *Cloud Security Alliance*
DoS *Denial of Service*
DPI *Deep Packet Inspection*
EC2 *Amazon Elastic Cloud Computing*
ENISA *European Network and Information Security Agency*
FaaS *Function as a Service*
FTP *File Transfer Protocol*
HTTP *Hypertext Transfer Protocol*
IaaS *Infrastructure as a Service*
IP *Internet Protocol*
LabP2D Laboratório de Processamento Paralelo Distribuído
LBaaS *Load Balance as a Service*
LVM *Logical Volume Manager*
LXC *LinuX Containers*
LXD *Linux container hypervisor*
MAC *Media Access Control*
MAPE *Mean Absolute Percentage Error*
MMX *Min Max Accuracy*
MV máquina virtual
NaaS *Networking as a Service*
NIC *Network Interface*
NIDS *Network Intrusion Detection Systems*
NIST *National Institute of Standards and Technology*
ONM OpenStack Network Monitor
OSMeF OpenStack *Measurement Framework*
P2P *Peer-to-Peer*
PaaS *Platform as a Service*

PBS Pesquisa Bibliográfica Sistemática

PPGCA Programa de Pós-Graduação em Computação Aplicada

QoS *Quality of Service*

REST *Representational State Transfer*

RPC *Remote Procedure Call*

RSA *Rivest-Shamir-Adleman*

SaaS *Software as a Service*

SDK *Software Development Kit*

SDN *Software Defined Networking*

SLA *Service Level Agreement*

SO sistema operacional

SPF *Single Point of Failure*

SSH *Secure Shell*

STP *Spanning Tree Protocol*

TCP Transmission Control Protocol

TI Tecnologia e Informação

UDESC Universidade do Estado de Santa Catarina

VLAN *Virtual Local Area Network*

VM *Virtual Machine*

VMM *Virtual Machine Manager*

WAN *Wide Area Network*

SUMÁRIO

1 INTRODUÇÃO

A computação em nuvem possibilita uma forma otimizada e sob demanda de fornecer e consumir recursos computacionais como processamento, armazenamento e rede. Neste contexto, as soluções de nuvem empregam sistemas operacionais, técnicas de virtualização e sistemas de arquivos que já existem há décadas, mas organizados e gerenciados de uma forma diferente. A virtualização, tanto de recursos computacionais como de redes, é um dos principais elementos que auxiliam no modelo de computação em nuvem. É por meio da virtualização que existem tecnologias como MVs e contêineres, que trazem benefícios principalmente à eficiência operacional, no sentido de aumentar a carga de trabalho que é possível ser realizada por *hardware* (SCARFONE; SOUPPAYA; HOFFMAN, 2011).

A criação e gerenciamento de nuvens computacionais *Infrastructure as a Service* (IaaS), de um modo geral, é realizada com um conjunto de softwares e ferramentas que aliam técnicas de sistemas distribuídos à virtualização, de modo a criar um ambiente no qual é possível fornecer recursos computacionais (OPENSTACK, 2019e). Neste contexto, existem soluções de código aberto voltadas à criação de nuvens computacionais públicas e privadas. Dentre estas soluções estão OpenStack (OPENSTACK, 2019f) e o CloudStack (PROJECT, 2019). O OpenStack, foco deste trabalho, é considerado um sistema operacional para nuvens, que controla um amplo grupo de recursos de computação, armazenamento e rede por todo o *data center*.

No geral, o OpenStack fornece métodos para que os administradores da nuvem possam gerenciar questões de segurança e desempenho da nuvem. Neste sentido, as implantações do OpenStack devem ser pensadas de modo que o isolamento entre os usuários da nuvem e seus projetos seja garantido. Existem diversos estudos feitos a partir do ponto de vista do usuário de nuvens OpenStack, que envolvem tanto a qualidade de serviço quanto a privacidade na nuvem (Chen; Zhao, 2012; Chaudhary et al., 2018; ALENEZI; ALMUSTAFA; MEERJA, 2019). Contudo, são escassos os estudos sobre componentes e funcionamento interno da nuvem, principalmente em relação à rede administrativa da nuvem, que é vital ao seu funcionamento. Neste sentido, propõe-se uma análise e caracterização do tráfego da rede administrativa do OpenStack, a fim de identificar serviços e operações em nível de camada de rede que estão sendo executadas durante o uso da nuvem. Além disso, a caracterização/classificação do tráfego interno do OpenStack ajuda nas configurações de redes, no sentido de melhor distribuir/configurar os recursos disponíveis (*e.g.*, separar as redes de acordo com políticas de segurança, uso de rede, alocar/desalocar recursos de rede) a fim de ter ganhos em desempenho e segurança da nuvem. Portanto, define-se um método de

pesquisa baseada em pesquisa aplicada. São propostos experimentos e testes, dentro de um cenário pré-estabelecido, que visam caracterizar como ocorrem as variações no tráfego administrativo do OpenStack de acordo com a instância de MV utilizada pelo usuário.

O domínio administrativo do OpenStack, por vezes visto como domínio de controle, alvo deste estudo, pode ser considerado o domínio mais operacional da nuvem. Todas as questões relacionadas à operação da nuvem estão, de certa forma, contidas no domínio administrativo da nuvem. Portanto, as redes contidas no domínio administrativo agregam a execução de serviços essenciais ao funcionamento da nuvem. A comunicação que ocorre na rede administrativa é interna, no sentido de que envolve elementos internos da nuvem. O mal funcionamento da rede administrativa (*e.g.*, congestionamento da rede) afeta diretamente o serviço de computação em nuvem. As funcionalidades (*e.g.*, uso de MVs, contêineres, gerenciamento de usuários da nuvem) podem ficar lentas ou até mesmo indisponíveis.

Dada a importância do domínio administrativo à nuvem, são identificados alguns aspectos que podem ser explorados:

- quais os elementos internos à nuvem (serviços) que se comunicam pela rede administrativa;
- como ocorre a comunicação entre os serviços;
- qual a intensidade da comunicação entre serviços na rede (volume de tráfego e requisições); e
- como planejar o domínio administrativo, em termos de configuração das redes e largura de banda.

Neste sentido, com a análise e caracterização do tráfego administrativo do OpenStack há uma grande contribuição ao entendimento dos pontos identificados.

O presente trabalho busca abordar assuntos relativos ao funcionamento interno de nuvens OpenStack, mais especificamente, tratando da infraestrutura da nuvem associada ao uso de instâncias de MVs (*e.g.*, configuração de redes nos *data centers* de implantações OpenStack e gerenciamento de MVs no OpenStack). O objetivo geral aqui identificado é a classificação/caracterização do tráfego da rede administrativa do OpenStack em operações de mudança de estados de MVs em um ciclo de vida induzido. O ciclo de vida induzido reflete as operacões sobre MV mais comuns ao usuários, *e.g.*, criar, suspender, resumir e parar. Dessa forma, também são abordados objetivos específicos, para que o objetivo geral seja atendido:

1. Identificar o funcionamento do OpenStack e como suas redes e serviços estão organizados;
2. Identificar e usar uma arquitetura base de implantação do OpenStack;
3. Coletar dados do tráfego da rede administrativa;
4. Analisar os dados coletados; e
5. Caracterizar/classificar o tráfego coletado por MV e operação de mudança de estado.

A organização do presente documento busca, de início, abordar conceitos fundamentais no contexto de nuvens OpenStack (Capítulo 2). Desta forma, é possível evoluir em conceitos específicos do OpenStack e, então, prover a base ao entendimento do problema e proposta de solução do trabalho. Principalmente durante a definição do problema, são retomados conceitos-chave, previamente discutidos, para entendimento e contextualização. Além disso, os resultados são apresentados em formatos de tabelas e gráficos, que mostram os dados coletados durante os experimentos. Neste sentido, o documento é organizado de modo que o Capítulo 2 introduz a fundamentação. O Capítulo 3 explica o funcionamento geral de instâncias de MVs no OpenStack, são discutidos a definição do problema e os trabalhos relacionados. O Capítulo 4 traz a proposta bem como sua implementação. O Capítulo 5 mostra os resultados obtidos. Por fim, o Capítulo 6 traz as considerações, possíveis direções futuras, contribuições do trabalho e publicações feitas até o presente momento.

2 FUNDAMENTAÇÃO

A computação em nuvem (apresentanda em detalhes na Seção 2.1) possibilita uma forma otimizada e sob demanda de fornecer e consumir recursos computacionais como processamento, armazenamento e rede. Neste contexto, as soluções de nuvem fazem uso de mecanismos de orquestração e gerenciamento para que diversas tecnologias já existentes possam ser utilizadas harmoniosamente. Assim, soluções de nuvem combinam sistemas operacionais, técnicas de virtualização e sistemas de arquivos, que já existem há décadas, mas organizados e gerenciados de uma forma diferente. A virtualização é um dos principais elementos envolvidos no conceito de nuvens IaaS, não apenas com as MVs (Seção 2.2), mas também com a virtualização de recursos de rede (Seção 2.3).

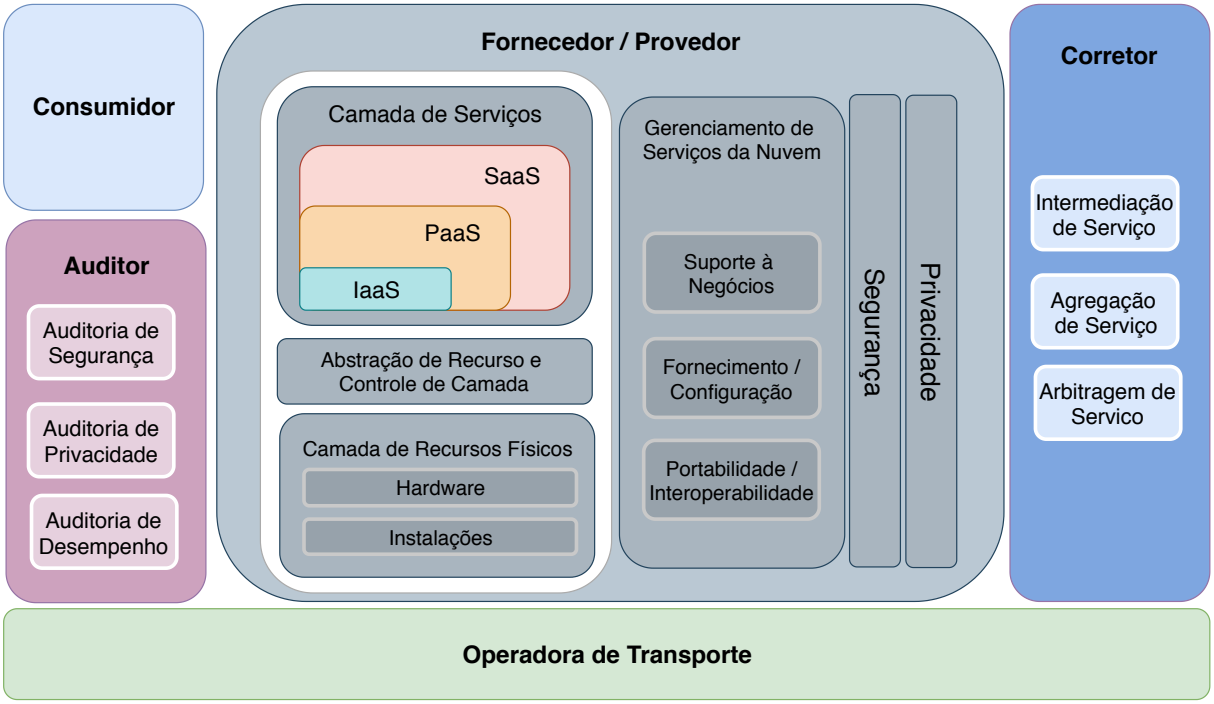
2.1 COMPUTAÇÃO EM NUVEM

O modelo de computação em nuvem permite o acesso a um conjunto de recursos computacionais (*e.g.*, armazenamento, rede e serviços) de maneira conveniente, ubíqua e escalável (MELL; GRANCE, 2011). Neste sentido, o serviço de computação em nuvem pode ser distribuído de diversas formas. De acordo com o NIST, a oferta deste serviço pode ser dividida em três categorias: IaaS, *Platform as a Service* (PaaS) e *Software as a Service* (SaaS) (Figura 1 - Camada de Serviços). O IaaS permite que o consumidor tenha acesso a recursos de mais baixo nível, como rede, armazenamento e processamento. Neste sentido, o consumidor pode até mesmo controlar o software na MV. Já no PaaS, o consumidor tem acesso a uma plataforma pré-configurada com um conjunto de linguagens e bibliotecas. Neste caso, o consumidor pode controlar as aplicações que serão instaladas por ele, bem como algumas configurações sobre o ambiente. Em relação ao modelo SaaS, as permissões concedidas ao consumidor são de mais alto nível. Então, há acesso às aplicações que executam sobre a infraestrutura da nuvem, bem como a algumas configurações específicas para aquela aplicação. É importante ressaltar que diversas variações e especificações das categorias foram propostas por diferentes provedores de nuvem, *e.g.*, *Function as a Service* (FaaS), *Container as a Service* (CaaS), *Load Balance as a Service* (LBaaS), entre outros.

O NIST também classifica as nuvens computacionais de acordo com seu modelo de implantação, sendo os modelos público e privado os mais comuns. Na nuvem pública, a infraestrutura é de uso aberto, e é gerenciada por alguma entidade (*e.g.*, empresa, organização governamental, organização acadêmica). Já as nuvens computacionais privadas operam sobre uma infraestrutura própria, que é mantida pela

organização que a possui, *i.e.*, toda a manutenção da nuvem, bem como aspectos de segurança e desempenho são de responsabilidade desta organização. Além disso, as nuvens privadas buscam atender aos propósitos da organização, e são acessíveis somente aos indivíduos com permissão, o que garante à organização total controle sobre os dados ali contidos (JADEJA; MODI, 2012). Dentre os softwares de código aberto que permitem a criação de nuvens públicas e privadas, destacam-se o OpenStack (OPENSTACK, 2019f) e o CloudStack (PROJECT, 2019). Sendo o OpenStack uma solução de código aberto com crescente uso mundial, principalmente por aumentar a eficiência operacional, acelerar a inovação e evitar *vendor lock-in* (OPENSTACK, 2018a).

Figura 1 – Modelo de referência para computação em nuvem de acordo com o NIST.



Fonte (BOHN et al., 2011)

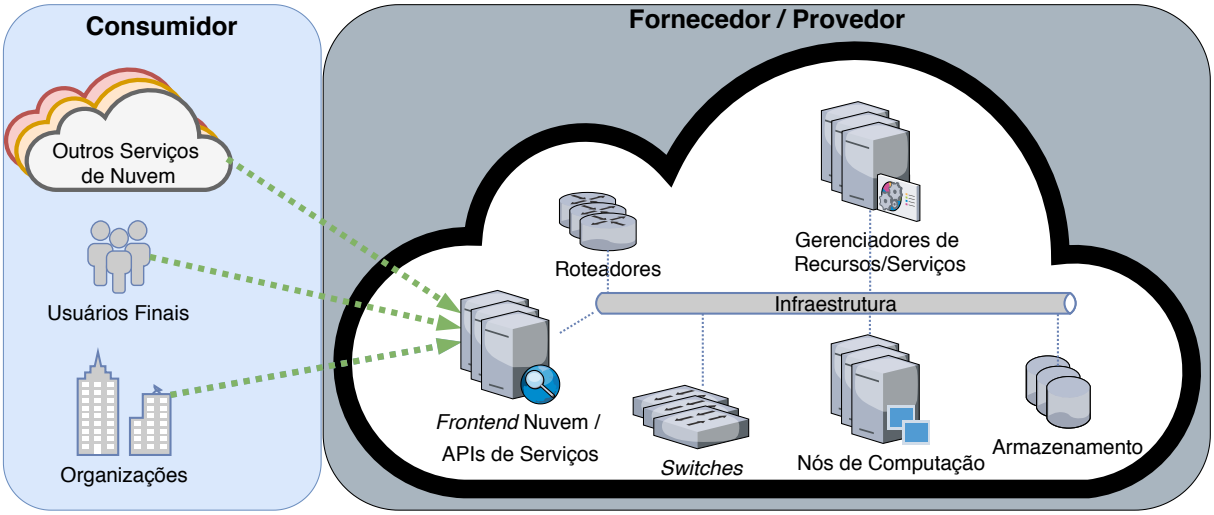
Os principais atores envolvidos no modelo de computação em nuvem (Figura 1) são:

- **Consumidor**: usam dos serviços da nuvem. Pode ser tanto uma pessoa quanto uma organização que faz uso dos serviços ofertados pelo provedor da nuvem;
- **Fornecedor/Provedor**: parte responsável por disponibilizar o serviço de computação em nuvem às partes interessadas;
- **Auditor**: uma pessoa ou entidade responsável por avaliar a nuvem de modo geral (*e.g.,* avaliações de segurança e desempenho);

- **Corretor**: Entidade que atua no controle de uso, desempenho e distribuição de serviços em nuvem; e
- **Operadora de Transporte**: atua como um intermediário entre o provedor e consumidor, de modo a possibilitar a conectividade e o transporte de serviços entre eles (BOHN et al., 2011).

A Figura 1 ilustra o modelo de referência para computação em nuvem de acordo com o NIST. Dessa forma, em análise a Figura 1, o presente trabalho situa-se na caracterização do tráfego gerado na rede da nuvem a partir da interação do Consumidor (em azul claro) com as operações em MVs. Assim, o foco está no tráfego do Provedor, oriundo de ações do Consumidor - Provedor para utilizar MVs, mais especificamente no que diz respeito à Infraestrutura do provedor para o Fornecimento/Configuração da nuvem (Figura 2, na área do Provedor).

Figura 2 – Principais atores do modelo de computação em nuvem.



Fonte: O próprio autor.

A Figura 2 apresenta o relacionamento entre alguns desses principais atores. Na Figura 2 é possível identificar diversos elementos da infraestrutura do Provedor envolvidos no processo de uso/disponibilização da computação em nuvem. Os usuários dos serviços solicitam os recursos para um provedor de nuvem. Cada serviço pode ser oferecido e executado sobre um conjunto de servidores virtualizados. Neste contexto, este trabalho leva em consideração a infraestrutura da nuvem, que é o meio pelo qual toda interação do usuário final com a nuvem é feita.

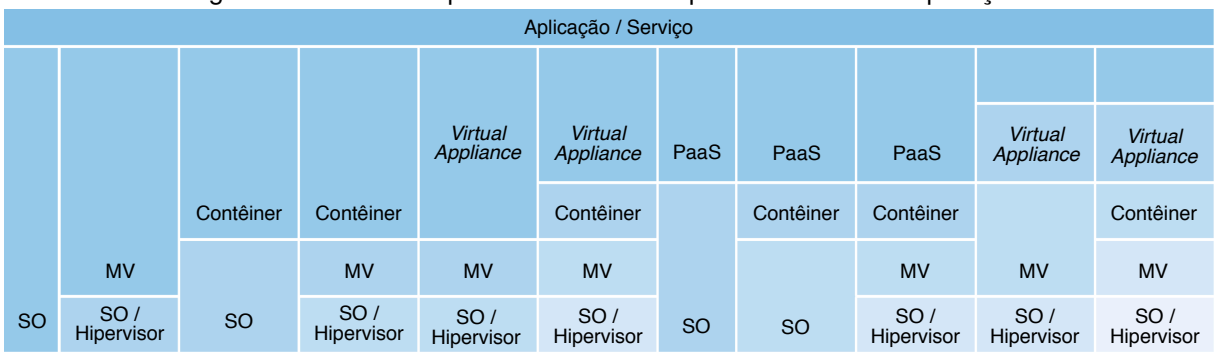
Uma das premissas deste modelo de computação é que vários consumidores possam solicitar os seus serviços simultaneamente. Dessa forma, todos os consumidores fazem uso da mesma infraestrutura da nuvem (Figura 2), que por sua vez, é composta por elementos como servidores, roteadores, *switches* e infraestrutura de

rede, que podem causar gargalos que afetam negativamente o fornecimento do serviço. Portanto, independente do modelo, cenário e solução de gerenciamento utilizado, o monitoramento e o conhecimento sobre as particularidades da carga computacional e de comunicação são essenciais para garantir qualidade na oferta de serviço e no gerenciamento da nuvem (*e.g.,* configurações de rede e alocação de recursos).

2.2 VIRTUALIZAÇÃO

Existem diversas formas de virtualização, inclusive, não apenas MVs podem ser criadas, como contêineres, *virtual appliances* e também, na parte de infraestrutura, redes virtualizadas (*e.g.*, roteadores, *switches* e VLANs). Neste sentido, um dos principais benefícios para a crescente adoção da virtualização é a eficiência operacional: as organizações conseguem aumentar a carga de trabalho por hardware, uma vez que um único computador pode hospedar diversas máquinas virtuais simultaneamente (SCARFONE; SOUPPAYA; HOFFMAN, 2011). Porém, quase toda a virtualização tem por objetivo propiciar alguma abstração para um serviço ou uma aplicação. Existem diferentes formas de fornecer recursos virtualizados (Figura 3).

Figura 3 – Formas de prover recursos computacionais a uma aplicação.



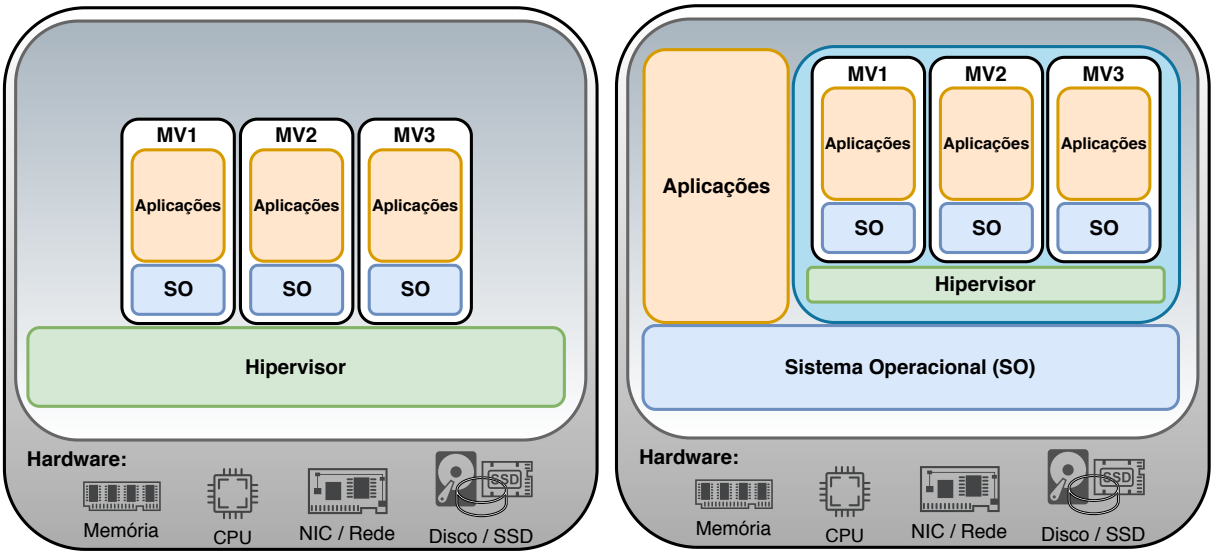
Fonte: (PANIZZON et al., 2019)

Observando a Figura 3, constata-se que é possível usar desde uma abordagem mais antiga com apenas o SO – Aplicação/Serviço até abordagens com SO/Hipervisor – MV – Contêiner – *Virtual Appliance* – Aplicação/Serviço. Apesar de existirem estas diversas combinações, e a adoção de contêineres estar crescendo consideravelmente, o uso de MVs continua muito popular quer na configuração mais tradicional SO/Hipervisor – MV – Aplicação/Serviço ou como base para Contêineres, PaaS ou *Virtual Appliance* (DAWSON, 2018). Neste sentido, embora a crescente adoção e evolução dos contêineres nas organizações de Tecnologia e Informação (TI), as MVs mostram-se bem consolidadas e estarão presentes por um bom tempo (DIAMANTI, 2018; DIAMANTI, 2019). Além disso, no cenário atual, a maioria das organizações de TI não dispõe das ferramentas e processos para gerenciar contêineres executando em um servidor *bare-metal*, as MVs reafirmam sua permanência (DIAMANTI, 2018).

A este ponto, a importância da virtualização já é justificada com a criação de MVs, contudo, sua aplicabilidade na computação em nuvem vai muito além. A virtualização permite criar infraestruturas lógicas, incluindo topologias de redes e computadores, que funcionam sobre os recursos físicos existentes. Dessa forma, além de criar as MVs, também é possível gerenciar o isolamento e a comunicação entre cada uma das máquinas. Geralmente, esta função é feita por algum software, como hipervisores, por exemplo. Além disso, um único servidor pode hospedar diversas instâncias (máquinas virtuais), que são gerenciadas por um hipervisor. No caso do OpenStack, que é o foco deste trabalho, os hipervisores comumente utilizados são o QEMU[1] e o KVM[2]. Por fim, embora a virtualização também possa ser realizada com contêineres, o escopo deste documento limita-se à utilização de máquinas virtuais.

Quando o hipervisor é instalado diretamente sobre o *hardware* (sem a existência de um sistema operacional no computador), tem-se a chamada virtualização Tipo 1, ou *bare metal*. Já na virtualização Tipo 2, o hipervisor executa em um sistema operacional. A Figura 4 ilustra a diferença entre os tipos de virtualização. O hipervisor é o principal responsável por todas as operações com as instâncias, como criação, que permite escolher um sistema operacional e criar a instância; edição, que possibilita a alteração de algumas configurações de hardware da instância; salvar estado da máquina (*snapshots*), que armazena informações da instância no momento em que foi solicitado, incluindo uso de memória e armazenamento/disco virtual; e exclusão, que permite a remoção total ou parcial da instância (CHANDRAMOULI, 2014).

Figura 4 – Exemplos de cenários com os dois principais tipos de virtualização.



Tipo 1 Tipo 2

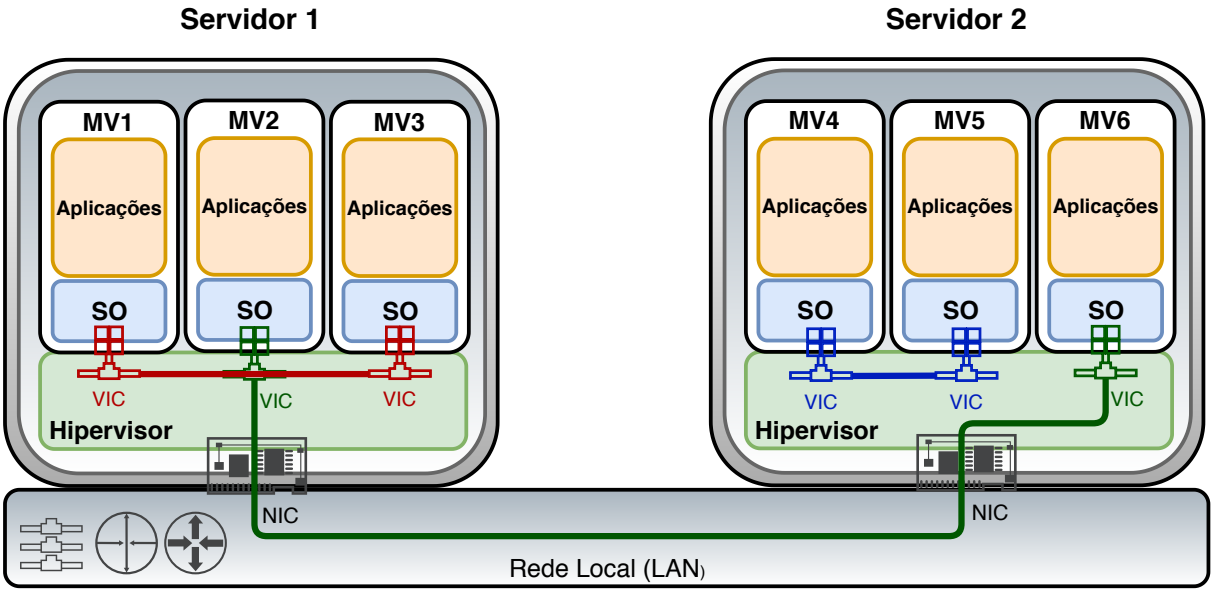
Fonte: O próprio autor.

[1] <www.qemu.org>
[2] <www.linux-kvm.org>

Além dos métodos de virtualização mostrados na Figura 4, também tem-se a virtualização baseada em contêineres. De fato, em *data centers*, por questões principalmente de desempenho, costuma-se aderir ao *bare metal* ou aos contêineres. Diferente da virtualização baseada em hipervisores, a virtualização baseada em contêineres não emula o hardware subjacente à aplicação, o que reduz consideravelmente as sobrecargas da virtualização (ROOZBEH, 2019). Ainda assim, já que o SO é responsável pelo isolamento entre os contêineres (feito com o *namespaces*, por exemplo, no caso de *Linux Containers*), os contêineres são altamente dependentes do SO da máquina servidor, o que não elimina totalmente as sobrecargas no processo de virtualização (ROOZBEH, 2019).

A virtualização possibilita que diversas MVs sejam hospedadas em um mesmo servidor. Além disso, atualmente, o hardware para servidores de virtualização torna possível hospedar dezenas e até mesmo centenas de MVs em um mesmo servidor (IBM GLOBAL SERVICES, 2016). Adicionando o fato que as aplicações cada vez tendem a ser mais distribuídas, tem-se a situação na qual a comunicação entre as aplicações/serviços podem ocorrer em MVs que estejam hospedadas em um mesmo servidor e/ou servidores distintos. A Figura 5 exemplifica como duas máquinas virtuais podem comunicar-se estando ou não hospedadas no mesmo servidor. Enquanto a comunicação de **MV3** com **MV1** e **MV5** com **MV4** é feita pelo hipervisor, a comunicação de **MV2** com **MV6**, além de passar pelo hipervisor, também passa pela rede física da infraestrutura do provedor.

Figura 5 – Comunicação entre máquinas virtuais estando ou não no mesmo servidor.



Fonte: O próprio autor.

Analisando a Figura 5, percebe-se que o hipervisor, neste cenário, não fornece apenas a virtualização de computadores (MVs), mas também precisa fornecer soluções para comunicação entre as MVs, uma vez que nem sempre haverá um recurso

de rede físico entre a MV de origem e destino de uma comunicação. Dessa forma, o tráfego gerado entre MV origem e destino não é comutado para a rede física, então, o monitoramento e filtro de pacotes não pode acontecer de um modo que não comprometa a segurança desse ambiente virtualizado (CALLEGATI et al., 2014). Neste contexto, a identificação do tráfego entre instâncias em um mesmo servidor, como mostrado na Figura 5 (comunicação entre MV1 - MV3 e MV4 - MV5), é dificultada em função das camadas de abstração adicionadas pela virtualização da infraestrutura física existente.

2.3 RECURSOS DE REDE VIRTUALIZADOS

A virtualização de recursos de comunicação é amplamente explorada em ambientes de computação em nuvem (CHOWDHURY; BOUTABA, 2010). A Figura 5 representa a comunicação entre máquinas virtuais, hospedadas ou não em um mesmo servidor. A comunicação entre as MVs pode envolver diversos elementos virtualizados, como roteadores, *switches*, interfaces de rede, *bridges* e endereços *Media Access Control* (MAC), bem como tecnologias tradicionais como VLAN e tunelamento *Internet Protocol* (IP).

Em paralelo, o gerenciamento dos recursos de comunicação, virtualizados ou não, pode ser realizado com o uso de *Software Defined Networking* (SDN) (Kreutz et al., 2015; MCKEOWN et al., 2008). A separação entre os planos de controle e de gerenciamento introduzida por SDN permitiu a composição de infraestruturas virtuais privadas, gerenciadas pelos usuários. É importante ressaltar que a virtualização de recursos computacionais está presente no oferecimento de serviços aos usuários finais, bem como no gerenciamento da nuvem.

Por outro lado, a virtualização de recursos de rede mostra-se fortemente relacionada à implementação do hipervisor e a capacidade deste em usar outras tecnologias/soluções (*e.g.*, *Open vSwitch*, OpenFlow, *etc.*) de modo integrado e facilmente interoperável. Assim, o processo de monitorar e analisar tráfego de rede em nuvens computacionais possui uma complexidade ampliada visto que métodos tradicionais (*e.g.*, configurar portas de eco em *switches*, colocar interfaces de rede em modo promíscuo, *etc.*) podem não coletar todo o tráfego.

2.4 OPENSTACK

O OpenStack[3] é um conjunto de *software* e ferramentas de código aberto para criar e gerenciar nuvens computacionais públicas e privadas do tipo IaaS. De acordo com o OpenStack Project (OPENSTACK, 2019f), trata-se de um sistema operacional

[3] <https://www.openstack.org>

para nuvens, que controla um amplo grupo de recursos de computação, armazenamento e rede por todo o *data center*. Atualmente, o OpenStack está na sua 21ª versão (codinome Ussuri).

O OpenStack foi criado a partir de uma colaboração entre Anso Labs (contratada pela NASA) e Rackspace, no início de 2010. O projeto foi oficialmente anunciado em Julho de 2010 e, ao longo do tempo, tornou-se uma solução de nuvem amplamente utilizada no mundo devido à quantidade de empresas envolvidas no seu desenvolvimento (OPENSTACK, 2019c). Tal fato fez com que as principais soluções de MVs (*e.g.*, KVM, QEMU, VMware ESX/ESXi, Citrix Xen, Microsoft Hyper-V, UML, Virtuozzo, IBM zVM) fossem incorporadas aos seus serviços de computação. Da mesma forma foram incorporados serviços de virtualização de comunicação (*e.g.*, SDN com OpenFlow) e armazenamento (*e.g.*, Gluster, CEPH).

O OpenStack opera de maneira distribuída, podendo separar todos os seus serviços pela infraestrutura da nuvem. Neste sentido, existem serviços considerados básicos para funcionamento da nuvem (*e.g.* Keystone, Nova e Glance), serviços de apoio (*e.g.*, MySQL e RabbitMQ) e opcionais (*e.g., Trove*).

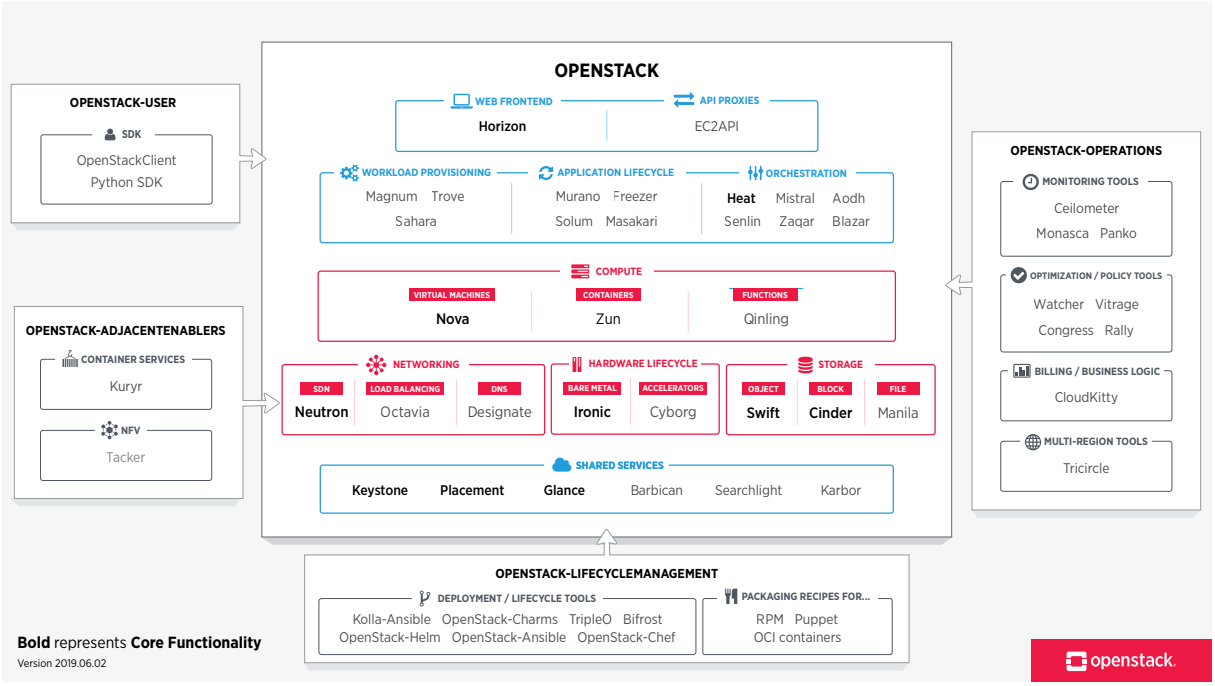
2.4.1 Serviços OpenStack

Os serviços do OpenStack, usados para o provisionamento de nuvens computacionais, são separados por módulos. Além disso, é possível que módulos opcionais sejam acoplados a uma instalação. A Figura 6 elenca os módulos de serviço do OpenStack, bem como serviços e recursos externos que podem ser incorporados a solução de nuvem.

Os principais módulos do OpenStack são implementados em Python, integrados com *shell script*, e são acessíveis via APIs baseadas em *Representational State Transfer* (REST). Além disso, a interface *web* permite gerenciar as funcionalidades do OpenStack (Scharf et al., 2015). Analisando a Figura 6, as funcionalidades essenciais do OpenStack são: Horizion, Heat, Nova, Neutron, Ironic, Swift, Cinder, Keystone, Placement e Glance. Dentre estas, os módulos comumente abordados ao iniciar uma instalação básica do OpenStack são:

- **Horizon**: Disponibiliza um serviço de *dashboard web* para uso e administração da nuvem;
- **Keystone**: Disponibiliza um serviço de autenticação e autorização para outros serviços do OpenStack;
- **Nova**: Responsável pela distribuição e gerenciamento das instâncias. Realiza tarefas como iniciação, escalonamento e desalocação de MVs;

Figura 6 – Visão geral dos módulo de serviço disponíveis no OpenStack.



Fonte: (OPENSTACK, 2019a)

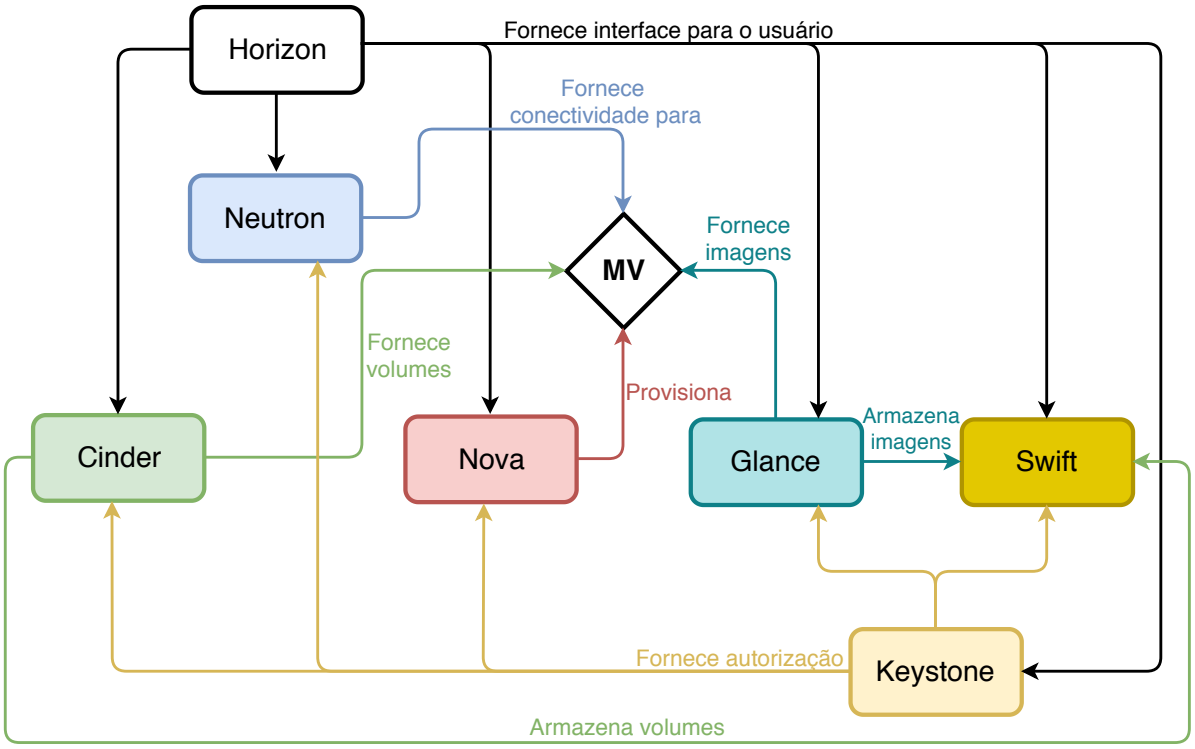
- **Neutron**: Fornece conectividade de rede entre os outros serviços, bem como disponibiliza uma API para que os consumidores configurem suas redes;
- **Glance**: Atua no processo de armazenamento e recuperação das imagens utilizadas nas MVs;
- **Swift**: Responsável pelo armazenamento e recuperação de objetos não estruturados. Usa técnicas de replicação de dados para tolerância de falhas; e
- **Cinder**: Provê armazenamento persistente em bloco para instâncias em execução.

Neste sentido, é necessário que exista harmonia entre os módulos do OpenStack para que todas as operações na nuvem (não apenas com MVs) sejam executadas corretamente. Em implantações do OpenStack em cenário de produção, os módulos são distribuídos entre os servidores da nuvem de modo que existam ganhos com desempenho e segurança da nuvem. Ou seja, serviços computacionalmente intensivos são posicionados em servidores distintos, buscando diminuir a interferência no desempenho final do OpenStack. Já em implantações menores, como em uma prova de conceito, os módulos acabam centralizados em um mesmo servidor e os demais servidores são destinados ao serviço de computação (*e.g.*, Nova). A documentação do OpenStack fornece exemplos de diversas arquiteturas de como distribuir serviços específicos do OpenStack para criar uma infraestrutura de nuvem voltada a um serviço

específico (*e.g.*, servidores *web*, contêineres e *streamming* de vídeo) (OPENSTACK, 2019b).

A Figura 7 relaciona os principais módulos do OpenStack, mostrando como interagem entre si para fornecer recursos/serviços que possibilitem o uso de MVs. Pode-se concluir que, de acordo com a definição da instalação da nuvem, alguns deste módulos podem estar em um mesmo servidor ou em servidores distintos.

Figura 7 – Principais operações dos módulo sobre uma MV.

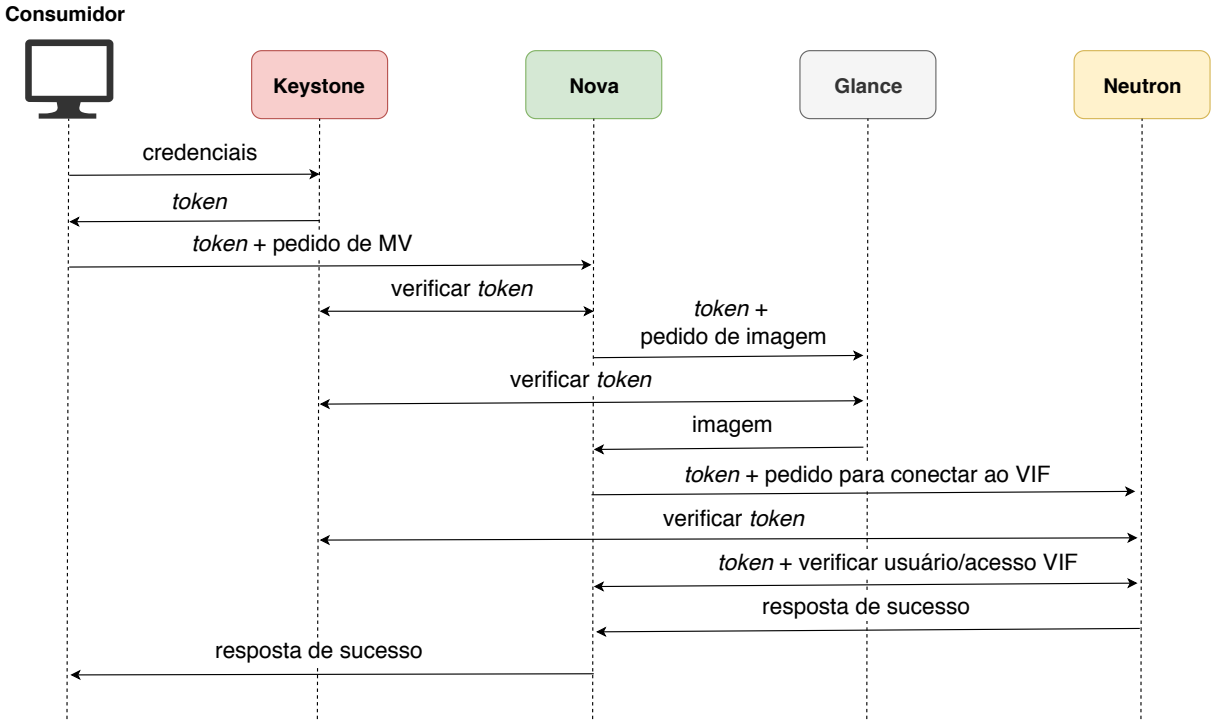


Fonte: O próprio autor.

Como apresentado na Figura 7, o funcionamento das MVs depende da integração de vários serviços e componentes do OpenStack. Além disso, essa integração de serviços faz com que o usuário possa controlar o ciclo de vida das instâncias das MVs, podendo criá-la, suspendê-la, reativá-la, ou realizar outras operações.

A Figura 8 traz uma visão simplificada do processo de criação de instâncias de MVs no OpenStack; todavia, é possível entender (mesmo que de alto nível) as principais etapas para criação de MVs: autenticação do usuário na nuvem; busca pela imagem da instância; resposta de criação. Neste sentido, a Figura 9 ilustra o mesmo processo de criação de instâncias, mas, com operações e componentes internos mais detalhados.

Figura 8 – Diagrama de sequência simplificado da criação de MVs.

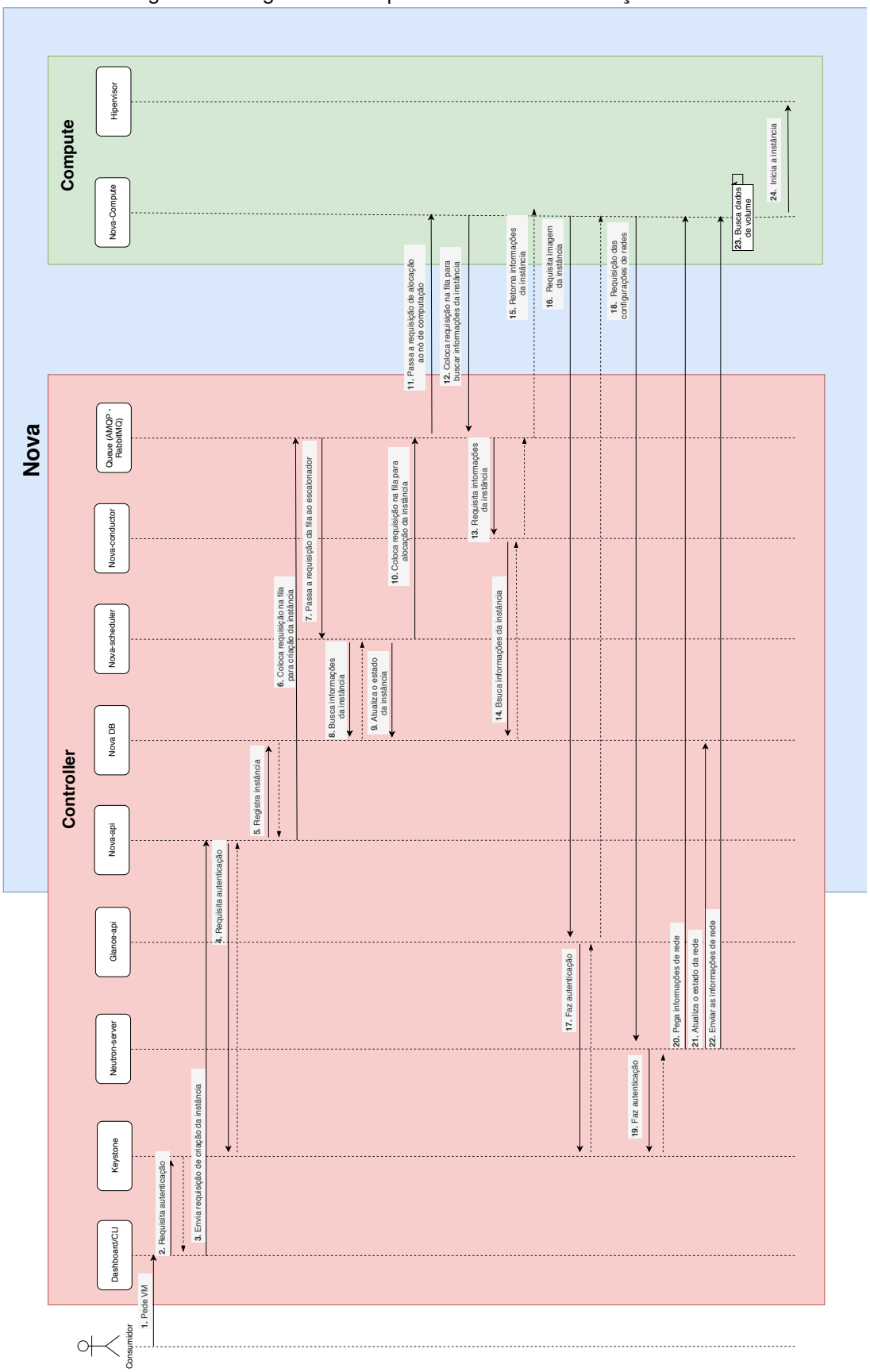


Fonte: O próprio autor.

De acordo com a Figura 9, os passos envolvidos no fluxo para criação de instâncias são (OPENSTACK, 2018d):

1. **Consumidor** faz o pedido de criação da MV (via **Dashboard** ou ***Command Line Interface* (CLI)**);
2. **Dashboard** ou **CLI** usam as informações do usuário consumidor para fazer uma chamada REST ao **Keystone** para autenticação. **Keystone** faz o processo de autenticação, cria e envia um *token*, que será usado em chamadas REST para outros componentes;
3. **Dashboard** ou **CLI** convertem a requisição de nova instância especificada no formato *"launch-instance"* ou *"nova-boot"* para REST API e enviam ao **nova-api**;
4. **nova-api** recebe a requisição e a envia ao **Keystone** para validação do *token* de autenticação e permissões de acesso. **Keystone** valida o *token* e envia uma resposta com as devidas permissões e papéis do usuário;
5. **nova-api** interage com o **nova-db** para criar os primeiros registros da nova instância;

Figura 9 – Diagrama de sequência detalhado da criação de MVs.



Fonte: Adaptado de: (OPENSTACK, 2018d).

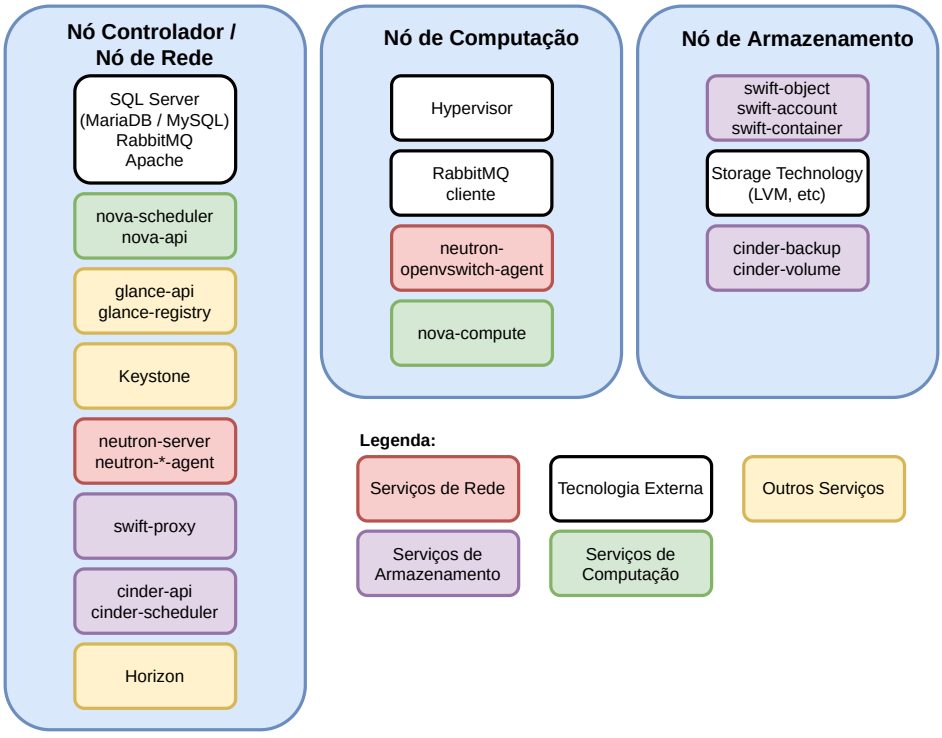
6. **nova-api** faz uma chamada *Remote Procedure Call* (RPC) ao **nova-scheduler** para atualizar o registro da instância com o *ID* do servidor. Essa chamada RPC é gerenciada pelo **RabbitMQ** (<www.rabbitmq.com>), que é responsável pela comunicação entre os serviços do Nova;
7. A chamada RPC é repassada da fila (**RabbitMQ**) ao **nova-scheduler**;
8. **nova-scheduler** interage com o **nova-database** para buscar informações (estado dos nós de computação) escolher um servidor;
9. **nova-scheduler** atualiza o estado da instância com o ID do servidor escolhido;
10. **nova-scheduler** faz uma chamada RPC ao **nova-compute** para subir a instância no servidor escolhido;
11. **nova-compute** recebe a requisição através do **RabbitMQ**;
12. **nova-compute** busca as informações da instância através do **nova-conductor** (chamada RPC);
13. **nova-conductor** obtém a requisição da fila;
14. **nova-conductor** busca as informações da instância no **nova-database**;
15. **nova-compute** busca as informações da fila;
16. **nova-compute** solicita a imagem ao **glance-api** através de uma chamada REST;
17. **glance-api** repassa o *token* de autorização ao **Keystone**. Após a validação, com o endereço da imagem, que é fornecido pelo **glance-api**, a imagem é buscada no repositório de imagens;
18. **nova-scheduler** requisita a configuração de rede da instância ao **neutron-server**
19. **neutron-server** valida *token* com o **Keystone**;
20. **neutron-server** obtém as informações de rede com o **nova-compute**;
21. **neutron-server** atualiza o estado da rede da instância no **nova-database**;
22. **neutron-server** passa informações da rede ao **nova-compute**;
23. **nova-compute** busca dados dos volumes via REST (cinder-api). São feitas as configurações de volume através de interações com o Cinder; e
24. **nova-compute** inicia a instância.

Com a Figura 9, nota-se a constante comunicação entre os serviços do OpenStack no processo de criação de MVs, principalmente entre os componentes do Nova, que podem estar distribuídos em vários servidores. Além disso, destaca-se o uso do *RabbitMQ* na implantação do *Advanced Message Queuing Protocol* (AMQP), que via chamadas RPCs, gerencia as filas e trocas de mensagens entre as diferentes aplicações envolvidas no processo, como entre o escalonador (*nova-scheduler*) e o *Nova-Compute* (OPENSTACK, 2020d).

2.4.2 Topologia OpenStack

As comunicações inter-serviço no OpenStack ocorrem independente da topologia na qual a nuvem foi configurada. Isto é, o OpenStack opera em um sistema altamente distribuído, no qual a troca de mensagens é essencial para o funcionamento da nuvem. Neste sentido, a configuração/instalação da nuvem pode ser feita de maneira muito flexível, sem a necessidade de seguir uma topologia específica. Dessa forma, as configurações mínimas tipicamente exigem apenas dois tipos de servidores, nos quais um exerce a função de controlador, com módulos como Keystone e Glance, e o outro para computação, tendo como base o Nova. De maneira geral, são usados diversos servidores para a finalidade de computação. Assim, é possível conferir capacidade computacional *i.e.*, quantidade de processadores/*cores* e memória principal) para criar MVs. Há também outras configurações, como a arquitetura com três servidores, apresentada na Figura 10.

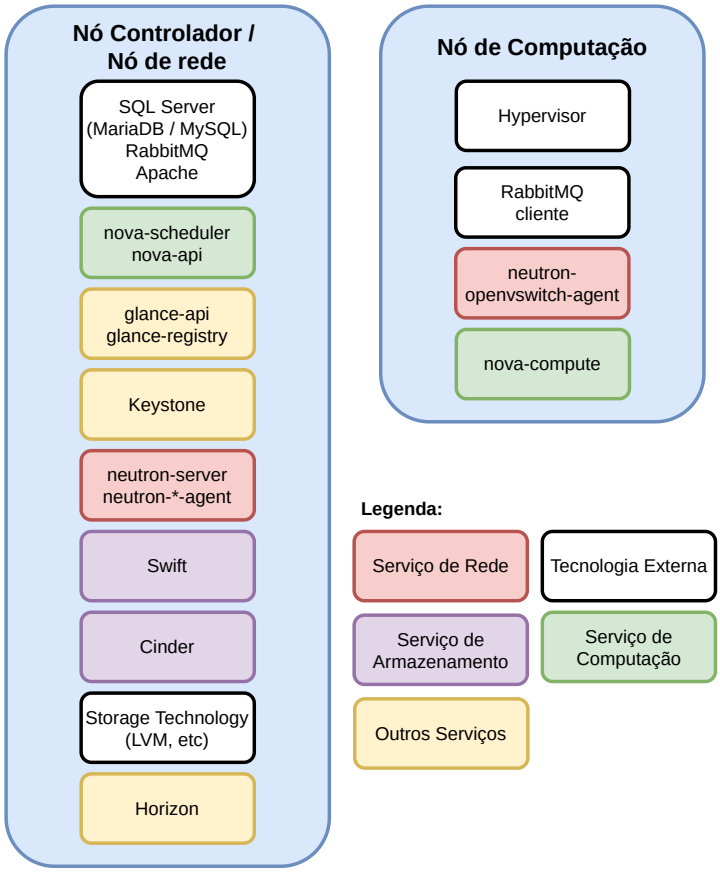
Figura 10 – Arquitetura básica do OpenStack.



Fonte: O próprio autor.

Na Figura 10 pode-se observar uma topologia com três tipos de servidores. Além disso, em implantações de produção, cada serviço básico do OpenStack pode ter um servidor dedicado. Então, questões como alta disponibilidade, desempenho, segurança e *Single Point of Failure* (SPF) tornam-se mais recorrentes nesses ambientes. Neste contexto, os serviços são distribuídos de acordo com suas funcionalidades. Por exemplo, no servidor Controlador e de Rede tem-se apenas serviços cujas funcionalidades são voltadas para o controle ou rede da nuvem, como é o caso do *neutron-server* (rede) e do Keystone (autenticação e autorização). O OpenStack faz uso de tecnologias externas para: comunicação entre aplicações internas, feita com o RabbitMQ; gerenciamento de volumes lógicos, com o *Logical Volume Manager* (LVM); servidor SQL, geralmente MariaDB ou MySQL; e virtualização, com hipervisores como QEMU e KVM.

Figura 11 – Arquitetura mínima do OpenStack.



Fonte: O próprio autor.

Tanto na Figura 10 quanto na Figura 11 observa-se que os serviços em cada servidor podem ser classificados em:

- **Serviço de Rede:** no OpenStack, o Neutron é o módulo responsável por toda conexão da nuvem. Assim, os componentes nesta categoria são, de maneira geral, serviços do Neutron espalhados pelos servidores da nuvem;

- **Serviço de Computação:** de maneira geral, serviços que fazem parte do módulo Nova. No controlador estão os componentes para acesso à API e ao escalonador (nova-scheduler), enquanto no servidor de computação tem-se o nova-compute, que interage diretamente com as instâncias de máquinas virtuais.
- **Serviço de Armazenamento:** os dois principais módulos de armazenamento disponíveis no OpenStack são o Swift e o Cinder (armazenamento de objetos não estruturados e persistência em bloco, respectivamente). Em cenários nos quais um servidor para armazenamento é criado (Figura 10), os componentes desses módulos são distribuídos pelos servidores de controle e armazenamento.
- **Tecnologia Externa ao OpenStack:** componentes que não são implementados pelo OpenStack, mas são utilizados em conjunto com os demais serviços e tecnologias;
- **Outros Serviços:** engloba os demais serviços presentes na configuração da nuvem. Por exemplo, serviço de interface (Horizon ou CLI), autenticação na nuvem (Keystone) e gerenciamento de imagens (Glance).

Em relação à Figura 11, tem-se uma configuração com dois servidores. Em comparação com a topologia mostrada na Figura 10, este modelo de configuração do OpenStack migra os serviços de armazenamento para dentro do servidor de controle. Além disso, os serviços Cinder e Swift têm seus componentes agrupados no controlador. Já na arquitetura com três servidores, seus componentes são distribuídos pelo nó de armazenamento e controlador. De maneira geral, as topologias que têm servidores dedicados aos serviços primários do OpenStack tendem a apresentar maior escalabilidade. Por outro lado, a complexidade desses ambientes altamente escaláveis tende a ser muito maior. Dessa forma, pela flexibilidade de configuração fornecida pelo OpenStack, conclui-se que a topologia cuja instalação da nuvem será feita depende especificamente do propósito dela. Ou seja, o planejamento da topologia deve levar em consideração fatores como o nível de escalabilidade que se deseja atingir, desempenho, disponibilidade e segurança.

Neste contexto de sistema distribuído, os módulos do OpenStack trabalham em conjunto via requisições REST e serviços de mensageria, como o RabbitMQ. No processo de criação de uma MV, por exemplo, diversos módulos são envolvidos para prover alocação do nó hospedeiro, imagem da MV, recursos de rede, entre outros. Neste sentido, é necessária a coordenação e troca de informações entre todos os módulos envolvidos no processo.

2.4.2.1 Comunicação entre os serviços OpenStack: REST

Os módulos OpenStack têm APIs REST expostas para uso na nuvem, que fornecem acesso às funcionalidades do módulo, *e.g.*, a API do Neutron permite requisições de criação, configuração e gerenciamento de redes e subredes. Além disso, a API do Keystone permite autenticação via *token* em todos os módulos. De um modo geral, com as APIs expostas dos módulos, é possível manter o trabalho conjunto para a realização de tarefas, *e.g.*, criação de instância de MV: módulo Nova gerencia nó hospedeiro; módulo Glance cria e fornece a imagem; e módulo Neutron cria e fornece redes e subredes. A organização e coordenação dessa troca de informações fica a cargo do serviço de mensageria, neste caso o RabbitMQ

Neste sentido, a comunicação via requisições REST viabiliza o uso de portas TCP bem definidas aos serviços/módulos OpenStack, *e.g.*, *nova-api*, com portas TCP/8773 e TCP/8775; e *glance-api*, com porta TCP/9292. Também é importante notar que existem serviços auxiliares, dos quais os serviços OpenStack dependem, *e.g.*, a Dashboard utiliza serviço *Hypertext Transfer Protocol* (HTTP) para comunicação não segura (OPENSTACK, 2020b). Diversos módulos OpenStack fazem uso de serviços de mensageria, por padrão disponibilizados pelo RabbitMQ, que faz uso do protocolo AMQP para comunicação entre serviços de um mesmo módulo e até mesmo entre serviços de módulos distintos. Por padrão, a porta TCP utilizada pelo RabbitMQ é a 5672.

2.4.2.2 Comunicação entre os serviços OpenStack: RabbitMQ

Serviços de mensageria, ou *Message Brokers*, são bem comuns em sistemas distribuídos. O OpenStack emprega esta tecnologia para comunicação interna de seus módulos/serviços. De maneira geral, os serviços de mensageria são responsáveis pela troca de mensagens assíncronas entre dois sistemas distintos. No caso do OpenStack, a troca de mensagens pode ocorrer entre serviços distintos dentro de um mesmo módulo (*e.g.*, *nova-scheduler*, serviço escalonador do Nova, comunicando-se com o *nova-compute*, serviço de computação do Nova, para que a instância seja inicializada no servidor escolhido) ou até mesmo entre serviços de módulos diferentes (*e.g.*, *neutron-server* passando configurações de rede da MV ao *nova-compute*).

Dois sistemas distintos podem trocar mensagens com intermédio de um *broker* de mensageria sem ao menos conhecerem um ao outro. A mensagem parte de um dos sistemas e é enviada ao *broker*. De acordo com as características da mensagem, *e.g.*, solicitação de algum tipo de recurso, o *broker* a encaminha a uma fila específica, na qual a mensagem aguarda até chegar ao sistema interessado (destinatário). Basicamente, trata-se de um mecanismo orientado a eventos para troca de mensagens entre sistemas distintos. Tradicionalmente, os *brokers* de mensageria, como o

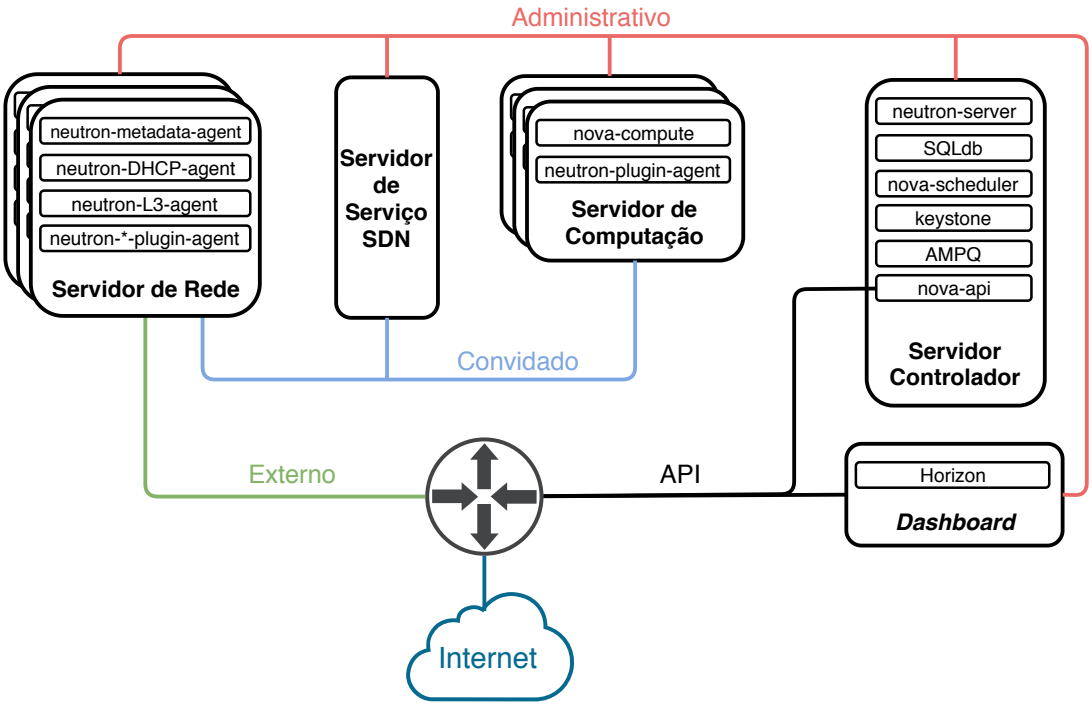
RabbitMQ, usam chamadas RPC e implementam o protocolo AMQP para fazer todo o gerenciamento das mensagens e filas.

No OpenStack, adota-se o modelo de publicação e inscrição (OPENSTACK, 2020a). Neste modelo, os dois principais atores envolvidos pelo intermédio do *broker* são os produtores e consumidores. Os consumidores assinam/inscrevem-se para atender a certos tipos de eventos. Ou seja, quando existem mensagens que se encaixam nos padrões os quais o consumidor mostrou interesse, essa aplicação consumidora será notificada. Neste sentido, os produtores são as aplicações que postam/enviam mensagens, enquanto que os consumidores buscam essas mensagens em filas especificas, selecionadas de acordo com implementações especificas do *broker*.

2.4.3 Arquitetura básica do OpenStack

Para realizar a comunicação entre os servidores que hospedam diferentes serviços são utilizadas múltiplas redes físicas e virtuais. Neste sentido, existem três domínios criados de acordo com políticas de segurança distintas (OPENSTACK, 2019f): Domínio Público, Domínio de Controle e Domínio de Convidados. A Figura 12 exemplifica a infraestrutura de comunicação recomendada como padrão pelo OpenStack. Podem ser utilizadas técnicas de virtualização de redes, como VLAN, para separação dos domínios de rede.

Figura 12 – Configuração de rede recomendada pelo OpenStack (OPENSTACK, 2019a).



Fonte: O próprio autor.

Domínio Público: são englobadas as redes Externa e de API, que são responsáveis pela visibilidade da API da nuvem criada na Internet e do acesso à Internet pelas

MVs. Todos os endereços IPs das redes devem ser visíveis a partir da Internet para seu pleno funcionamento (OPENSTACK, 2019f).

Domínio de Convidados é formado pela rede de comunicação entre as MVs. Os consumidores do serviço de oferta de MVs são referenciados como convidados por não possuírem vínculo direto com a administração da nuvem, logo, não há garantias sobre quem estes convidados são e quais os seus objetivos. Com o serviço Neutron, são utilizadas tecnologias de virtualização de rede tanto para isolar o tráfego entre as diversas MVs hospedadas em um servidor, quanto para criar redes virtuais entre elas e tornar possível a comunicação quando assim configurado pelo consumidor (OPENSTACK, 2019f).

Domínio de Controle: é formado principalmente pela rede responsável pelo tráfego de controle do OpenStack, que corresponde ao tráfego gerado pela comunicação entre os seus serviços. Esta é a rede mais interna da nuvem, sendo que somente os administradores devem acessá-la, já que se trata do gerenciamento da nuvem. Portanto, esta rede dispensa o uso de técnicas de virtualização da rede. Tanto os servidores de controle quanto os de computação devem ter uma interface física de rede que se conecta com a rede de controle. O uso de múltiplas interfaces de rede permite a conectividade à rede de controle e às outras redes conforme o caso, como a rede de convidados no caso dos servidores de computação. Logo, em casos normais, uma MV em execução nunca poderá acessar a rede de controle (KRUTZ; VINES, 2010).

A maior parte das pesquisas preocupa-se em analisar as redes da nuvem do ponto de vista dos usuários (*e.g.*, Domínio de Convidados), deixando de fora toda a parte de operações internas e comportamento do provedor da nuvem (Aishwarya. K; Sankar, 2015; Chen; Zhao, 2012; Shete; Dongre, 2017). O dimensionamento e controle de tráfego na rede administrativa de uma nuvem OpenStack depende da estratificação de uso e, consequente impacto de operações de controle oriundas de demandas das ações dos usuários. A ausência de monitoração ou caracterização de tráfego na rede administrativa de uma nuvem OpenStack pode causar colapsos e gargalos na rede degradando a qualidade ou mesmo gerando indisponibilidade do serviço.

2.5 CARACTERIZAÇÃO DE TRÁFEGO

A classificação de tráfego não é um tópico de pesquisa novo. Inclusive, vem acompanhando o nascimento e crescimento da Internet (Finsterbusch et al., 2014). Neste contexto, a caracterização de tráfego tem sido uma tarefa de considerável importância na área de gerenciamento e segurança de redes. Dessa forma, com o uso de técnicas para classificação/caracterização de tráfego, podem ser alcançados benefícios como o aumento na precisão para alocação de recursos de redes. Diversas

tarefas relacionadas ao gerenciamento de redes podem ser melhoradas, como *Quality of Service* (QoS) e segurança de redes (Finsterbusch et al., 2014). Portanto, a caracterização de tráfego também é uma tarefa usada a fim de entender e resolver problemas relacionados ao desempenho em redes de computadores (DAINOTTI; PESCAPE; VENTRE, 2006).

No contexto de nuvens computacionais, a análise e a caracterização de tráfego mostram-se como uma importante abordagem no levantamento de informações relativas tanto à segurança quanto ao desempenho da nuvem. Com a caracterização de tráfego pode-se, *e.g.*, identificar operações em nível de camada de rede bem como o que está trafegando e sua finalidade. Embora esta técnica ainda seja incipiente no contexto da parte de controle em nuvens computacionais, quando se trata de caracterização de tráfego de aplicações tradicionais, como em servidores web, por exemplo, torna-se amplamente empregada (WILLIAMSON, 2001; BRAUN; CLAFFY, 1995; GILL et al., 2007).

As nuvens computacionais em geral têm dois locais nos quais a monitoração e análise do tráfego podem ser empregadas. Em primeiro lugar, a rede dos usuários pode ser monitorada a fim de aplicar controles de privacidade e uso da nuvem. Além disso, também é possível monitorar o tráfego administrativo da nuvem. Desse modo, alguns recursos, tanto computacionais quanto de redes e comunicação, podem ser melhor dimensionados de acordo com o tráfego administrativo analisado, *e.g.*, aumentar a largura de banda quando necessário.

De maneira geral, o estudo do tráfego é separado em duas etapas: medição e análise (DAINOTTI; PESCAPE; VENTRE, 2006). Na primeira, tem-se a coleta de dados que trafegam na rede, enquanto na segunda, é feita uma análise do tráfego para identificar características relevantes ao problema.

Durante a etapa de medição de tráfego, podem ser utilizadas diversas ferramentas para capturar os dados que trafegam pela rede (*e.g.*, TCPdump e SNORT). Além disso, dependendo de como é implementada, a medição pode ser classificada em duas categorias (VILELA, 2006; WILLIAMSON, 2001):

- **Ativa**: os sistemas de monitoramento ativo operam com a inserção de tráfego na rede. O propósito da inserção de tráfego está na extração de informações sobre características da rede, como largura de banda disponível, gargalos e atrasos de ida e volta. Neste sentido, o tráfego que é inserido para sondagem também compete por recursos da rede, o que acaba sendo conflitante com o fluxo padrão. Então, embora o monitoramento ativo forneça um método preciso para determinar as características da rede, a sua escalabilidade é severamente afetada pela disputa de recursos da rede (Landfeldt; Sookavatana; Seneviratne, 2000); e

- **Passiva**: os problemas de escalabilidade do tipo ativo associados a sobrecarga e atrasos não estão presentes no monitoramento passivo, já que este tipo de monitoramento apenas monitora os fluxos em progresso na rede. Através da observação do fluxo da rede, os sistemas de monitoramento passivo inferem o *status* da rede ao invés de injetar tráfego para sondagem (Landfeldt; Sookavatana; Seneviratne, 2000).

Uma abordagem de monitoramento passivo amplamente utilizada, principalmente por ser altamente escalável, é a exportação de fluxos (*flow export*). Nesta abordagem, os pacotes são agregados em fluxos e exportados para armazenamento e análise. De acordo com a RFC 7011 (Claise; Trammell; Aitken, 2013), os fluxos, neste caso, são definidos como um conjunto de IPs que passam por um determinado ponto de observação na rede dentro de um certo intervalo de tempo. Os IPs pertencentes a um determinado conjunto devem ter características em comum, como IPs e portas de origem e destino, conteúdo do pacote e metainformação. A *Deep Packet Inspection* (DPI) é frequentemente relacionada ao *flow export*, contudo, refere-se ao processo de análise da carga dos pacotes. Neste sentido, é possível observar duas principais diferenças entre as abordagens: (1) *flow export* tradicionalmente considera apenas o cabeçalho dos pacotes, então, é menos sensível à privacidade que DPI; e (2), *flow export* baseia-se na agregação dos pacotes, enquanto DPI tradicionalmente considera pacotes individuais (Hofstede et al., 2014).

Na etapa de análise também aplicam-se diversas técnicas de análise de tráfego, como a modelagem estatística simples, que permite analisar atributos genéricos do tráfego e correlacioná-los com as aplicações correspondentes que estão em execução (NGUYEN; ARMITAGE, 2008). Neste sentido, são utilizados atributos como as portas de origem e destino dos pacotes, que indicam o serviço correspondente a estes pacotes. Adicionalmente, é importante notar que o processo de caracterização de tráfego é consideravelmente variável de acordo com o ambiente (configuração de rede).

2.5.1 Técnicas para classificação de tráfego

Iniciando com as abordagens baseadas em portas, as metodologias para classificação de tráfego de rede evoluíram consideravelmente ao longo do tempo. Assim, chega-se às abordagens como a utilização de modelagem estatística, por exemplo, que caracteriza os pacotes sem violar restrições legais e de privacidade dos usuários (Finsterbusch et al., 2014). Neste sentido, as técnicas mais significantes utilizadas na classificação de tráfego da Internet são:

- **Baseada em portas (*Port-based*)**: É o método mais comum para classifica-

ção de tráfego. Consiste na análise das portas de comunicação do cabeçalho TCP/UDP, de modo a criar uma associação com as aplicações referentes a elas. Um dos principais fatores que motivam o uso deste método é a facilidade de acesso aos números das portas. De maneira geral, não há problemas com criptografia nem privacidade de dados, o que agiliza o processo de classificação. Além disso, este método geralmente é empregado em regras de *firewall* e *Access Control List*s (ACLs). Em contrapartida, nem todos os protocolos podem ser classificados através deste método. Uma vez que protocolos como *Peer-to-Peer* (P2P) podem usar portas aleatórias e algumas aplicações podem usar portas associadas a outros protocolos. Por fim, o tradicional método baseado em portas ainda é a técnica mais simples, rápida e frequentemente usada quando a precisão não é um requisito crítico (Dainotti; Pescape; Claffy, 2012; Finsterbusch et al., 2014);

- **Estatística**: Esta abordagem faz uso de parâmetros independentes da carga do pacote, tais como tamanho, tempo entre chegadas e duração de fluxo de pacotes. Assim, o método estatístico tem aplicação mais abrangente que os demais métodos que requerem acesso ao *payload* do pacote, já que em determinados cenários, o acesso a essa carga é restrito (Dainotti; Pescape; Claffy, 2012; Finsterbusch et al., 2014; PISKAC; NOVOTNY, 2011);

- **Correspondência de padrões**: Esta técnica tem como base a DPI, que é recorrente tanto na classificação de tráfego quanto na implementação de *Network Intrusion Detection Systems* (NIDS). Neste sentido, é possível comparar os conteúdos dos pacotes com um conjunto de regras pré-montadas. Além disso, para evitar problemas com a flexibilidade das regras, que geralmente são feitas no formato de *string*, são utilizadas expressões regulares (*RegEx*), que fornecem maior escalabilidade à inspeção do conteúdo dos pacotes (Dainotti; Pescape; Claffy, 2012; Finsterbusch et al., 2014; Lin et al., 2006); e

- **Decodificação de protocolo**: O método de decodificação de protocolo baseia-se na reconstrução com estado das sessões e informações das aplicações obtidas através do conteúdo dos pacotes. A identificação dos protocolos é feita de acordo com características dos cabeçalhos dos protocolos e sequências de pacotes. Além disso, também é feita uma análise do comportamento do protocolo, que geralmente é modelado a partir de uma máquina de estados. No geral, esta abordagem apresenta alta precisão e poucos falsos negativos, mas tem maior complexidade na etapa de desenvolvimento (Finsterbusch et al., 2014; Risso et al., 2008).

A Tabela 1 apresenta uma comparação das técnicas usadas para classificação de tráfego. A comparação é feita de acordo com o nível de aplicabilidade e problemas

de cada uma. Uma avaliação e comparação mais rigorosa das técnicas para classificação do tráfego requer uma padronização de testes e procedimentos, bem como uma definição de métricas a serem levadas em consideração. Tipicamente, a métrica mais aceita nestes cenários é a precisão geral do método. Neste sentido, a precisão aqui refere-se ao número de objetos corretamente classificados em relação ao total (Dainotti; Pescape; Claffy, 2012).

Tabela 1 – Técnicas para classificação de tráfego com base na aplicabilidade e problemas.

Técnicas	**Aplicabilidade**	**Problemas**
Baseada em Portas	Alta	-Requer conhecimento sobre a rede -Conflito de portas -Portas não registradas no IANA
Método estatístico	Alta	-Variabilidade das características dos pacotes de acordo com o cenário -Protocolos com características parecidas
Correspondência de padrões	Média	-Criptografia -Privacidade -Recursos computacionais
Decodificação de protocolo	Baixa	-Alta complexidade -Manter atualizações -Criptografia e protocolos proprietários

De acordo com a Tabela 1, nota-se que o método estatístico, juntamente com o método baseado em portas, são os métodos com maior aplicabilidade geral, mesmo que não apresentem alta precisão em todos os casos. Em relação ao método baseado em portas, sua alta aplicabilidade se dá devido a simplicidade e eficácia de execução. Esta abordagem mostra-se bastante útil, principalmente em cenários nos quais há conhecimento prévio sobre a rede e protocolos ali utilizados. Os principais problemas relacionados à classificação de tráfego baseada em portas são os conflitos de portas entre diferentes aplicações e protocolos cujas portas não são registradas no IANA.

Nos métodos de classificação estatísticos, como não é feita inspeção de pacotes, não há problemas com privacidade e criptografia. Neste sentido, a aplicabilidade deste tipo de método é alta, embora existam problemas principalmente com a variabilidade de parâmetros independentes da carga do pacote. Já na correspondência de padrões, tem-se uma alta precisão, já que o método utiliza inspeção de pacotes. Contudo, a aplicabilidade da correspondência de padrão não é tão alta devido aos problemas com criptografia, privacidade e recursos computacionais necessários. Por fim, os métodos que utilizam a decodificação dos protocolos têm uma aplicabilidade baixa, que é justificada pelo grau de complexidade de implementação, quantia de protocolos existentes (mantimento da ferramenta atualizada) e inviabilidade de uso quando há criptografia.

2.6 CONSIDERAÇÕES PARCIAIS

A computação em nuvem é um modelo que permite acesso a recursos computacionais de maneira conveniente, ubíqua e escalável. Neste sentido, são envolvidos atores como o consumidor, que utiliza os serviços da nuvem, e o fornecedor, que é a parte responsável por disponibilizar o serviço de computação em nuvem. Além disso, é apresentado o OpenStack, um sistema operacional para nuvens computacionais. O OpenStack usa a computação distribuída juntamente com tecnologias como a virtualização para criar ambientes IaaS. Neste contexto, a caracterização de tráfego mostra-se incipiente no que diz respeito à infraestrutura da nuvem, mais especificamente em seu domínio administrativo. Por fim, nota-se que existem diversos métodos para que seja feita esta análise e caracterização. Pela simplicidade e eficácia da abordagem baseada em portas, esta técnica mostra-se altamente aplicável em ambientes nos quais há conhecimento prévio da rede e das aplicações ali utilizadas, como é o caso da rede administrativa do OpenStack. Contudo, uma parcela do tráfego, que é gerada pelo RabbitMQ, a partir da comunicação entre serviços, torna o cenário desafiador à aplicação da abordagem baseada em portas. Uma vez que o RabbitMQ faz uso de chamadas RPC, não mantendo portas predefinidas para cada serviço. Neste sentido, é necessário mapear as portas em uso pelo RabbitMQ e a quais serviços essas portas estão dedicadas.

3 CICLO DE VIDA DAS MÁQUINAS VIRTUAIS EM OPENSTACK

A criação e uso de instâncias de MVs no OpenStack aparenta ser simples aos usuários, mas na realidade, o processo de provisionamento de instâncias é complexo e envolve diversos componentes da nuvem. Além disso, a nuvem também precisa tratar as funcionalidades disponíveis para MVs, como suspender a instância, criar *backups*, entre outras atividades. Portanto, para dar suporte a todas estas operações, é necessário manter organização na nuvem, tanto em nível de rede (tráfego), quanto em nível de tarefas computacionais.

3.1 GERENCIAMENTO DE MVS NO OPENSTACK

O gerenciamento de instâncias de MVs pode ser feito tanto por administradores quanto por usuários gerais. O nível de permissão que um usuário vai ter sobre uma MV depende das configurações do projeto no qual ele está inserido. Também, caso o usuário seja dono da MV, algumas permissões adicionais podem ser concedidas. Além disso, algumas operações mais de baixo nível ou que se referem a elementos mais internos da nuvem podem ser limitadas a uso exclusivo dos administradores (*e.g.* migrar uma instância de um servidor para outro).

O OpenStack oferece uma lista de ações que podem ser executadas tanto por administradores quanto por usuários gerais para gerenciamento das instâncias de MVs. Na documentação fornecida pelo OpenStack, essas ações são chamadas de *Server Actions* (OPENSTACK, 2019d):

- **Reboot:** função usada para realizar *soft* ou *hard reboot* de uma instância. Com o *soft reboot*, o sistema operacional da instância manuseia a reinicialização da instância, isto é, todos os processos em execução no momento são terminados pelo próprio sistema operacional. Já o *hard reboot* é equivalente a um corte de energia na instância, então, os processos em execução não são encerrados corretamente pelo sistema operacional;
- **Rebuild:** faz uma reconstrução da instância de MV. Todos os dados na instância são removidos e uma nova imagem é especificada. São mantidos o ID, IP e *flavor* da instância;
- **Evacuate:** quando um serviço *nova-compute* fica *offline*, não há como reportar o estado atual das instâncias que estavam no servidor. Assim, estas instâncias são listadas como ativas para sempre. Para contornar esse problema, o **evacuate**

faz com que os administradores possam reconstruir essas instâncias em outro servidor;

- **Resize:** esta função é usada para converter uma instância em um diferente *flavor* (aumentar ou reduzir recursos). É possível reverter a operação dentro de um certo período de tempo, já que a instância original é armazenada neste tempo. Ao confirmar a operação de **resize**, a instância antiga é removida;
- **Pause, Unpause:** pausa uma instância e salva o seu estado na memória RAM. Uma instância que se encontra em **pause** continua a sua execução, mas em um estado congelado. **Unpause** retorna a instância ao seu estado ativo;
- **Suspend, Resume:** ao suspender uma instância, o seu estado e memória são salvos em disco e a MV é parada. Similar ao processo de hibernação em dispositivos gerais. Os recursos utilizados pela instância (memória e vCPU) ficam disponíveis para criação de novas instâncias. A operação **Resume** faz com que a instância retorne para seu estado ativo;
- **Snapshot:** cria uma imagem do estado atual do disco raiz da instância e a deixa disponível no repositório de imagens do Glance. Posteriormente, é possível iniciar instâncias com esta imagem;
- **Backup:** mesmo processo que o **snapshot**, contudo, as imagens salvas servem como cópia de segurança e, os *snapshots* mais antigos são removidos;
- **Start:** liga a instância;
- **Stop:** desliga a instância;
- **Delete, Restore:** desliga, desaloca todos os recursos da instância e então a exclui. Dependendo das configurações, é possível restaurar a instância (**restore**) dentro de um certo período de tempo;
- **Shelve, Shelve offload, Unshelve:** a operação **shelve** desliga a máquina, liberando recursos como memória RAM em uso, faz uma **snapshot** (se o *boot* for feito com uma imagem) e libera demais recursos usados pela MV. O **snapshot** é armazenado e a instância pode ficar disponível novamente com o **unshelve**. De modo geral, fazer o **shelve** de uma instância significa que esta não será necessária e, por isso, poderá ser removida dos hipervisores (**shelve offload** faz essa remoção). Dependendo das configurações, a operação **shelve** pode não remover as imagens dos hipervisores imediatamente, que torna a operação **unshelve** menos custosa e mais rápida enquanto as imagens ainda estiverem disponíveis. Contudo, os recursos alocados para aquela instância não serão liberados durante este período de tempo;

- **Lock, Unlock:** a operação **lock** bloqueia a execução de determinadas ações para usuários não administradores. **Unlock** desbloqueia essas ações. A lista completa de ações afetadas por **lock** e **unlock** pode ser consultada na documentação do OpenStack (OPENSTACK, 2019d);
- **Rescue Unrescue:** a ação **rescue** permite que uma instância seja iniciada em modo de recuperação a partir de uma imagem especial. Esta operação é utilizada para tentar recuperar um sistema operacional falho. Em relação ao **unrescue**, trata-se do inverso da operação **rescue**. A instância criada a partir do **rescue** será excluída;
- **Set administrator password:** adiciona a senha para root/administrador da instância de MV;
- **Migrate, Live migrate:** administradores, tipicamente, podem fazer uso da operação **migrate** para mover uma instância de um servidor para outro. A instância é desligada e reconstruída em outro servidor. Esta operação faz uso da **resize**, mas mantém o mesmo *flavor* utilizado na criação da instância. **Live migrate** também é usada para migrar instâncias, mas com a diferença de que a instância não será desligada. Geralmente é usada em casos nos quais os servidores precisam de manutenção; e
- **Trigger crash dump:** utilizada geralmente por administradores ou pelo dono da MV. A **trigger crash dump** cria um arquivo *dump* da imagem de memória dentro da MV.

Por fim, como são diversas as ações que podem ser feitas em instâncias de MVs, é necessário que exista algum esquema para identificar quais operações foram ou estão sendo executadas em determinado momento. Além disso, algumas ações só podem ser executadas caso a instância se encontre em uma determinada situação, *e.g.*, é impossível executar a ação **start** quando a MV já está ativa ou alguma outra ação/operação está sendo feita no mesmo momento. Portanto, o OpenStack trabalha com um esquema de variáveis que indica a situação atual da instância (estado). Dessa forma, além de identificar o estado atual da máquina, também é possível saber quais ações estão sendo executadas em um dado momento.

3.2 ESTADOS DAS MVS

No OpenStack, o estado de uma instância de MV indica as suas condições atuais. Neste sentido, o usuário tem acesso, via API do Nova Compute, ao ”Server Status”. O ”Server Status” é calculado de acordo com duas variáveis que são visíveis apenas por administradores: ”VM_STATE” e ”TASK_STATE”. O estado da tarefa,

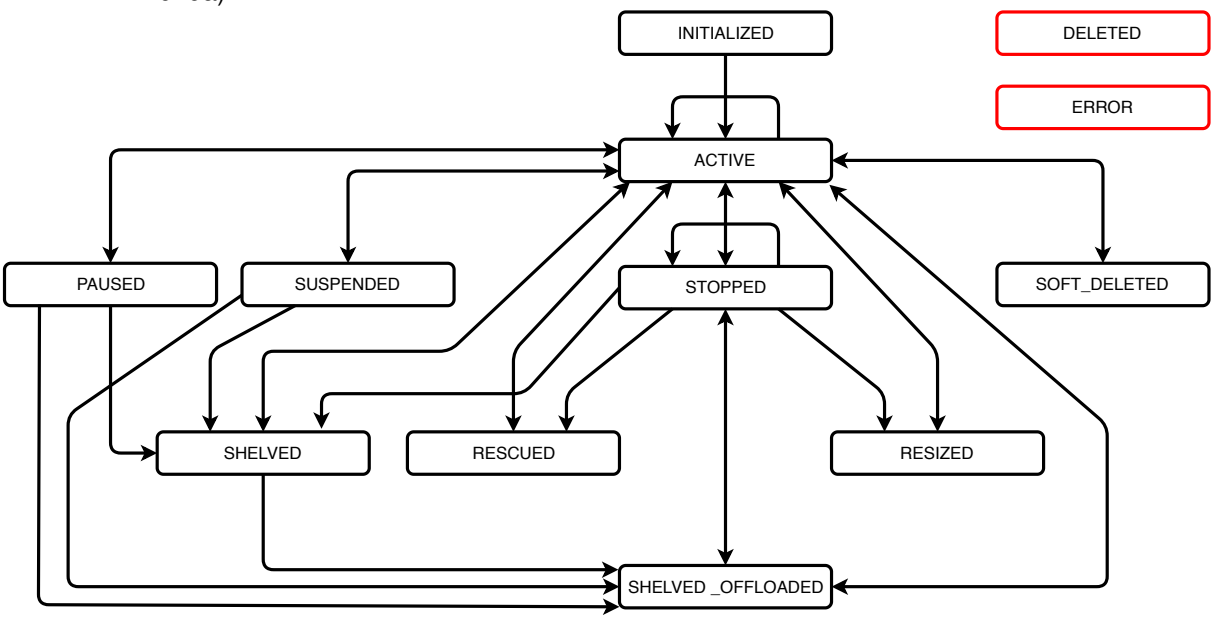
”TASK_STATE”, sempre indica uma ação em execução no momento (transição de estado), enquanto o ”VM_STATE” indica uma ação que já foi concluída (estado estável atual da MV). Quando os combinados, ”TASK_STATE” e ”VM_STATE” podem representar melhor o estado da instância. O ”Server Status” pode assumir os seguintes valores:

- **ACTIVE**: instância ativa;
- **BUILD**: instância ainda em processo de construção;
- **DELETED**: instância deletada;
- **SOFT_DELETED**: instância foi excluída, mas permanecerá na nuvem por algum tempo (configurável);
- **ERROR**: a instância apresenta erros;
- **HARD_REBOOT**: reinicialização forçada da instância (equivalente a um corte de energia) ;
- **MIGRATING**: instância em processo de migração;
- **PASSWORD**: está sendo feito um *reset* na senha da instância;
- **PAUSED**: instância em pausa;
- **REBOOT**: instância sendo reinicializada (*soft reboot*);
- **REBUILD**: instância sendo reconstruída;
- **RESCUE**: instância em modo de recuperação;
- **RESIZE**: executando processo de redimensionamento;
- **VERIFY_RESIZE**: verificando a operabilidade do sistema após um redimensionamento;
- **REVERT_RESIZE**: redimensionamento falhou e está sendo revertido;
- **SHELVED**: instância engavetada;
- **SHELVED_OFFLOADED**: instância engavetada e removida dos hipervisores. Necessário executar *unshelve* para ser usada novamente;
- **SHUTOFF**: instância desligada;
- **SUSPENDED**: instância suspensa; e

- **UNKNOWN**: o estado da instância é desconhecido.

Além de ”VM_STATE” e ”TASK_STATE”, também há o ”POWER_STATE”, que indica o estado atual da MV no hipervisor. O ”POWER_STATE” deve ser alterado periodicamente no banco de dados para refletir o valor no hipervisor. Também é necessário alterá-lo ao finalizar a execução de tarefas na instância. Embora não exista uma relação direta entre ”POWER_STATE” e ”VM_STATE”, em alguns casos, quando não há nenhuma tarefa em execução na MV, ambos os valores devem ser condizentes. Caso não exista coerência entre eles, significa que algum erro ocorreu e uma análise é necessária. Um exemplo no qual ”POWER_STATE” e ”VM_STATE” não são condizentes seria ”POWER_STATE=SHUTOFF”, que indica ”desligar” no hipervisor, e ”VM_STATE=ACTIVE”, indicando que a instância está ativa em execução, quando na realidade deveria ter valor STOPPED, indicando que a instância não está em execução. A Figura 13 mostra as possíveis transições de estado de uma instância de MV no OpenStack.

Figura 13 – Transições de estados para o gerenciamento de uma MV no OpenStack (OPENSTACK, 2019a)



Adaptado de: (OPENSTACK, 2018d).

Os estados apresentados na Figura 13 são os possíveis valores de ”VM_STATE”, pois, indicam o estado estável e atual da instância. Só é permitida a transição entre estados conforme mostra a Figura 13, *e.g.*, não é possível transitar diretamente do estado STOPPED ao PAUSED. A transição entre cada estado, representada pela seta, é resultado da execução de alguma das operações com instâncias de MVs. No momento em que esta operação está sendo executada, o TASK_STATE é alterado, indicando qual ação está sendo feita.

Por fim, em termos de comunicação e volume de tráfego gerado na rede, as transições de estado mais custosas são aquelas que requisitam ou persistem dados relacionados às instâncias de MVs. Já que na maioria das implantações do OpenStack, os serviços de armazenamento estão em servidores separados dos serviços de computação, tornando necessário o envio de dados via rede. Além disso, muitos dos processos de transição de estados envolvem diversos serviços do OpenStack. Portanto, também existe a necessidade de trocas de mensagens entre serviços (chamadas API ou até mesmo RPC).

3.3 ORGANIZAÇÃO DE TRÁFEGO NA NUVEM

O OpenStack tem requisitos de redes complexos que devem ser levados em consideração para uma implantação com alta escalabilidade, desempenho e segurança. Um dos principais motivos para esta complexidade é o alto nível de interação entre diferentes componentes do sistema. Além disso, o fluxo de dados no OpenStack também possui sua complexidade (OPENSTACK, 2018b). Na nuvem, todo dado que trafega entre as instâncias pela rede (tráfego interno) é identificado como tráfego leste-oeste. Já o fluxo de tráfego gerado com redes externas (*e.g.*, clientes ou serviços remotos) é chamado de tráfego norte-sul (OPENSTACK, 2018c).

As redes da nuvem podem ser divididas em zonas lógicas. A documentação recomenda uma divisão com no mínimo quatro zonas lógicas (OPENSTACK, 2018c):

1. **Rede Física**: conecta os servidores de armazenamento, computação e controle, chamada de *Underlay*;
2. ***Overlay***: construída em cima da rede física, que geralmente faz uma conectividade de *Layer 3* com os componentes da nuvem, como é o caso do *Neutron overlay*, que utiliza SDN;
3. ***Edge***: zona na qual há uma transição de tráfego do *Overlay* da nuvem ao ambiente de rede tradicional (rede física, *Layer 2*); e
4. ***External***: definida por todos os componentes e configurações necessárias para acesso aos recursos da nuvem. Além disso, todos os componentes fora dos *gateways* de borda da nuvem são considerados parte da rede externa (OPENSTACK, 2018c).

Para finalizar, a Figura 12 (vista no Capítulo 2) traz uma arquitetura de rede física (básica) para implantação do OpenStack. É possível identificar as quatro zonas lógicas recomendadas: *Underlay*, representada pela rede administrativa; *Overlay*, serviço SDN, rede de convidados; *Edge*, representado pelo próprio servidor do serviço

SDN; e *External*, formada pela rede externa e API. Neste contexto, é possível notar que, a zona de *Underlay* é a principal envolvida na execução das operações com MVs. Inclusive, é por meio da rede administrativa que são enviadas as imagens e demais dados necessários para certas operações, *e.g.*, na operação ***shelve***, é necessário armazenar o *snapshot* da instância.

3.4 DEFINIÇÃO DO PROBLEMA

A infraestrutura dos provedores de nuvem é parte essencial no fornecimento e boa qualidade de seu serviço. É necessário garantir um bom funcionamento, alto desempenho e escalabilidade da nuvem. Neste sentido, a infraestrutura da nuvem, como um todo, torna-se um alvo de estudos. Do ponto de vista de desempenho, há interesse em identificar e analisar operações que estão sendo executadas no provedor da nuvem, mais especificamente em pontos vitais para o fornecimento do serviço, como a rede administrativa da nuvem. Embora existam diversas pesquisas voltadas para a análise das redes da nuvem do ponto de vista dos usuários (Aishwarya. K; Sankar, 2015; Chen; Zhao, 2012; Shete; Dongre, 2017; Chaudhary et al., 2018; ALENEZI; ALMUSTAFA; MEERJA, 2019), a avaliação de operações internas e comportamento do provedor ainda é escassa.

A literatura especializada aponta que questões relacionadas ao desempenho de uma nuvem dependem da identificação de comportamentos e operações que ocorrem durante o seu uso (Bruneo, 2014). Além disso, é perceptível a necessidade de identificar operações, muitas das quais ocorrem no nível da camada de rede e demandam análise e compreensão do que está trafegando e sua finalidade. Dessa forma, a caracterização de tráfego auxilia este entendimento, por meio do emprego de técnicas e métodos que possibilitam a coleta e a identificação do tráfego de forma sistematizada.

Portanto, o problema identificado trata-se da falta de informações sobre as operações que ocorrem no nível da camada de rede dos provedores de nuvem. Mais especificamente, o problema em questão é sobre a identificação do tráfego interno da nuvem. Acredita-se que questões relacionadas à qualidade de serviço podem estar intimamente ligadas à organização de redes, bem como ligadas ao tráfego interno (administrativo) da nuvem. Neste sentido, o objetivo geral proposto é uma análise e caracterização do tráfego administrativo em nuvens OpenStack.

É por meio da rede administrativa do OpenStack que são feitas muitas (se não todas) operações internas da nuvem. Até mesmo as interações do usuário com operações em instâncias de MVs (*e.g.*, criação, parada, *backup*) demandam certo uso desta rede. Então, a ocorrência de gargalos e colapsos na rede administrativa pode

afetar significativamente a qualidade do serviço, podendo até mesmo gerar a sua indisponibilidade. Neste contexto, devido ao seu impacto no desempenho da nuvem, é necessário um controle adequado de tráfego e dimensionamento da rede administrativa do OpenStack.

3.5 DEFINIÇÃO DE REQUISITOS

A fim de propor uma solução ao problema da identificação de operações e tráfego na rede administrativa do OpenStack, são levantas as seguintes premissas:

- Adotar uma arquitetura base com dois servidores para realização dos experimentos de análise e caracterização do tráfego. Um dos servidores é encarregado das funções de controlador, enquanto o outro das funções de computação;
- Montar um cenário pertinente à coleta de tráfego na nuvem. Executar operações em instâncias de MVs para que seja gerado tráfego na rede de controle da nuvem;
- Ter acesso a uma nuvem OpenStack em nível de acesso à infraestrutura (SO e dispositivos de rede) cuja configuração de rede atenda às necessidades para coleta e análise do tráfego, que são:
  - Domínios de rede criados de acordo com as recomendações do OpenStack[1]; e
  - Tráfego devidamente separado por interfaces de rede (físicas ou virtuais).

Uma vez que as premissas são atendidas, é possível elencar requisitos funcionais e não funcionais para a realização de experimentos e solução do problema:

- **RF1**: Coletar o tráfego na rede administrativa da nuvem OpenStack;
- **RF2**: Separar o tráfego de acordo com a mudança de estado que está sendo executada na MV;
- **RF3**: Analisar o tráfego coletado de modo a identificar a qual serviço do OpenStack este se refere (*e.g.*, nova, glance);
- **RF4**: Armazenar o tráfego caracterizado em um banco de dados;
- **RF5**: Identificar o tempo no qual cada pacote foi coletado;

[1] <docs.openstack.org/security-guide/networking/architecture.html>

- **RNF1**: O tempo de coleta de cada pacote deve ser mostrado em segundos. Além disso, o tempo deve ser medido em relação à execução do experimento. Tempo 0 indica início do experimento (coleta de tráfego); e
- **RNF2**: As técnicas e ferramentas adotadas para a medição de tráfego na nuvem não devem afetar o desempenho da mesma.

3.6 TRABALHOS RELACIONADOS

Para conduzir a análise e caracterização de tráfego na rede administrativa do OpenStack, foram identificados trabalhos com escopo similar. Neste sentido, com a Tabela 2 lista os trabalhos relacionados que fora identificados e os compara de acordo com os requisitos definidos.

Tabela 2 – Trabalhos relacionados atendem ou não os requisitos.

	(WANG; NG, 2010)	(VENZANO; MICHIARDI, 2013)	(MCGRATH; RAYCROFT; BRENNER, 2015)	(SANKARI; VARALAKSHMI; DIVYA, 2015)	(SHARMA et al., 2015)	(SCIAMMARELLA et al., 2016)	(GUSTAMAS; SHIDIK, 2017)	(FLITTNER; BAUER, 2017)	(He et al., 2019)
RF1	Não atende	Não atende	Não atende	Atende parcialmente. O documento analisa tráfego de data centers SDN	Atende	Atende	Atende	Não atende	Atende
RF2	Não atende	Não atende	Não atende	Não atende	Não atende	Atende parcialmente	Não atende	Não atende	Atende parcialmente
RF3	Não atende	Não atende	Não atende	Não atende	Atende	Não atende	Não atende	Não atende	Não atende
RF4	Não informado pelo autor	Não informado pelo autor	Não atende	Não informado pelo autor	Não informado pelo autor	Não informado pelo autor	Não atende	Não informado pelo autor	Não informado pelo autor
RF5	Não informado pelo autor	Não informado pelo autor	Não atende	Atende	Não informado pelo autor	Atende	Não atende	Não informado pelo autor	Atende

Muitos dos trabalhos relacionados selecionados fazem estudo da infraestrutura de redes de nuvens computacionais. Os trabalhos (SCIAMMARELLA et al., 2016; SHARMA et al., 2015; He et al., 2019), apresentados na Tabela 2 são os que mais assemelham-se ao problema aqui apresentado, no sentido de tratarem do domínio de controle do OpenStack. Diferente do trabalho de (SCIAMMARELLA et al., 2016), que é, de todos os trabalhos, o que mais apresenta maior semelhança com o problema aqui identificado, tem-se maior enfoque nas operações que são feitas internamente pelo provedor da nuvem. Enquanto (SCIAMMARELLA et al., 2016) preocupa-se especificamente com o tráfego de controle gerado pela criação/destruição de múltiplas instâncias em nuvens colaborativas geo-distribuídas, o presente trabalho busca analisar o tráfego gerado por operações, de modo geral, em MVs em nuvens OpenStack. Além disso, o trabalho de (He et al., 2019) também preocupa-se com o tráfego gerado no domínio de controle por algumas operações em MVs, contudo, são avaliadas apenas operações de inicialização, migração com a MV ativa e finalização. No trabalho de (He et al., 2019) também destaca-se o uso de redes SDN com uma implantação do OpenStack em um cenário de produção, usando a versão Newton (*end of life* desde 2017).

Por fim, também foram selecionados trabalhos que fazem caracterização/classificação de tráfego em *data centers* de nuvens que não necessariamente são implan-

tações OpenStack. Além disso, diferente destes trabalhos, a saber (SANKARI; VARALAKSHMI; DIVYA, 2015; FLITTNER; BAUER, 2017; GUSTAMAS; SHIDIK, 2017), que tem maior enfoque em redes SDN, o presente trabalho busca a análise e caracterização do tráfego de uma rede física.

3.7 CONSIDERAÇÕES PARCIAIS

Do ponto de vista interno da nuvem, o processo de criação e uso das MVs é complexo e envolve diversos componentes. Além de dar suporte à criação e r emoção das instâncias de MVs, também é necessário dar suporte a operações, *e.g.*, suspensão da MV e criação de *backups*. Neste sentido, o OpenStack faz uso de um esquema de estados de MVs, permitindo que o sistema tenha conhecimento sobre a situação das instâncias em um dado momento. Com as informações de estados das MVs, também é possível reportar ao usuário sobre o comportamento e possíveis operações da instância.

O ciclo de vida das MVs é parte fundamental no entendimento do funcionamento do OpenStack. É possível notar a frequente comunicação entre serviços para garantir todo o gerenciamento das instâncias. Além disso, nota-se que certas operações produzem mais tráfego administrativo devido ao envio de mensagens que contém imagens e *snapshots*, por exemplo. Neste sentido, ainda do ponto de vista da infraestrutura da nuvem, existe uma necessidade de identificar operações, muitas das quais ocorrem em nível de camada de rede. Portanto, propõe-se uma análise e caracterização de tráfego da rede administrativa do OpenStack.

Adicionalmente, analisando os trabalhos relacionados, não há nenhum documento cujo foco de estudos se encaixa exatamente com o problema da identificação do tráfego de controle do OpenStack. Contudo, o trabalho de (SCIAMMARELLA et al., 2016) é o que mais se aproxima, no sentido de tratar do domínio de controle do OpenStack em nuvens geo-distribuídas.

4 PROPOSTA & IMPLEMENTAÇÃO

Este capítulo introduz a proposta e a implementação da solução do problema descrito na Seção 3.4. São abordados os passos e a organização da solução, além de discutir algumas hipóteses sobre a caracterização de tráfego e possíveis cenários de testes. Portanto, neste capítulo preocupa-se em explicar o funcionamento da solução proposta e como a colocar em prática.

4.1 PROPOSTA DA SOLUÇÃO

Conforme a Seção 3.4, por tratar-se de uma parte vital ao fornecimento do serviço de computação em nuvem, a rede administrativa do OpenStack torna-se alvo de estudos voltados principalmente a questões de desempenho. Neste sentido, propõe-se uma caracterização do tráfego desta rede. A partir desta caracterização é possível, por exemplo, identificar operações que estão executando no momento, bem como seu impacto na rede. Neste sentido, para que o objetivo final de caracterização do tráfego seja atendido, são necessários alguns passos:

- Entender sobre o funcionamento do OpenStack. Como é feita a organização de redes, como os serviços comunicam-se e como é feito o gerenciamento das instâncias de MVs (Capítulo 2);
- Identificar uma arquitetura base de implantação do OpenStack (Subseção 4.1.1);
- Coletar dados sobre o tráfego na rede administrativa do OpenStack (Subseção 4.1.3);
- Analisar os dados (Subseção 4.1.3); e
- Caracterizar o tráfego coletado (Subseção 4.1.3).

Conforme o Capítulo 2, o OpenStack é um sistema operacional para nuvens computacionais e seu funcionamento é um tanto complexo. Os serviços do OpenStack estão disponíveis em módulos e interagem entre si para que os usuários tenham acesso a recursos computacionais em ambientes virtualizados (seja com MVs ou contêineres). Neste sentido, até mesmo quando visto de maneira resumida, o diagrama de sequência da criação de uma instância (Figuras 8 e 9), por exemplo, já fornece boas noções em relação à comunicação interna do OpenStack.

Tendo em vista este cenário distribuído, no qual uma instalação do OpenStack sugere a utilização de diversos servidores e que exercem funções diferentes (Sub-

seções 2.4.2 e 2.4.3), também é necessário conhecimento sobre sua infraestrutura e organização de rede. Neste sentido, nota-se a importância das divisões de redes da nuvem. Tanto por questões de segurança quanto por questões de desempenho, as implantações do OpenStack exigem separação entre os domínios **Público**, **Convidado** e **Controle**. Dessa forma, para que uma coleta de tráfego seja conduzida em um ambiente como este, é necessário planejamento de uma arquitetura conforme as especificações para implantações do OpenStack.

Uma vez que a arquitetura e organização de redes é definida, pode-se proceder com a etapa de medição/coleta de tráfego administrativo. Nesta etapa, é necessário saber o que coletar da rede. Pode-se conduzir uma medição passiva ou ativa, no sentido de que a coleta de tráfego pode ou não interferir na rede. Também devem ser levados em consideração o nível de agregação de pacotes e as informações relevantes de cada pacote (cabeçalho e/ou carga do pacote).

Já em relação às fases de análise e caracterização do tráfego coletado, podem ser utilizadas uma ou mais técnicas (Subseção 2.5.1). A escolha da técnica a ser utilizada depende das particularidades da rede em questão. É necessário avaliar o nível de conhecimento sobre a rede, serviços envolvidos, uso de criptografia e privacidade de dados. Em redes nas quais há certa previsibilidade do tráfego, como é o caso da rede administrativa do OpenStack, abordagens baseadas em portas acabam sendo mais utilizadas, principalmente pela sua simplicidade e eficácia. Além disso, devido ao serviço de mensageria do OpenStack (feito pelo RabbitMQ), o tráfego também pode ser agregado em fluxos (*flow*).

4.1.1 Ambiente para caracterização

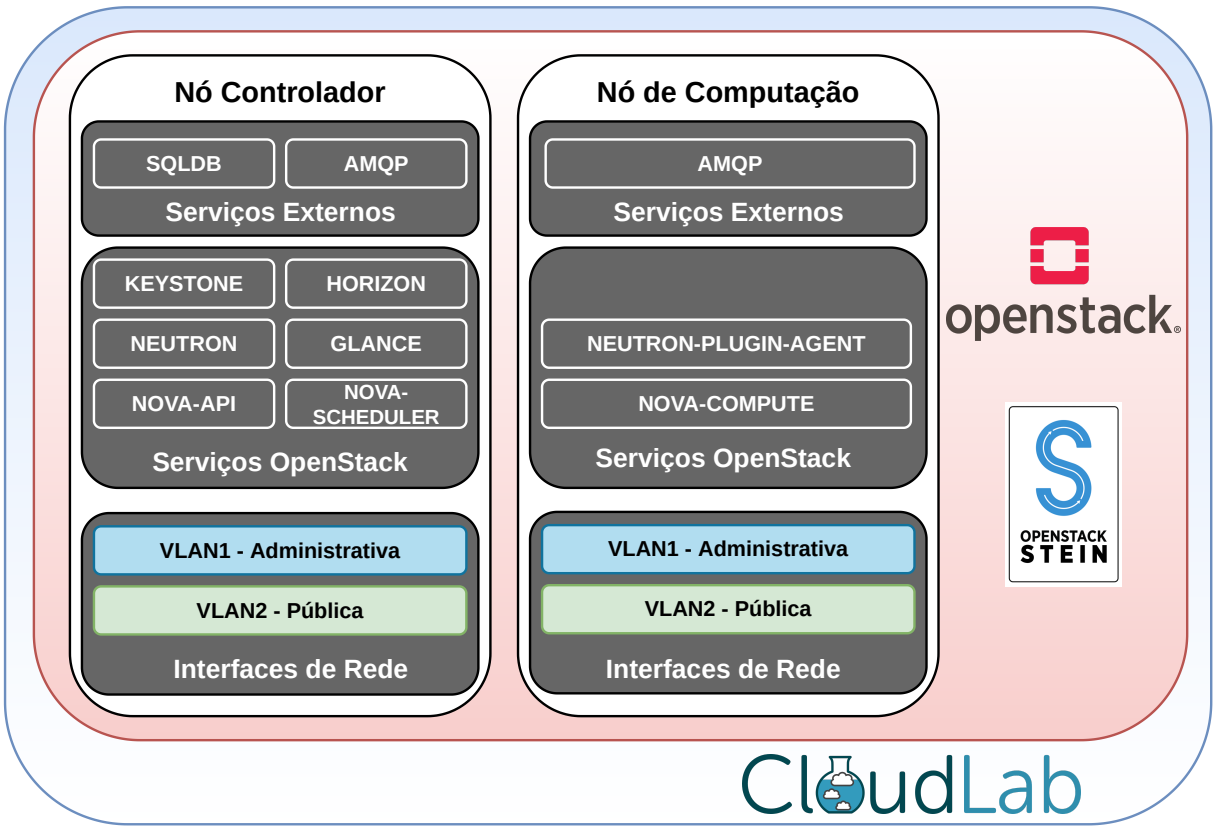
A fim de realizar a experimentação da solução proposta, é necessário que um ambiente padrão seja definido. Dessa forma, propõe-se uma implantação de nuvem OpenStack com arquitetura mínima de dois nós (Figura 11). A escolha desta arquitetura justifica-se pelo fato de que se trata de uma configuração básica para que ocorram comunicações e troca de dados via rede entre dois servidores do OpenStack. A Figura 14 mostra a arquitetura proposta com uma implantação do OpenStack Stein (versão estável do OpenStack no momento da realização dos experimentos) no CloudLab[1], plataforma na qual é possível ter acesso às infraestruturas para pesquisas voltadas para computação em nuvem.

Em relação aos componentes do OpenStack usados neste ambiente de experimentos, tratam-se dos serviços básicos e de apoio para funcionamento da nuvem, devidamente explicados no Capítulo 2 e ilustrados na Figura 14. Quanto ao hardware dos servidores, a infraestrutura da nuvem é provida pelo CloudLab (CLOUDLAB,

[1] http://cloudlab.us

2019), que de acordo com o perfil dos experimentos, fornece dois servidores, um para computação e outro para controle da nuvem. O CloudLab permite que os servidores sejam alocados em um dos seus três *clusters* principais, Utah, Clemsom ou Wisconsin. Em média[2], cada um dos servidores conta com 256GB de memória e dois processadores com 2.4Ghz cada, que totalizam 16 núcleos. Além disso, é possível escolher entre rede 1Gb/s e 10Gb/s.

Figura 14 – Configuração utilizada durante os experimentos, redes Pública e Administrativa são isoladas em VLANs.



Fonte: O próprio autor.

A arquitetura apresentada na Figura 14, além de fornecer as configurações de interfaces necessárias (com VLAN), viabilizando a separação entre os tráfegos, também facilita a reprodução dos experimentos. Em adição, é válido lembrar que, mesmo em configurações de redes diferentes, as operações realizadas pelo OpenStack são as mesmas. Contudo, a caracterização de tráfego é muito dependente do ambiente no qual é aplicada, ou seja, a forma como a coleta e caracterização são conduzidas pode mudar significativamente de acordo com as configurações da rede.

[2] Especificação de *hardware* completa em <www.cloudlab.us/hardware.php>

4.1.2 Mecanismo para automatização de experimentos e coleta de dados

Todas as operações que podem ser feitas via *dashboard* do OpenStack também podem ser feitas via CLI. Neste sentido, é possível criar *scripts* que servem para a automatização dos experimentos, principalmente para executar operações em MVs, *e.g.*, criar e parar instâncias. Adicionalmente, também é viável a criação de *scripts* para iniciar e finalizar a captura de pacotes em uma determinada interface de rede.

A fim de combinar tanto os *scripts* de automatização dos experimentos quanto os de captura de pacote, implementou-se uma ferramenta em Python, que faz uso das APIs do OpenStack *e.g.*, keystoneclient e novaclient. Basicamente, a ferramenta é utilizada para fazer a captura de pacotes, com auxílio de TCPdump, e executar as operações em instâncias de MVs. Além disso, a ferramenta também é empregada para repetir o experimento *n* vezes, tipicamente 10 vezes.

4.1.3 Processo de medição, análise e caracterização de tráfego

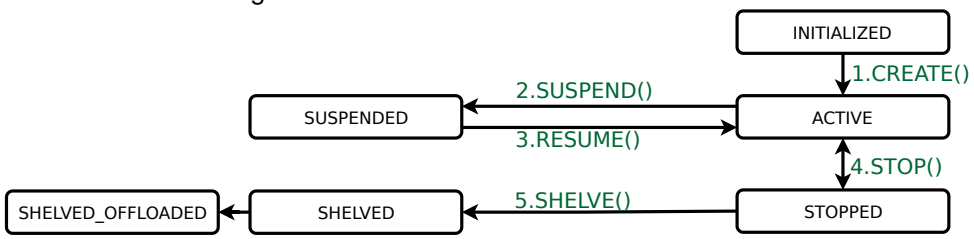
O processo de medição/coleta de tráfego pode ser feito de maneira passiva ou ativa. Nos experimentos aqui realizados, adota-se o método de medição passiva, visto que não há intrusão na rede, apenas coleta de tráfego. Contudo, como o tráfego presente na rede é produto das operações executadas pelo algoritmo, o método de medição também pode ser interpretado como híbrido. Além disso, é adotado um nível de granularidade no qual todos os cabeçalhos dos pacotes são coletados.

Sobre a análise e caracterização do tráfego coletado, todos os cabeçalhos são interpretados com o uso do método baseado em portas. Como tem-se conhecimento prévio sobre a rede administrativa do OpenStack (*e.g.*, possíveis serviços operantes) e esta também não faz uso de protocolos com criptografia, o método baseado em portas tem uma aplicabilidade adequada neste cenário. Além disso, o OpenStack faz uso de chamadas API, separando cada serviço com sua respectiva porta, o que torna o método baseado em portas ainda mais propício ao cenário. Por fim, após classificado/caracterizado, o tráfego é salvo em um banco de dados. Dessa forma, é possível separar o tráfego por serviço e operação do OpenStack.

4.1.4 Plano de Testes

Para que o tráfego administrativo seja mensurado, é necessário que existam instâncias de MVs criadas e em execução. Neste sentido, definiu-se um conjunto de operações em MVs que é chamado de "ciclo de vida induzido", uma vez que as instâncias passam por estados que simulam seu uso padrão. A Figura 15 mostra as operações, bem como os estados assumidos pelas instâncias durante a execução deste ciclo.

Figura 15 – Ciclo de vida induzido das MVs.



Fonte: o próprio autor.

O tráfego presente na rede administrativa durante a execução dos experimentos é oriundo das operações do ciclo de vida induzido. De acordo com a Figura 15, com a operação CREATE, as instâncias partem do estado **INITIALIZED** ao estado **ACTIVE**. Em seguida, a operação SUSPEND faz com que a instância saia do seu estado **ACTIVE** e passe a ter um estado **SUSPENDED**. A operação RESUME retorna o estado **ACTIVE** da instância que está **SUSPENDED**. Então, a instância é parada com o STOP, atingindo um estado **STOPPED** e depois engavetada, com SHELVE, chegando em seu estado final **SHELVED** e **SHELVED_OFFLOADED**, que é atingido após a instância ser removida dos hipervisores.

Algumas hipóteses são que, de acordo com o tipo (extensão) da imagem usada para a criação da instância, o volume de tráfego produzido por algumas operações pode variar. Possivelmente, extensões mais compactas, *e.g.*, *qcow2*, podem resultar na criação de um volume de tráfego menor, principalmente na etapa de criação da instância. Adicionalmente, o *flavor* da instância também pode influenciar no volume de tráfego gerado pelas operações executadas. Já que o *flavor* define informações como a quantidade de memória RAM disponível à instância, algumas operações nas quais o estado da memória é salvo, podem apresentar maiores alterações no volume de tráfego, dependendo dos níveis de uso da memória da instância. Além disso, ao mensurar chamadas API, a implementação das operações envolvidas no ciclo de vida pode influenciar no número de chamadas observadas para cada serviço do OpenStack, no sentido de que há diversas formas de executar operações em MVs. Em Python é possível utilizar a interface Connection.Compute ou a *novaclient* API para gerenciar as MVs, por exemplo. Neste sentido, são executados os seguintes experimentos nomeados:

1. **Análise de tráfego por imagens do SO**: Identificar o volume de tráfego e chamadas API por serviço do OpenStack de acordo com o ciclo de vida induzido das instâncias de MVs. Identificar variações no tráfego administrativo em instâncias que usam diferentes imagens de SOs, partindo de um estado *clean slate*, principalmente para evitar técnicas como o *caching* das imagens das instâncias;
2. **Análise de tráfego baseado no tipo de *Flavors***: Identificar variações no volume

de tráfego administrativo em instâncias que usam o mesmo SO, mas que são criadas com *flavor* diferentes; e

3. **Predição de Tráfego**: Aplicar regressão linear para predição do volume de tráfego administrativo para criação da instância de MV de acordo com o tamanho da imagem do SO.

4.2 IMPLEMENTAÇÃO

Sobre o processo de implementação da solução, com o auxílio do CloudLab, que fornece o *testbed* para os experimentos, cria-se uma nuvem OpenStack com a versão Stein. A versão Stein foi adotada, pois, representa a versão mais recente e estável no momento da execução dos experimentos. Contudo, a experimentação pode ocorrer independente da versão do OpenStack. A nuvem criada possui dois servidores, um para computação e outro para controle da nuvem, tal como mostra a Figura 14. É importante que, ao criar a nuvem, seja especificada a divisão existente entre as redes, e que essa divisão deve ser feita com VLAN. Fisicamente, seria possível fazer essa divisão de redes com duas interfaces diferentes (por servidor), mas o ambiente de testes do CloudLab fornece apenas uma interface. Então, a separação de tráfego é feita via VLAN.

Neste contexto, o acesso ao controlador é feito via *Secure Shell* (SSH) e todo o experimento é executado neste servidor. Para iniciar a experimentação são necessários alguns preparativos:

- Identificar, dentre as VLANs existentes, qual delas é referente à rede administrativa. Pode ser feito ao observar o tráfego nas VLANs. Já que são criadas apenas duas, uma para rede pública e outra para rede administrativa, ao criar uma instância de MV, por exemplo, o tráfego correspondente ao tamanho da imagem da instância passa pela VLAN administrativa; e
- Preparar o ambiente, no sentido de obter a ferramenta criada para monitoramento da rede administrativa do OpenStack, OpenStack Network Monitor (ONM)[3], bem como todas as demais bibliotecas necessárias ao ambiente. As implementações são feitas, basicamente, em Python 3.6. Logo, alguns pacotes precisam ser obtidos/atualizados para preparar o ambiente. Além disso, também é necessário fazer *download* das imagens que serão utilizadas nas instâncias de MVs.

Ao executar o ONM, as MVs são criadas e submetidas às operações do ciclo de vida induzido, uma a uma, como mostra a Figura 15. Enquanto as operações em

[3] <github.com/Adnei/openstack_monitor>

MV estão sendo executadas, o tráfego na VLAN administrativa é coletado. Ao completar o experimento, o principal produto resultante é um banco de dados, contendo informações relevantes sobre o tráfego coletado, operações, número de execuções e imagens utilizadas. O processo de implementação da solução pode ser resumido com os procedimentos (Figura 16):

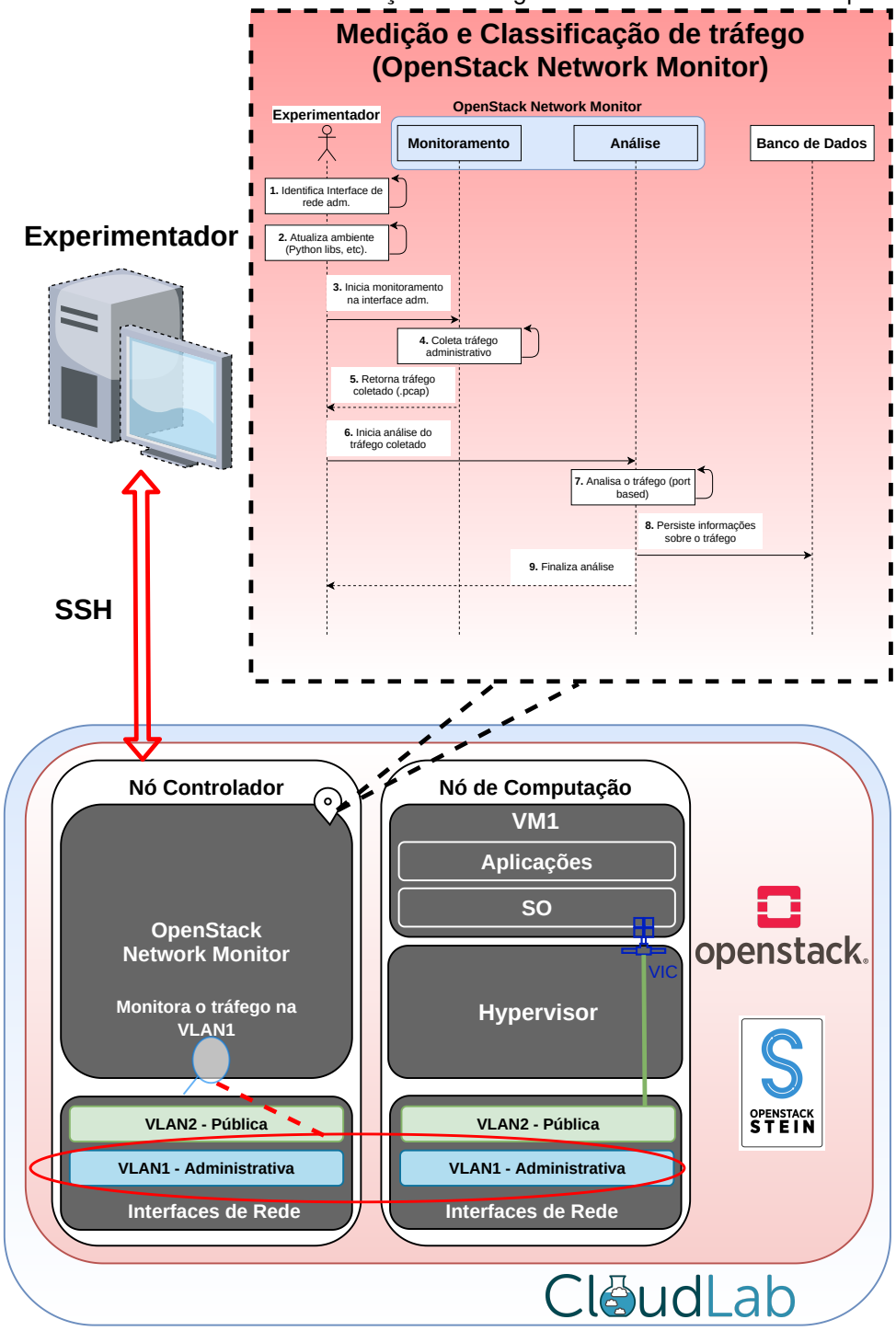
1. Acessar o controlador da nuvem e identificar a interface de rede referente à rede administrativa;
2. Iniciar o monitoramento da interface de rede administrativa;
3. Executar tarefas em instâncias de MVs. Aqui também considera-se a própria criação da MV;
4. Finalizar o monitoramento e classificar/caracterizar o tráfego coletado; e
5. Salvar o tráfego classificado em um banco de dados para posterior análise.

De acordo com a Figura 16, identifica-se que todo o processo de experimentação é executado a partir do nó controlador da nuvem. A experimentação é iniciada pelo usuário experimentador, no Passo 1, com a identificação da interface de rede referente à rede administrativa do OpenStack. É importante notar que a identificação das redes é variável de acordo com a infraestrutura/ambiente de caracterização, e geralmente requer maior atenção, visto que a coleta de tráfego é feita de acordo com as redes identificadas. No Passo 2, o experimentador atualiza o ambiente, baixando/atualizando bibliotecas e também APIs do OpenStack para Python, assim a ferramenta de monitoramento pode ser executada. Na sequência (Passo 3), o monitoramento é iniciado e a ferramenta é responsável por coletar os pacotes que passam pela interface de rede administrativa (Passo 4).

A coleta de tráfego administrativo (Passo 4) sempre ocorre em conjunto com a execução de operação em MV, neste caso, as operações executadas atendem ao ciclo de vida induzido apresentado na Figura 15. Sempre que o ciclo de vida induzido da MV é executado por completo, contabiliza-se uma execução do experimento. Ou seja, cada execução consiste em cinco operações em MV (Figura 15). Além disso, cada execução do ciclo de vida inicia com um estado *clean slate*, cuja finalidade é evitar técnicas como o *caching* de imagens de SOs para MVs. Portanto, também é possível dividir as execuções em operações do ciclo, *e.g.*, criação da primeira execução, criação da segunda execução.

Ao término de uma execução de coleta de tráfego (Passo 4), tem-se os dados brutos em formato *.pcap* (Passo 5). Para cada execução é feita uma análise do tráfego

Figura 16 – Fluxo da coleta e caracterização do tráfego na rede administrativa do OpenStack.



Fonte: O próprio autor.

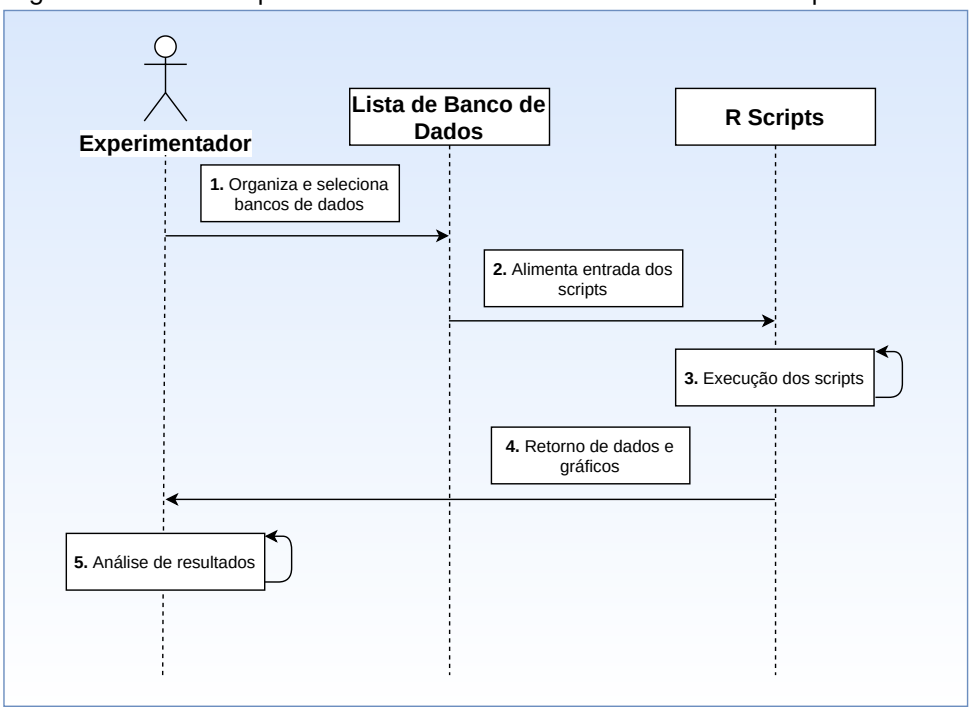
coletado (Passos 6 ao 9). A análise do tráfego coletado resulta em um banco de dados, que contém informações de cabeçalho dos pacotes que são relevantes ao problema, *e.g.*, *timestamp*, porta origem e destino e tamanho do pacote. Também são persistidas informações como identificadores de execução, operação, imagem da MV e serviço OpenStack relacionado ao pacote.

O Passo 6 refere-se ao início da fase de análise de tráfego, e o Passo 7 refere-

se à execução da análise pela ferramenta (classificação do tráfego de acordo com portas de origem e destino). Na etapa seguinte, Passo 8, cria-se um banco de dados contendo todo o tráfego coletado e devidamente classificado, e no Passo 9 o processo de monitoramento e classificação de tráfego é finalizado. O objetivo deste processo é o levantamento e análise do tráfego oriundo das operações em MVs. Assim, informações como o volume de tráfego por operação e chamadas API por serviço, podem ser utilizadas no gerenciamento proativo da rede,*e.g.*, criar uma rede isolada para tráfego das imagens das instâncias ou regular recursos de rede (*e.g.*, largura de banda). Também é possível identificar quais serviços do OpenStack são mais ativos durante as operações com instâncias, qual o volume de tráfego gerado, quais operações são responsáveis pela maior parte do tráfego, entre outras informações. Além disso, é possível também identificar um padrão de tráfego administrativo (*e.g.*, quanto de tráfego administrativo é gerado por determinada operação).

A etapa de análise dos dados levantados pela coleta de tráfego é melhor detalhada na Figura 17. Nesse processo de análise, ocorre a transformação do dado em informação. São utilizados *scripts* em linguagem R, que utilizam como entrada os bancos de dados oriundos da medição e caracterização de tráfego (Figura 16, Passos 1 ao 9) para interpretação dos dados. Essa interpretação dos dados permite, por exemplo, a identificação de uma possível relação entre o tamanho da imagem de SO utilizada na MV e o volume de tráfego gerado na rede administrativa.

Figura 17 – Fluxo do processamento e análise dos dados coletados pelo monitor.



Fonte: O próprio autor.

O processamento de dados (Figura 17), mesmo que seja feito com auxilio de

scripts, é um processo que exige muito do experimentador, no sentido de que cabe ao experimentador interpretar os dados, criar inferências e tirar suas próprias conclusões. É durante a interpretação de dados que são analisados gráficos e medidas de tendência central, por exemplo, que guiam o experimentador. Neste sentido, os Passos 1 ao 5, apresentados na Figura 17, mostram como o experimentador utiliza os bancos de dados, que contém as informações sobre o tráfego coletado, para criar e interpretar gráficos, tabelas e demais resultados.

A técnica de classificação de tráfego baseada em portas, usando o TCP, é muito útil no cenário da rede administrativa do OpenStack, na qual os serviços operam em portas específicas. A identificação por portas dos serviços operantes do OpenStack durante a medição de tráfego (Figura 16, Passo 4) é bastante eficaz. Contudo, o *middleware* de comunicação entre serviços RabbitMQ torna o contexto um pouco mais complexo do que uma simples aplicação do método baseado em portas. O RabbitMQ implementa o protocolo AMQP, que é responsável pelas filas de mensagens que são trocadas entre os diversos serviços do OpenStack (OPENSTACK, 2020a). As implementações do protocolo AMQP envolvem chamadas RPC, que inviabilizam a utilização de portas predefinidas para cada serviço do OpenStack.

Neste sentido, a implementação da solução de coleta e classificação de tráfego da rede administrativa do OpenStack utiliza uma forma de mapeamento das portas TCP utilizadas pelo RabbitMQ durante a etapa de coleta de tráfego na rede. Neste sentido, apesar da porta TCP ser a mesma, a associação com o processo do serviço OpenStack que interage com o serviço de mensageria AMQP é diferente. Logo, é possível identificar qual serviço OpenStack está usando o AMQP mapeando a requisição ao processo no SO. O mapeamento é feito via comando *lsof*, utilizando os parâmetros *-i :5672*. O que, de modo geral, indica uma lista de todos os processos utilizando a porta TCP/5672, que é a porta utilizada pelo RabbitMQ (Figura 18).

Com as informações dos processos, obtidas pelo *lsof*, é possível identificar os serviços associados às suas respectivas portas em execução no momento da coleta de tráfego. Dessa forma, cria-se um mapa de serviços e portas a partir do *lsof*, com a finalidade de caracterizar o tráfego gerado em rede pelas trocas de mensagens do RabbitMQ. Nota-se, pelo *snippet* da Figura 18, como as conexões TCP podem ser mapeadas pelo campo NAME. Na linha 6, por exemplo, o serviço Glance usa a porta TCP 56828 do nó controlador (apresentado como *ctl* no arquivo) para conectar-se à porta *amqp*, que é a 5672, usada pelo RabbitMQ. Neste sentido, criou-se uma ferramenta utilitária[4], que foi acoplada ao monitor de rede, para mapear os serviços e suas portas com conexão estabelecida com o RabbitMQ.

[4] <github.com/Adnei/service_identifier>

Figura 18 – Exemplo de saída padrão do comando *lsof* para a porta TCP 5672 em um dos experimentos.

```
1  COMMAND     PID       USER   FD   TYPE  DEVICE SIZE/OFF NODE NAME
2  ceilomete  6225 ceilometer    7u  IPv4  194228      0t0  TCP ctl:52264->ctl:amqp (ESTABLISHED)
3  /usr/bin/  6442    neutron    5u  IPv4  149342      0t0  TCP ctl:50828->ctl:amqp (ESTABLISHED)
4  /usr/bin/  6442    neutron    6u  IPv4  149343      0t0  TCP ctl:50830->ctl:amqp (ESTABLISHED)
5  /usr/bin/  6442    neutron    7u  IPv4  149344      0t0  TCP ctl:50832->ctl:amqp (ESTABLISHED)
6  glance-ap  7613     glance   10u  IPv4  287851      0t0  TCP ctl:56828->ctl:amqp (ESTABLISHED)
7  glance-ap  7613     glance   11u  IPv4  287460      0t0  TCP ctl:56872->ctl:amqp (ESTABLISHED)
8  glance-ap  7615     glance   10u  IPv4  302164      0t0  TCP ctl:54956->ctl:amqp (ESTABLISHED)
9  glance-ap  7615     glance   11u  IPv4  302165      0t0  TCP ctl:54958->ctl:amqp (ESTABLISHED)
10 glance-ap  7620     glance   10u  IPv4 3321341      0t0  TCP ctl:40258->ctl:amqp (ESTABLISHED)
11 glance-ap  7620     glance   11u  IPv4 3321342      0t0  TCP ctl:40260->ctl:amqp (ESTABLISHED)
12 cinder-sc  7647     cinder    6u  IPv4  208300      0t0  TCP ctl:59590->ctl:amqp (ESTABLISHED)
13 cinder-sc  7647     cinder    7u  IPv4  208301      0t0  TCP ctl:59592->ctl:amqp (ESTABLISHED)
14 cinder-sc  7647     cinder    8u  IPv4  208323      0t0  TCP ctl:59608->ctl:amqp (ESTABLISHED)
15 cinder-vo  7856     cinder    8u  IPv4  226481      0t0  TCP ctl:59606->ctl:amqp (ESTABLISHED)
16 cinder-vo  7856     cinder    9u  IPv4  226482      0t0  TCP ctl:59610->ctl:amqp (ESTABLISHED)
17 cinder-vo  7856     cinder   10u  IPv4  226483      0t0  TCP ctl:59612->ctl:amqp (ESTABLISHED)
```

Fonte: O próprio autor.

4.3 CONSIDERAÇÕES PARCIAIS

Neste capítulo é feita a proposta para a solução do problema de caracterização de tráfego no domínio de controle do OpenStack, mais especificamente, na sua rede administrativa. Adicionalmente, são discutidos o ambiente, processo de análise e caracterização e algumas hipóteses sobre os experimentos de caracterização de tráfego de acordo com o cenário. Também são apresentados o plano de testes e a implementação, que explica como os testes/experimentos são conduzidos na nuvem. Algumas limitações identificadas na implementação da solução são a escalabilidade e o tempo de execução do experimento.

5 RESULTADOS

Os resultados apresentados neste capítulo são obtidos seguindo de acordo com o plano de testes (Subseção 4.1.4). Separam-se os resultados por seção de acordo com o experimento realizado: A Seção 5.1 explica a coleta e análise de tráfego feita com diferentes imagens de SO para um mesmo *flavor*; a Seção 5.2 mostra os resultados com variações no *flavor* de uma única imagem; e a Seção 5.3 traz uma análise para aplicação de um modelo de regressão linear com os dados da Seção 5.1.

As principais métricas avaliadas nas experimentações são:

- **Volume de tráfego**. Esta métrica diz respeito ao volume total de tráfego (MB) mensurado na rede administrativa do OpenStack durante a execução do experimento. O volume de tráfego é tratado na sua totalidade (montante total da operação), bem como por serviço OpenStack (total por serviço) e por segundo (no decorrer da operação);
- **Número de chamadas API**. Interpreta-se a totalidade de chamadas API feitas durante uma operação e a totalidade por serviço OpenStack; e
- **Tempo de execução por operação**. Tempo decorrido, em segundos, desde o início até o fim da operação.

Todas as métricas são organizadas apresentando a média dentre todas as execuções e seu desvio padrão (*sd*).

5.1 ANÁLISE DO TRÁFEGO POR IMAGENS DO SO

A Análise do tráfego por imagens do SO consiste em reproduzir o ciclo de vida induzido das MVs, conforme proposto no plano de testes (Seção 4.1.4), de modo que seja feita uma variação na imagem do SO. Os experimentos com variação de imagens contam com a alternância entre dez imagens diferentes. Para cada imagem, o tráfego de rede gerado durante o experimento é capturado e analisado. A experimentação em cada instância de MV se dá com o ciclo de vida induzido (Figura 15):

1. **Criação**: É feita a criação de uma instância de MV. É importante ressaltar que parte-se de um estado *clean slate*, a fim de evitar técnicas como o *caching* da imagem das instâncias, que reduziria o volume de tráfego na rede. A instância é iniciada em um estado INITIALIZED e atinge o estado ACTIVE ao término da operação CREATE;

2. **Suspensão**: Executa-se a operação SUSPEND, então, a instância que encontra-se ativa é suspendida e seu estado é alterado para SUSPENDED;
3. **Retomada de execução**: Com a operação RESUME, a instância retorna ao seu estado ativo;
4. **Parada**: A instância é parada com a operação STOP; e
5. **Engavetada**: Com a operação SHELVE, a instância é engavetada, que indica o término da experimentação.

Neste contexto, durante a execução do ciclo de vida induzido das instâncias, o tráfego administrativo, bem como as chamadas de API são contabilizados. Adota-se o *flavor* ***m1.small*** (padrão do OpenStack) para todas as instâncias, que especifica 1 vCPU, 20 GB para armazenamento e 2 GB para memória principal. Com o objetivo de identificar as alterações no volume de tráfego medido, são escolhidas imagens de SOs de diferentes tamanhos. Neste sentido, são utilizadas as imagens de SOs GNU/Linux CirrOS, GNU/Linux Fedora Cloud, GNU/Linux Ubuntu, duas imagens com tamanhos diferentes do GNU/Linux CentOS, GNU/Linux Mint, GNU/Linux Debian e MS-Windows Server, que são detalhadas na Tabela 3.

Tabela 3 – Imagens de SOs utilizadas nos experimentos.

SO	Versão	Tamanho (MB)	Formato
GNU/Linux CirrOS	0.4.0	15	QCOW2
GNU/Linux Fedora Cloud	32-1.6	289	QCOW2
GNU/Linux Fedora Cloud	31-1.9	319	QCOW2
GNU/Linux Ubuntu Server	18.04 LTS (Bionic Beaver)	329	QCOW2
FreeBSD	12.0	454	QCOW2
GNU/Linux Ubuntu Server	20.04 LTS (Focal Fossa)	519	QCOW2
GNU/Linux Debian	10	550	QCOW2
GNU/Linux CentOS	7	898	QCOW2
GNU/Linux CentOS	7	1300	QCOW2
MS-Windows Server	2012 R2	6150	QCOW2.GZ

Fonte: O próprio autor.

São escolhidas imagens com formato QCOW2, pelo suporte do QEMU[1] e também para padronização, uma vez que, imagens de diferentes formatos poderiam influenciar na execução de determinadas operações, como no SHELVE, o que resulta em um volume de tráfego variável. A imagem do MS-Windows 2012 R2 utiliza um formato comprimido (*.gz*) devido seu tamanho.

Nos experimentos com variação de imagens, cada SO listado na Tabela 3, é executado a partir de uma instalação *clean slate* do OpenStack Stein. Cada teste com

[1] <https://www.qemu.org/>

SO foi realizado 30 vezes (Tabela 4). São selecionadas imagens com variedade de tamanho devido a relação existente com o volume de tráfego administrativo resultante da sua transferência pela rede.

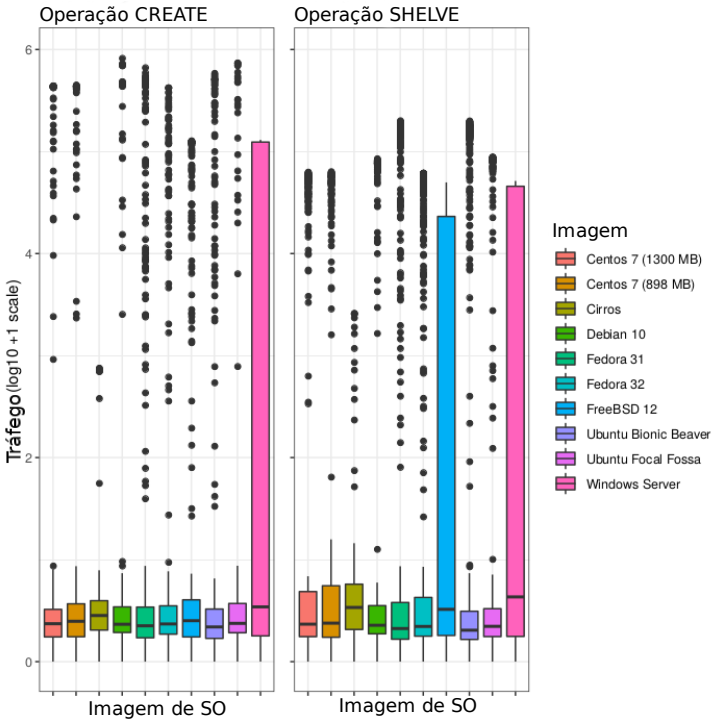
Tabela 4 – Resumo de dados obtidos nos experimentos com todas as imagens de SO apontadas na Tabela 3. Desvio padrão é representado por *sd*.

Imagem	Operação	Tráfego Total - MB (média ± *sd*)	Chamadas API (média ± *sd*)	Tempo de Exec. - segundos (média ± *sd*)
Ubuntu Bionic Beaver	CREATE	358,759 ± 1,988	67 ± 2,801	25,433 ± 1,455
	SUSPEND	2,359 ± 0,773	10 ± 1,717	6 ± 0
	RESUME	1,821 ± 0,404	13 ± 0,745	4 ± 0
	STOP	3,291 ± 0,874	16 ± 2,985	9,8 ± 0,761
	SHELVE	1129,658 ± 5,195	85 ± 10,845	37,767 ± 1,695
Centos 7 (1300 MB)	CREATE	1419,955 ± 0,709	100 ± 2,541	48,6 ± 1,776
	SUSPEND	3,232 ± 0,293	13 ± 1,453	8,1 ± 0,316
	RESUME	2,007 ± 0,215	13 ± 0,407	4,1 ± 0,316
	STOP	1,518 ± 0,21	6 ± 0,651	4 ± 0
	SHELVE	1424,304 ± 0,639	108 ± 1,442	57,2 ± 1,229
Centos 7 (898 MB)	CREATE	876,562 ± 0,738	79 ± 3,886	33 ± 1,563
	SUSPEND	2,736 ± 0,324	11 ± 2,535	6,3 ± 0,675
	RESUME	2,031 ± 0,255	14 ± 0,548	4 ± 0
	STOP	2,556 ± 0,308	11 ± 3,218	6 ± 0
	SHELVE	936,403 ± 0,453	82 ± 3,806	39,8 ± 1,317
Cirros	CREATE	25,65 ± 0,559	55 ± 1,94	16,3 ± 0,675
	SUSPEND	1,85 ± 0,314	7 ± 2,533	4 ± 0
	RESUME	1,998 ± 0,198	14 ± 2,583	4 ± 0
	STOP	22,649 ± 0,187	96 ± 0,675	63,2 ± 0,422
	SHELVE	33,749 ± 0,271	31 ± 2,833	7,2 ± 0,632
Debian 10	CREATE	592,787 ± 0,676	78 ± 2,644	32 ± 0,667
	SUSPEND	2,563 ± 0,384	9 ± 0,801	6 ± 0
	RESUME	1,989 ± 0,139	13 ± 1,031	4 ± 0
	STOP	2,339 ± 0,509	10 ± 3,059	5,6 ± 0,843
	SHELVE	1539,051 ± 0,808	119 ± 2,062	63 ± 1,563
Fedora 31	CREATE	368,741 ± 2,292	71 ± 5,238	24,633 ± 1,608
	SUSPEND	2,359 ± 0,808	11 ± 2,606	6 ± 0
	RESUME	1,803 ± 0,405	13 ± 0,838	3,967 ± 0,183
	STOP	1,999 ± 0,536	10 ± 2,684	5,767 ± 0,728
	SHELVE	993,256 ± 3,7	73 ± 9,138	30,033 ± 0,669
Fedora 32	CREATE	317,146 ± 0,667	71 ± 2,282	26,4 ± 0,968
	SUSPEND	3,29 ± 0,251	13 ± 1,437	8,067 ± 0,254
	RESUME	1,918 ± 0,311	13 ± 0,89	3,967 ± 0,183
	STOP	2,44 ± 0,372	10 ± 2,683	5,867 ± 0,507
	SHELVE	853,332 ± 0,797	83 ± 4,452	40 ± 1,145
Ubuntu Focal Fossa	CREATE	555,52 ± 1,143	75 ± 4,109	28,8 ± 1,229
	SUSPEND	4,317 ± 0,6	17 ± 3,607	10,2 ± 0,632
	RESUME	2,17 ± 0,282	14 ± 1,96	4 ± 0
	STOP	4,318 ± 0,607	17 ± 3,161	10,2 ± 0,632
	SHELVE	1391,565 ± 0,695	118 ± 3,347	61,7 ± 1,947
FreeBSD 12	CREATE	487,157 ± 1,812	66 ± 4,554	23,367 ± 1,991
	SUSPEND	1,329 ± 0,419	7 ± 2,139	4 ± 0
	RESUME	1,675 ± 0,51	13 ± 1,597	4,067 ± 0,365
	STOP	18,031 ± 0,567	97 ± 2,837	62,833 ± 0,379
	SHELVE	483,855 ± 1,243	44 ± 3,809	15,533 ± 0,507
MS Windows Server	CREATE	6645,582 ± 1,593	163 ± 10,563	94,9 ± 2,771
	SUSPEND	1,24 ± 0,433	6 ± 1,517	4 ± 0
	RESUME	1,829 ± 0,368	13 ± 1,234	4 ± 0
	STOP	18,084 ± 0,463	98 ± 2,511	62,933 ± 0,254
	SHELVE	6668,887 ± 7,262	230 ± 6,801	137,967 ± 4,03

Fonte: O próprio autor.

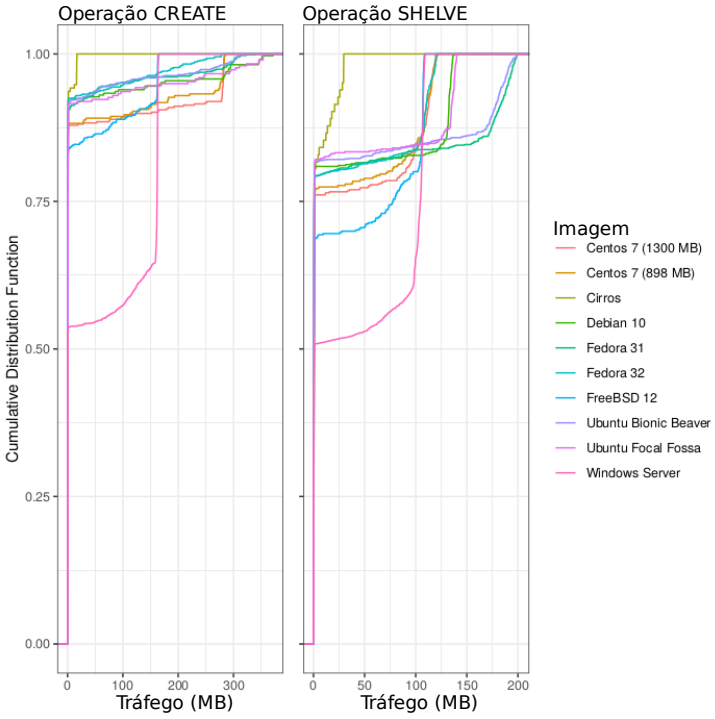
A Tabela 4 contém os resultados obtidos por imagem do SO. Os gráficos das Figuras 19 e 20 trazem, respectivamente, um *box plot* e um CDF para melhor detalhamento dos dados coletados durante as operações CREATE e SHELVE, que foram as duas operações que resultaram em maior tráfego de rede, conforme Tabela 4. O desvio padrão obtido a partir de 30 observações mostra, para todas as métricas da Tabela 4, que não houveram observações discrepantes que invalidem os dados cole-

Figura 19 – *Box plot* do volume de tráfego (MB) por segundo para cada uma das imagens de SO.



Fonte: O próprio autor.

Figura 20 – Gráfico CDF do volume de tráfego (MB) por segundo para cada uma das imagens de SO.



Fonte: O próprio autor.

tados. Em outras palavras, o desvio padrão mostra que há certa constância durante cada iteração dos experimentos e, portanto, valida a confiabilidade dos dados levantados pela experimentação.

A Figura 19 mostra a distribuição dos dados coletados. Os *outliers* indicam picos de medição do tráfego por segundo. Intervalos de um segundo, nos quais o somatório do tráfego medido caracteriza um pico em relação ao resto da distribuição. Principalmente durante a operação CREATE, nota-se certa constância na distribuição dos dados. O tráfego medido mantém-se baixo com diversos picos. O que ocorre com a operação SHELVE não é muito diferente. Contudo, é perceptível que os valores medidos encontram-se mais distribuídos do que na operação CREATE. Muito embora, também é observa-se a medição de baixos valores de volume tráfego de rede com diversos picos. Em alguns casos, tanto na operação CREATE quando na SHELVE, os *box plots* não apresentam *outliers*, como é o caso do MS-Windows. Isso ocorre devido à medição constante de altos valores de tráfego de rede. Constância essa, que se mantém por boa parte da duração da operação. Transformando, portanto, o que seriam valores *outliers*, em observações do Q3 do *box plot*. Neste sentido, o gráfico da Figura 20 ajuda a explorar melhor os picos de medição de tráfego por segundo.

Para melhor analisar e interpretar os dados medidos, associa-se o gráfico CDF da Figura 20 às informações obtidas com o gráfico *box plot* (Figura 19). A linha (em rosa) que descreve as experimentações com o MS-Windows, detalha o volume de tráfego medido por segundo com o decorrer das operações CREATE e SHELVE. Dessa forma, entende-se que em ambas as operações com a imagem do Windows, cerca de 50% da operação teve medição de tráfego próxima ou igual a 0 MB. Até a completude da operação CREATE nota-se um pico de aproximadamente 160 MB. Enquanto na operação SHELVE, tem-se um pico de aproximadamente 110 MB. Com as demais imagens (com exceção do FreeBSD, cujo caso é semelhante ao MS-Windows, na operação SHELVE), as linhas que descrevem a medição de tráfego na rede representam crescimento por menos de $1/4$ da operação.

As Tabelas 5 e 6 detalham os resultados da Tabela 4, separando, respectivamente, as chamadas API e o volume de tráfego por serviço OpenStack. Conforme explicado na Subseção 4.2, utiliza-se a técnica de classificação de tráfego baseada em portas de comunicação TCP. Neste sentido, faz-se um mapeamento de portas e serviços OpenStack com API REST. Além disso, também é feito um mapeamento relacionando aos serviços que fazem uso do RabbitMQ, conforme detalhado na Subseção 4.2. Portanto, alguns serviços que produzem tráfego na rede, podem não aparecer na tabela de chamadas API. Uma vez que o tráfego produzido por esses serviços é produto do AMQP.

A Tabela 5 traz um levantamento de chamas API feitas a serviços OpenStack durante o ciclo de vida induzido das MVs. O levantamento das chamadas é feito com base nas portas TCP usadas por cada API. É importante ressaltar que a implementação do ciclo de vida induzido das MVs pode influenciar no montante de chamadas

API realizadas. Não são feitos experimentos para comparação entre diferentes implementações. Contudo, cada implementação própria do cenário (ciclo de vida induzido) pode levar a diferentes resultados. Neste sentido, utiliza-se a interface *Connection*, do Python *Software Development Kit* (SDK) do OpenStack, a fim de obter acesso aos serviços necessários. Outro ponto característico da implementação utilizada no ciclo de vida induzido é a constante verificação do estado da máquina virtual, com o objetivo de identificar o término das operações. Portanto, uma implementação diferente pode guiar à medição de valores diferentes na Tabela 5. Na sequência, a Tabela 6 apresenta os dados resultantes da medição de volume de tráfego separado por serviço OpenStack, que podem ser interpretados em conjunto à Tabela 5. A Tabela 6 traz um resumo dos serviços classificados. A tabela com todos os serviços classificados encontra-se disponível no Anexo A.

Em análise à Tabela 6, é perceptível o considerável volume de tráfego gerado pelo serviço Glance, que é responsável pelo gerenciamento das imagens das instâncias, em comparação aos demais serviços durante as operações CREATE e SHELVE. Além disso, principalmente durante a criação das instâncias, nota-se que o volume de tráfego produzido pelo Glance gira em torno do tamanho da imagem utilizada no experimento. Por exemplo, o tamanho da imagem utilizada do MS-Windows Server é de 6150 MB, enquanto o tráfego produzido pelo Glance é de aproximadamente $6615,876$. Portanto, pode-se assumir que o volume de tráfego puramente administrativo é obtido pela subtração do tamanho da imagem do montante total de tráfego medido (*e.g.*, no caso do MS-Windows Server, $6615,876 - 6150$). O que ocorre na operação SHELVE é semelhante à criação da instância. Contudo, ao invés da transferência da imagem do SO, tem-se a transferência do *snapshot* da instância.

Algumas outras considerações ainda podem ser feitas sobre a Tabela 6. As células, cujo valor indica **NA**, são produtos de eventos que não foram observados o suficiente para obter-se a média e/ou desvio padrão (*sd*). O mesmo ocorre na Tabela 5. Por exemplo, dentre trinta experimentações, apenas uma delas mede tráfego do RabbitMQ. Dessa forma, o desvio padrão destas medições é marcado como **NA** (Tabela 6). Além disso, a coluna **MISC** agrega serviços gerais não caracterizados, como o MySQL.

Tabela 5 – Chamadas API / Serviço (Média ± *sd*).

Operação	Imagem	Glance	Keystone	Neutron	Nova
CREATE	Ubuntu Bionic Beaver	4 ± 0	9 ± 1,416	41 ± 1,589	14 ± 0,89
	Centos 7 (1300 MB)		13 ± 1,252	60 ± 1,494	24 ± 0,816
	Centos 7 (898 MB)		11 ± 1,636	47 ± 2,066	18 ± 1,252
	Cirros		10 ± 1,265	32 ± 0,85	10 ± 0,675
	Debian 10		10 ± 1,033	47 ± 2,348	18 ± 0,699
	Fedora 31		10 ± 1,655	41 ± 1,691	14 ± 0,986
	Fedora 32		10 ± 1,042	42 ± 1,524	15 ± 0,855
	Ubuntu Focal Fossa		13 ± 2,003	43 ± 2,394	16 ± 0,738
	FreeBSD 12		11 ± 2,074	39 ± 2,313	13 ± 0,95
	MS Windows Server		14 ± 2,141	100 ± 10,785	47 ± 1,502
SUSPEND	Ubuntu Bionic Beaver	NA	1 ± NA	6 ± 1,143	4 ± 0,484
	Centos 7 (1300 MB)		2 ± 0,707	8 ± 0,568	5 ± 0,422
	Centos 7 (898 MB)		1 ± NA	8 ± 2,685	4 ± 0,316
	Cirros		2 ± NA	5 ± 2,406	3 ± 0,316
	Debian 10		1 ± NA	6 ± 0,699	3 ± 0
	Fedora 31		1 ± 0	6 ± 1,424	4 ± 0,484
	Fedora 32		2 ± 0,707	8 ± 1,39	5 ± 0,379
	Ubuntu Focal Fossa		2 ± NA	12 ± 2,908	6 ± 0,675
	FreeBSD 12		NA	5 ± 2,132	3 ± 0,183
	MS Windows Server		1 ± 0	4 ± 1,455	3 ± 0,254
RESUME	Ubuntu Bionic Beaver		1 ± NA	11 ± 0,774	2 ± 0
	Centos 7 (1300 MB)		NA	11 ± 0,422	
	Centos 7 (898 MB)		1 ± NA	11 ± 0,316	3 ± 0,316
	Cirros		NA	12 ± 2,677	2 ± 0
	Debian 10			11 ± 0,85	3 ± 0,422
	Fedora 31		1 ± 0	11 ± 0,679	3 ± 0,183
	Fedora 32		2 ± 0,548	11 ± 0,466	
	Ubuntu Focal Fossa		3 ± NA	11 ± 0,816	3 ± 0,422
	FreeBSD 12		2 ± NA	11 ± 1,57	3 ± 0,254
	MS Windows Server		2 ± 0,707	11 ± 0,724	3 ± 0,346
STOP	Ubuntu Bionic Beaver		2 ± 0,535	11 ± 2,614	6 ± 0,607
	Centos 7 (1300 MB)		NA	4 ± 0,632	3 ± 0,316
	Centos 7 (898 MB)			8 ± 3,373	4 ± 0,316
	Cirros		2 ± 0,447	65 ± 0,422	30 ± 0
	Debian 10		2 ± 0,707	6 ± 2,53	3 ± 0,667
	Fedora 31		1 ± 0	7 ± 2,811	3 ± 0,403
	Fedora 32			7 ± 2,479	3 ± 0,32
	Ubuntu Focal Fossa		2 ± 0,707	12 ± 2,983	6 ± 0,483
	FreeBSD 12		2 ± 1,166	66 ± 1,213	31 ± 0,669
	MS Windows Server		2 ± 1,029	66 ± 1,474	31 ± 0,583
SHELVE	Ubuntu Bionic Beaver	10 ± 0,183	6 ± 1,383	46 ± 4,071	19 ± 1,159
	Centos 7 (1300 MB)	9 ± 0	6 ± 0,85	66 ± 0,789	27 ± 0,471
	Centos 7 (898 MB)		8 ± 1,269	47 ± 3,498	20 ± 0,919
	Cirros		5 ± 0,516	14 ± 2,547	4 ± 0,422
	Debian 10		6 ± 0,675	74 ± 1,43	31 ± 0,738
	Fedora 31	10 ± 0,183	6 ± 1,184	38 ± 3,845	15 ± 0,615
	Fedora 32		6 ± 1,599	49 ± 3,83	20 ± 0,964
	Ubuntu Focal Fossa	9 ± 0	7 ± 1,524	72 ± 2,716	31 ± 0,738
	FreeBSD 12		5 ± 0,968	23 ± 3,358	8 ± 0,479
	MS Windows Server		6 ± 0,997	150 ± 4,938	66 ± 2,069

Fonte: O próprio autor.

Tabela 6 – Tabela resumida de Volume de Tráfego (MB) / Serviço (Média $\pm$ *sd*). Tabela completa no Anexo A.

Operação	Imagem	Cinder	Glance	Keystone	MISC	Neutron	Nova
CREATE	Ubuntu Bionic Beaver	0,008 +/- 0,007	346,61 +/- 0,01	0,14 +/- 0,016	9,552 +/- 1,605	0,345 +/- 0,159	1,11 +/- 0,194
	Centos 7 (1300 MB)	0,018 +/- 0,015	1398,295 +/- 0,093	0,195 +/- 0,027	18,041 +/- 0,919	0,902 +/- 0,36	1,739 +/- 0,449
	Centos 7 (898 MB)	0,01 +/- 0,006	860,562 +/- 0,123	0,163 +/- 0,027	13,229 +/- 1,015	0,55 +/- 0,293	1,362 +/- 0,232
	Cirros	0,003 +/- 0,004	15,784 +/- 0,004	0,145 +/- 0,026	7,938 +/- 0,404	0,434 +/- 0,276	1,129 +/- 0,04
	Debian 10	0,01 +/- 0,007	575,644 +/- 0,005	0,146 +/- 0,023	14,482 +/- 0,537	0,481 +/- 0,164	1,366 +/- 0,357
	Fedora 31	0,007 +/- 0,006	356,159 +/- 0,017	0,137 +/- 0,027	9,547 +/- 1,734	0,374 +/- 0,236	1,025 +/- 0,209
	Fedora 32	0,009 +/- 0,009	303,511 +/- 0,011	0,141 +/- 0,023	11,6 +/- 0,638	0,362 +/- 0,135	0,988 +/- 0,236
	Ubuntu Focal Fossa	0,005 +/- 0,005	539,857 +/- 0,003	0,159 +/- 0,033	13,408 +/- 1,038	0,447 +/- 0,177	1,094 +/- 0,185
	FreeBSD 12	0,008 +/- 0,005	476,316 +/- 0,003	0,151 +/- 0,037	8,58 +/- 1,673	1,051 +/- 0,155	0,828 +/- 0,15
	Windows Server	0,024 +/- 0,006	6615,876 +/- 1,037	0,243 +/- 0,033	25,98 +/- 1,927	1,16 +/- 0,606	1,429 +/- 1,006
SUSPEND	Ubuntu Bionic Beaver	0,001 +/- 0,002	0,003 +/- 0,002	0 +/- 0,001	1,775 +/- 0,556	0,035 +/- 0,042	0,156 +/- 0,034
	Centos 7 (1300 MB)	0,005 +/- 0,006		0,012 +/- 0,02	2,808 +/- 0,327	0,127 +/- 0,111	0,185 +/- 0,119
	Centos 7 (898 MB)	0,002 +/- 0,003		0,004 +/- 0,007	2,436 +/- 0,345	0,059 +/- 0,028	0,153 +/- 0,021
	Cirros	0,003 +/- 0,004	0,001 +/- 0,001	0,012 +/- 0,017	1,599 +/- 0,288	0,03 +/- 0,008	0,181 +/- 0,07
	Debian 10	0,003 +/- 0,005	0,002 +/- 0,002	0,002 +/- NA	2,237 +/- 0,38	0,076 +/- 0,126	0,104 +/- 0,045
	Fedora 31	0,001 +/- 0,003	0,001 +/- 0,001	0,012 +/- 0	1,694 +/- 0,579	0,038 +/- 0,041	0,145 +/- 0,034
	Fedora 32	0,006 +/- 0,007	0,004 +/- 0,005	0,006 +/- 0,01	2,937 +/- 0,277	0,071 +/- 0,085	0,116 +/- 0,074
	Ubuntu Focal Fossa	0,003 +/- 0,004	0,003 +/- 0,003	0,008 +/- 0,014	3,815 +/- 0,513	0,097 +/- 0,119	0,155 +/- 0,09
	FreeBSD 12	0,002 +/- 0,004	0 +/- 0	0 +/- 0	1,054 +/- 0,425	0,174 +/- 0,053	0,072 +/- 0,025
	Windows Server		0,003 +/- 0,005	0,001 +/- 0,003	1,03 +/- 0,422	0,085 +/- 0,05	0,064 +/- 0,015
RESUME	Ubuntu Bionic Beaver	0,001 +/- 0,004	0,001 +/- 0,002	0 +/- 0	1,46 +/- 0,395	0,041 +/- 0,057	0,2 +/- 0,041
	Centos 7 (1300 MB)	0,009 +/- 0,01		NA	1,626 +/- 0,22	0,056 +/- 0,021	0,207 +/- 0,055
	Centos 7 (898 MB)	0,003 +/- 0,005	0 +/- 0	0,001 +/- NA	1,624 +/- 0,238	0,093 +/- 0,108	0,241 +/- 0,11
	Cirros	0 +/- 0	0,001 +/- 0,001	NA	1,733 +/- 0,189	0,032 +/- 0,017	0,194 +/- 0,019
	Debian 10	0,004 +/- 0,004	0,005 +/- 0,009		1,671 +/- 0,147	0,052 +/- 0,025	0,128 +/- 0,046
	Fedora 31	0,003 +/- 0,004	0 +/- 0	0,006 +/- 0,007	1,383 +/- 0,397	0,051 +/- 0,047	0,189 +/- 0,033
	Fedora 32		0,002 +/- 0,004	0,012 +/- 0,011	1,646 +/- 0,287	0,033 +/- 0,018	0,106 +/- 0,037
	Ubuntu Focal Fossa	0,002 +/- 0,003	0,002 +/- 0,002	0,037 +/- NA	1,863 +/- 0,309	0,035 +/- 0,017	0,127 +/- 0,087
	FreeBSD 12	0,002 +/- 0,004	0 +/- 0	0,02 +/- 0,033	1,345 +/- 0,505	0,205 +/- 0,05	0,094 +/- 0,029
	Windows Server	0,002 +/- 0,003	0,002 +/- 0,004	0,013 +/- 0,021	1,545 +/- 0,345	0,123 +/- 0,067	0,1 +/- 0,033
STOP	Ubuntu Bionic Beaver	0,003 +/- 0,004	0,002 +/- 0,002	0,016 +/- 0,018	2,823 +/- 0,91	0,053 +/- 0,02	0,198 +/- 0,068
	Centos 7 (1300 MB)	0,001 +/- 0,002	0,001 +/- 0,001	NA	1,253 +/- 0,229	0,046 +/- 0,046	0,141 +/- 0,059
	Centos 7 (898 MB)	0,004 +/- 0,004	0,002 +/- 0,002		2,195 +/- 0,306	0,072 +/- 0,129	0,192 +/- 0,09
	Cirros	0,014 +/- 0,005	0,02 +/- 0,003	0,013 +/- 0,001	20,769 +/- 0,301	0,451 +/- 0,166	0,795 +/- 0,105
	Debian 10	0 +/- 0	0,002 +/- 0,002	0,009 +/- 0,014	2,086 +/- 0,49	0,027 +/- 0,012	0,1 +/- 0,083
	Fedora 31	0,001 +/- 0,004	0,001 +/- 0,001	0,009 +/- 0,009	1,607 +/- 0,548	0,039 +/- 0,043	0,138 +/- 0,047
	Fedora 32	0,004 +/- 0,005	0,002 +/- 0,002	0,007 +/- 0,007	2,161 +/- 0,367	0,032 +/- 0,012	0,093 +/- 0,051
	Ubuntu Focal Fossa	0,004 +/- 0,004	0,001 +/- 0,002	0,009 +/- 0,014	3,94 +/- 0,574	0,124 +/- 0,144	0,098 +/- 0,036
	FreeBSD 12	0,014 +/- 0,005	0,004 +/- 0,004	0,018 +/- 0,023	15,12 +/- 0,663	1,939 +/- 0,182	0,482 +/- 0,205
	Windows Server	0,016 +/- 0,006	0,018 +/- 0,009	0,024 +/- 0,018	16,308 +/- 0,771	0,664 +/- 0,426	0,486 +/- 0,123
SHELVE	Ubuntu Bionic Beaver	0,01 +/- 0,006	1113,557 +/- 0,088	0,072 +/- 0,02	11,663 +/- 1,583	0,307 +/- 0,141	0,824 +/- 0,114
	Centos 7 (1300 MB)	0,029 +/- 0,028	1401,211 +/- 0,56	0,067 +/- 0,01	20,03 +/- 0,874	0,79 +/- 0,334	1,423 +/- 0,537
	Centos 7 (898 MB)	0,013 +/- 0,013	919,581 +/- 0,527	0,089 +/- 0,014	14,844 +/- 0,441	0,462 +/- 0,242	0,898 +/- 0,133
	Cirros	0,004 +/- 0,005	28,721 +/- 0,027	0,056 +/- 0,008	4,087 +/- 0,345	0,118 +/- 0,082	0,587 +/- 0,052
	Debian 10	0,018 +/- 0,005	1512,123 +/- 0,106	0,062 +/- 0,011	24,048 +/- 0,655	0,728 +/- 0,214	0,971 +/- 0,221
	Fedora 31	0,01 +/- 0,008	979,696 +/- 0,071	0,064 +/- 0,011	9,809 +/- 1,647	0,269 +/- 0,143	0,654 +/- 0,1
	Fedora 32	0,015 +/- 0,013	836,53 +/- 0,118	0,07 +/- 0,019	14,959 +/- 0,759	0,339 +/- 0,159	0,704 +/- 0,285
	Ubuntu Focal Fossa	0,019 +/- 0,008	1365,056 +/- 0,087	0,081 +/- 0,018	23,964 +/- 0,848	0,697 +/- 0,108	0,799 +/- 0,25
	FreeBSD 12	0,003 +/- 0,004	476,908 +/- 0,084	0,058 +/- 0,012	5,598 +/- 1,214	0,669 +/- 0,124	0,43 +/- 0,122
	Windows Server	0,032 +/- 0,004	6406,869 +/- 1208,735	0,064 +/- 0,016	37,046 +/- 1,885	1,54 +/- 0,951	222,094 +/- 1209,865

Fonte: O próprio autor.

Por fim, a análise do tráfego de rede gerado por imagem do SO mostra que há certa constância na rede. O tráfego na rede se mantém constante durante a maior parte do tempo. Embora os serviços de rede (Neutron) sejam os mais requisitados, em termos de números de chamadas API, são a transferência da imagem e do *snapshot* os responsáveis pelos picos de tráfego na rede. O que é confirmável através do volume de tráfego gerado pelo serviço Glance, responsável pelas imagens, que é condizente com os tamanhos das imagens.

5.2 ANÁLISE DE TRÁFEGO BASEADO NO TIPO DE *FLAVOR*

Nos experimentos, cuja análise do tráfego é feita por imagem do SO (Seção 5.1), fica clara a alteração no volume de tráfego gerado por cada imagem. Neste sentido, surge também uma dúvida quanto a possíveis alterações no volume de tráfego durante o ciclo de vida induzido das MVs adotando diferentes *flavors*. O *flavor* da MV consiste em uma alocação básica de recursos para a máquina. A gestão de unidades armazenamento da máquina é feita à parte, via *volumes*. Neste sentido, a Tabela 7 traz a configuração dos *flavors* padrão fornecidos pelo OpenStack. Os administradores podem sempre criar/configurar *flavors*.

Tabela 7 – Lista de *flavors* disponíveis por padrão no OpenStack.

Flavor	**VCPUs**	**Disco (GB)**	**RAM (MB)**
m1.tiny	1	1	512
m1.small	1	20	2048
m1.medium	2	40	4096
m1.large	4	80	8192
m1.xlarge	8	160	16384

Fonte: (OPENSTACK, 2020c)

Foram adotados os *flavors m1.small*, *m1.medium*, *m1.large* e *m1.xlarge* para experimentação. A imagem utilizada em todas as execuções é a GNU/Linux Ubuntu Server Bionic Beaver, também utilizada no experimento de variação de imagens (Seção 5.1). O *flavor m1.tiny* não foi utilizado, visto que não aloca recursos suficiente para criação da MV. A Tabela 8 apresenta um sumário dos dados coletados.

Em análise aos dados da Tabela 8, não foi possível identificar um padrão de crescimento no volume de tráfego de acordo com o aumento de recursos alocados pelo *flavor*. Além disso, os valores registrados para volume de tráfego total tendem a variar entre (aproximadamente) $358 \pm 1,7$ MB e 373 ± 1 MB. Portanto, a alteração no *flavor*, por si só, não apresenta alterações no volume de tráfego resultante do ciclo de

Tabela 8 – Resumo das medições feitas baseadas na alternância de tipo de *flavor*. Desvio padrão é representado por *sd*.

Imagem	Operação	Tráfego Total - MB (média ± sd)	Chamadas API (média ± sd)	Tempo de Exec. - segundos (média ± sd)
Ubuntu Bionic Beaver (*m1.small*)	CREATE	358,003 ± 1,663	67 ± 2,9	25,433 ± 1,455
	SUSPEND	2,032 ± 0,586	10 ± 1,464	6 ± 0
	RESUME	1,752 ± 0,436	13 ± 0,819	4 ± 0
	STOP	3,16 ± 0,968	16 ± 3,066	9,8 ± 0,761
	SHELVE	1126,785 ± 1,582	79 ± 5,279	37,767 ± 1,695
Ubuntu Bionic Beaver (*m1.medium*)	CREATE	373,715 ± 1,074	78 ± 4,247	30,7 ± 1,725
	SUSPEND	3,182 ± 0,398	13 ± 2,218	8,1 ± 0,305
	RESUME	2,047 ± 0,219	14 ± 1,813	4 ± 0
	STOP	3,447 ± 0,536	15 ± 3,431	8,567 ± 1,04
	SHELVE	1173,608 ± 1,114	103 ± 5,703	52,933 ± 2,196
Ubuntu Bionic Beaver (*m1.large*)	CREATE	368,394 ± 2,163	68 ± 3,256	24,4 ± 1,192
	SUSPEND	2,19 ± 0,799	13 ± 3,126	7,867 ± 0,507
	RESUME	1,548 ± 0,406	13 ± 1,991	4 ± 0
	STOP	1,976 ± 0,873	12 ± 3,178	7,467 ± 1,383
	SHELVE	1163,87 ± 1,966	81 ± 5,386	38,433 ± 1,524
Ubuntu Bionic Beaver (*m1.xlarge*)	CREATE	368,903 ± 1,613	67 ± 3,604	24,167 ± 1,289
	SUSPEND	2,564 ± 0,77	13 ± 2,091	8 ± 0
	RESUME	2,478 ± 0,508	17 ± 2,95	5,967 ± 0,183
	STOP	2,756 ± 0,791	13 ± 2,891	7,867 ± 1,042
	SHELVE	1172,323 ± 1,356	85 ± 4,241	41 ± 1,174

Fonte: O próprio autor.

vinda induzido das MVs. Contudo, o uso desses recursos extra, alocados pelo *flavor*, é que pode indicar alguma alteração no tráfego de rede medido. Dessa forma, experimentos que aplicam carga de memória e/ou processamento (que não é o propósito das observações aqui realizadas) podem medir as variações de tráfego na rede de acordo com as cargas empregadas.

A coleta e caracterização do tráfego de rede criado com variação no tipo de *flavor* das instâncias revela que não há nenhuma alteração significativa no volume de tráfego gerado na rede. Uma vez que a experimentação não considera o uso de cargas de memória ou disco, a simples alteração do *flavor* utilizado nas MVs não impacta diretamente no volume de tráfego gerado pelas operações do ciclo de vida induzido. Portanto, torna-se viável identificar uma *baseline* de tráfego que é aplicável independente do tipo de *flavor* adotado.

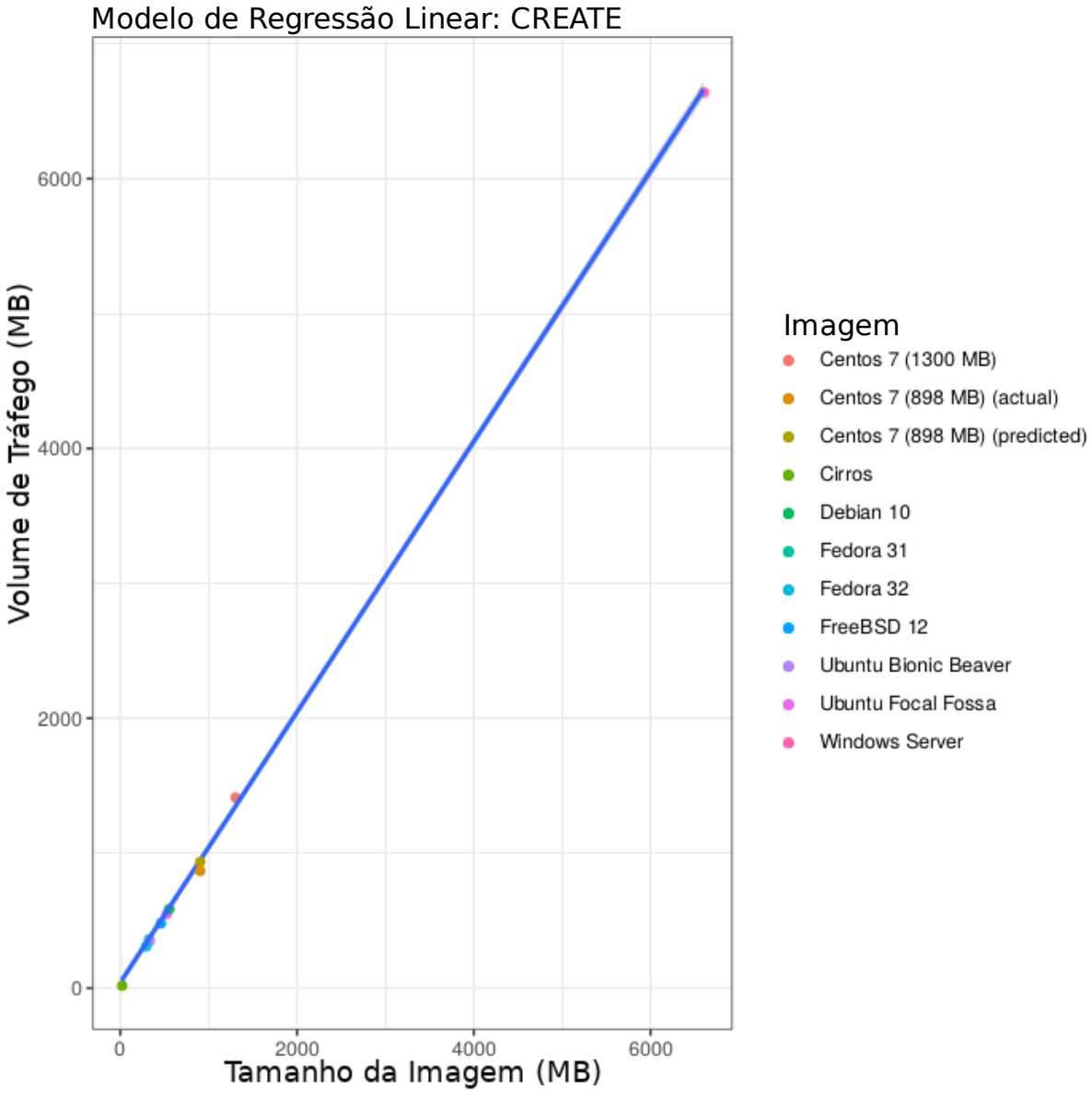
5.3 PREDIÇÃO DE TRÁFEGO

Os resultados obtidos com o experimento de variação de imagens (Seção 5.1) revelam que, tanto na operação CREATE quanto na operação SHELVE, o volume de tráfego na rede administrativa do OpenStack é influenciado pelo tamanho da imagem utilizada. Neste sentido, a criação de um modelo de regressão linear ajuda a entender a relação entre o crescimento do volume de tráfego em função do tamanho da imagem utilizada. A Figura 21 traz o modelo de regressão linear à operação CREATE, descrito por $y = 40,700864 + 1,002707x$. Já a Figura 22, apresenta o modelo de regressão linear à operação SHELVE, descrito por $y = 425,4478 + 0,9401x$. Em ambos os modelos, a variável de resposta é representada por Y (volume de tráfego em MB), enquanto a variável preditora representa-se por X (tamanho da imagem em MB). Os modelos de regressão linear são criados a partir dos dados obtidos com o experimento de variação

de imagens (Seção 5.1). São utilizadas 9 imagens para geração do modelo, mais uma última imagem para comparação dos resultados preditos com os que foram medidos durante as experimentações. A imagem para comparação entre valores preditos e reais é escolhida aleatoriamente.

No primeiro modelo (Figura 21), adota-se um nível de confiança de 90%. Os indicadores de precisão são bons, com *Min Max Accuracy* (MMX) $\approx 93\%$ e *Mean Absolute Percentage Error* (MAPE) $\approx 7\%$. O valor identificado para *intercept*, constante que indica o volume de tráfego esperado, se levado em consideração a média do tamanho de todas as imagens do conjunto de dados, é de $40,700864$. Em outras palavras, o volume de tráfego gerado pela média do tamanho das imagens é de $40,700864$. Já para o *slope*, que indica o efeito que o tamanho da imagem tem sobre o volume de tráfego gerado, registra-se $1,002707$. Ou seja, para cada 1 MB de imagem, o volume de tráfego produzido é aumentado em $1,002707$ MB. Além disso, os coeficientes do *slope* indicam que existe uma forte relação entre o tamanho da imagem e o tráfego de rede produzido. O coeficiente *Pr*, neste caso de valor $4e-14$, indica que há uma probabilidade muito baixa de existirem valores extremos que guiem para valores de coeficiente igual a 0 (chamada de *null hypothesis*). Neste caso, o valor de *Pr* é suficientemente baixo para que a *null hypothesis* seja discartada. Em relação aos coeficientes do *intercept*, a relação entre tamanho de imagem e volume de tráfego não é tão, mas ainda assim válida (*Pr* de $0,0139$). Além disso, o modelo de regressão linear para a operação CREATE apresentou índices *R-squared* e *p-value* fortemente significantes, de $0,9998$ e $3,997e-14$. O erro residual padrão encontrado é de $32,3$ MB com 7 graus de liberdade.

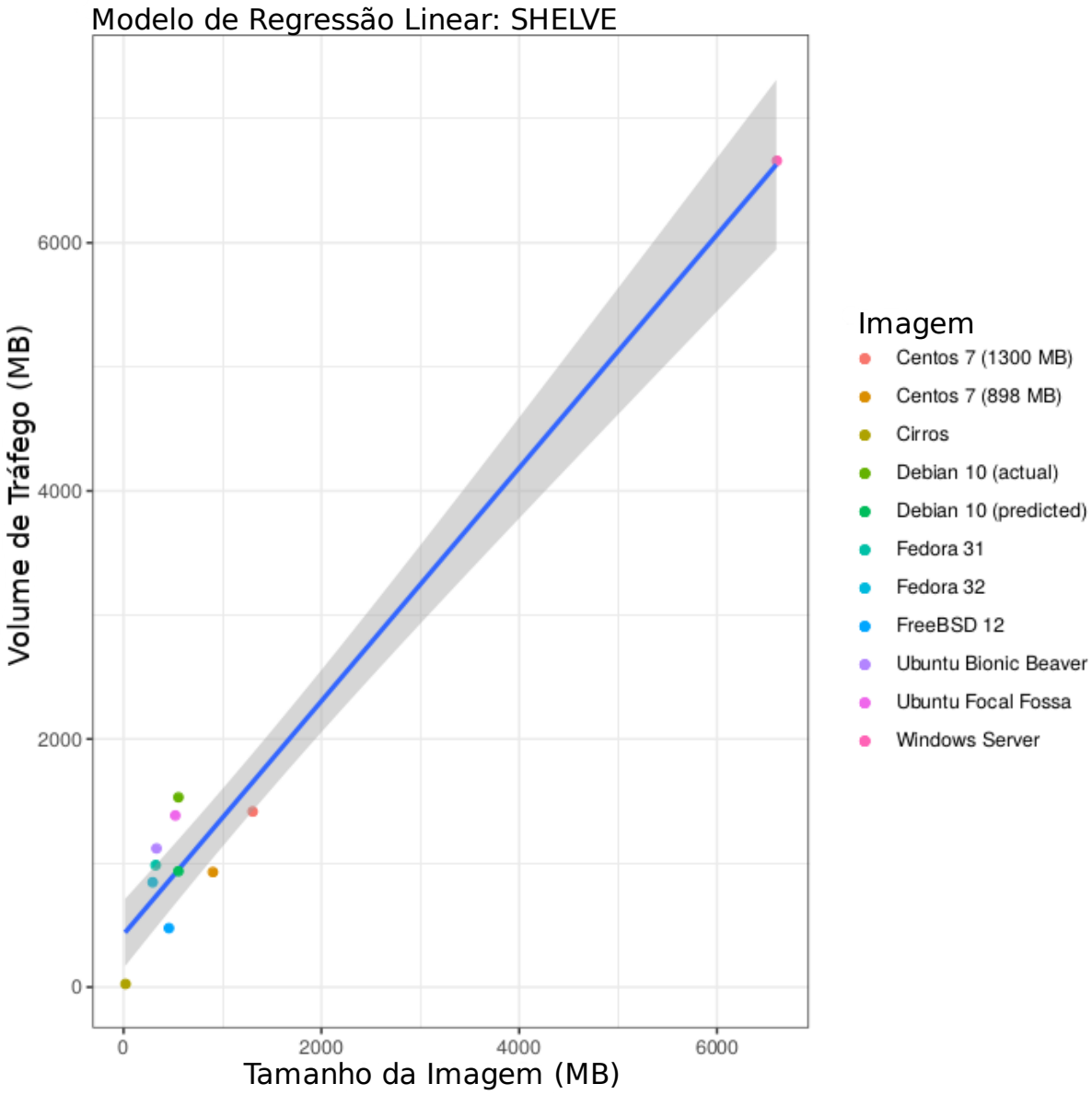
Figura 21 – Modelo de regressão linear da operação CREATE. A imagem CentOS 7, com 898 MB é utilizada para comparação dos valores reais com os valores preditos.



Fonte: O próprio autor.

Na operação SHELVE, o modelo de regressão linear (Figura 22) obteve indicadores de precisão não tão bons quanto o modelo da operação CREATE. Para um nível de confiança de 90%, os valores encontrados são: MMX MMX $\approx 61\%$ e MAPE $\approx 39\%$. Em relação ao *intercept* e *slope*, resgistra-se respectivamente $425,4478$ e $0,9401$. O valor *Pr* dos coeficientes do *intercept* indicam uma relação viável entre as variáveis preditora e resposta (X e Y), de $0,0208$. Já o *Pr* dos coeficientes do *slope* indicam uma forte relação entre o tamanho da imagem e o tráfego produzido, com valor de $1,43e-06$. Tanto *R-squared* quanto *p-value* são significantes no contexto, com valores de $0,9697$ and $1,425e-06$, respectivamente. O erro residual padrão encontrado é de $366,5$ MB com 7 graus de liberdade.

Figura 22 – Modelo de regressão linear da operação SHELVE. A imagem Debian 10 é utilizada para comparação dos valores reais com os valores preditos.



Fonte: O próprio autor.

Em análise aos resultados de ambos os modelos de regressão linear (Figuras 21 e 22), percebe-se que é possível ter boas noções sobre o volume de tráfego gerado na rede de acordo com o tamanho da imagem utilizada. Principalmente na operação CREATE, cujo modelo de regressão apresenta maior precisão. No entanto, para que os modelos tornem-se estatisticamente significantes, não apenas significantes ao contexto de estudo, é necessário um conjunto de dados maior. Trabalhar com um *dataset* mais amplo, com uma maior variedade de tamanhos de imagens, poderia guiar os modelos a melhores índices de precisão. Ainda assim, o uso dos modelos é viável para planejamento de recursos de redes (*e.g.*, largura de banda), ajudando a identificar o montante de tráfego produzido na rede durante as operações CREATE e

SHELVE.

5.4 CARACTERIZAÇÃO

De acordo com os resultados explorados, o que melhor descreve o tráfego de rede administrativo do OpenStack é a comunicação e organização entre os serviços da nuvem. Também é perceptível o volume de tráfego criado a partir da transferência da imagem/*snapshot* da instância (operações CREATE e SHELVE), que é variável de acordo com o tamanho da imagem utilizada. Por outro lado, o tráfego de comunicação inter-serviço mostrou fortes indícios em depender unicamente do tempo de execução da operação (mais perceptível nas operações SUSPEND, RESUME e STOP, já que não há tempo e tráfego extra para eventos como a transferência da imagem do SO, como ocorre na operação CREATE), que pode variar por diversos motivos (a execução da operação, por si só, depende principalmente de elementos do *hypervisor* e do próprio SO da instância). Ou seja, não foi identificado nenhum serviço do OpenStack destacando-se por produzir um volume excessivo de tráfego durante as operações do ciclo de vida induzido das MVs. Além disso, em todas as operações, o volume de tráfego **MISC**, que refere-se a serviços não caracterizados, como o MySQL, é predominante se desconsiderado o tráfego da imagem/*snapshot* do Glance (Tabela 6).

Em relação às chamadas API, o serviço Neutron é o mais requisitado em todas as operações. Também percebe-se um comportamento no qual o número de chamadas apresenta certa constância de acordo com o tempo de execução da operação. Foram observados casos, nos quais diferentes imagens apresentam um tempo de execução similar em determinada operação, que tendem a medir números de chamadas API muito próximos. Por exemplo, o experimento com a imagem FreeBSD 12 leva cerca de 63 segundos para finalizar a operação STOP, medindo um total de aproximadamente 97 ± 3 chamadas API (Tabela 4), das quais, 2 são do serviço Keystone, 66 do Neutron e 31 do Nova (Tabela 5). Analogamente, o experimento com a imagem MS-Windows Server 2012 R2 foi constatado o tempo da operação STOP em aproximadamente 63 segundos, com um total de chamadas API de aproximadamente 98 ± 3 (Tabela 4), das quais, 2 são do serviço Keystone, 66 do Neutron e 31 do Nova (Tabela 5). A Tabela 9 reúne alguns exemplos que reforçam a relação entre o tempo de execução da operação *vs.* chamadas API.

Levando em consideração o expressivo volume de tráfego resultante da transferência de imagens e *snapshots*, pode-se inserir uma rede de armazenamento ao domínio administrativo do OpenStack. A transferência de dados do usuário para persistência no Cinder, e.g., também pode ser feita através da rede de armazenamento. Embora o ciclo de vida induzido comprove a importância de uma rede dedicada à transferência da imagem/*snapshot*, não fica claro quanto à persistência de dados no

Tabela 9 – Exemplos de imagens diferentes que leval tempos similares para executar determinada operação, resultando em um número de chamadas API muito próximo. O desvio padrão é representado por *sd*.

Imagem	Operação	Chamadas API (média $\pm$ *sd*)	Tempo de Exec. - segundos (média $\pm$ *sd*)
Ubuntu Bionic Beaver	CREATE	67 $\pm$ 2,801	25,433 $\pm$ 1,455
FreeBSD 12		66 $\pm$ 4,554	23,367 $\pm$ 1,991
Ubuntu Bionic Beaver	SUSPEND	10 $\pm$ 1,717	6 $\pm$ 0
Debian 10		9 $\pm$ 0,801	6 $\pm$ 0
Fedora 32	RESUME	13 $\pm$ 0,89	3,967 $\pm$ 0,183
Ubuntu Focal Fossa		14 $\pm$ 1,96	4 $\pm$ 0
FreeBSD 12	STOP	97 $\pm$ 2,837	62,833 $\pm$ 0,379
MS Windows Server		98 $\pm$ 2,511	62,933 $\pm$ 0,254
Debian 10	SHELVE	119 $\pm$ 2,062	63 $\pm$ 1,563
Ubuntu Focal Fossa		118 $\pm$ 3,347	61,7 $\pm$ 1,947

Fonte: O próprio autor.

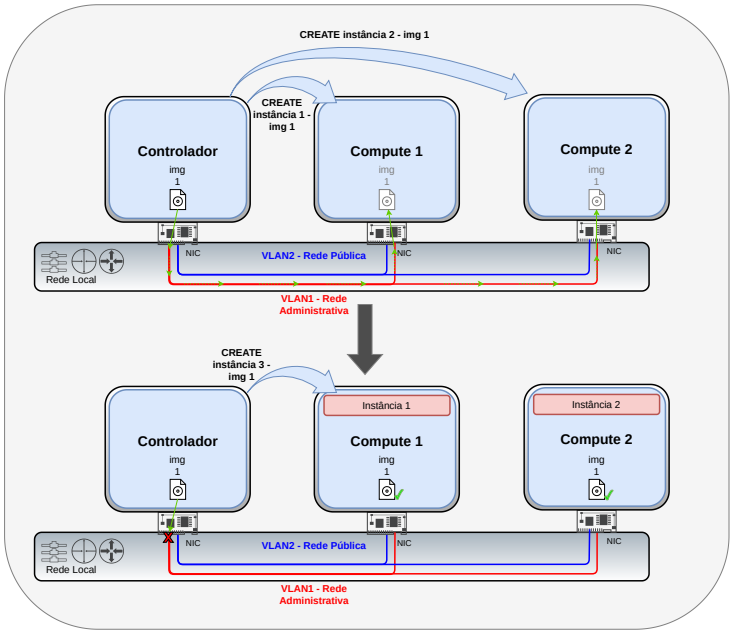
Cinder, Trove e Swift, por exemplo. Para fins de organização da rede, os serviços de armazenamento podem ser mantidos em uma rede dedicada, mesmo que a justificativa de altas taxas de transferência não possa ser utilizada para todos. Portanto, tem-se aqui uma característica (neste caso manifestando-se como uma limitação) do ciclo de vida induzido utilizado (Figura 15). O ciclo de vida induzido não toma ciência do uso das MVs (no sentido da existência de um usuário realizando atividades na máquina), apenas trata da *baseline* de tráfego oriundo de cada operação contemplada. Com o ciclo, não é possível caracterizar o tráfego de serviços que dependem do uso das MVs.

Para melhor explorar o termo *baseline*, o objetivo desta caracterização de tráfego no domínio administrativo do OpenStack é obter uma informação base sobre o tráfego gerado por operações em MVs. O conhecimento da *baseline* de tráfego gerado por operações que são comuns em MVs pode ser crucial no planejamento da nuvem. Entender o impacto de uma operação, por si só, abre caminho para análises mais amplas. É possível extrapolar os valores para aplicá-los em cenários mais complexos. Cenários nos quais é importante relembrar que em um ambiente de computação em nuvem, diversos usuários fazem uso dos recursos da infraestrutura da nuvem ao mesmo tempo. Portanto, serão diversos usuários que podem executar operações simultâneas em MVs.

Ao extrapolar os valores da *baseline* de tráfego, é necessário entender o cenário de aplicação (*e.g.*, nuvem OpenStack privada com 10 nós Compute e 2 nós Controladores). Também é preciso considerar o *caching* de imagens. Não faz sentido transmitir uma imagem pela rede cujo nó Compute já está em posse. No entanto, a imagem em *cache* eventualmente pode tornar-se obsoleta e a transmissão pela rede ser necessária novamente. Além disso, há cenários nos quais, pelos mais diversos motivos (até mesmo por questões legais), os consumidores podem fazer o pedido de

servidores específicos. A Figura 23 mostra um exemplo de funcionamento do *caching* de imagens na nuvem.

Figura 23 – Em um primeiro momento, são criadas duas instâncias de MV com a imagem *img 1*, que é enviada pela rede aos nós de computação. Em um segundo momento, cria-se uma terceira instância com a *img 1*. Nesta última, a imagem não precisa ser transferida.

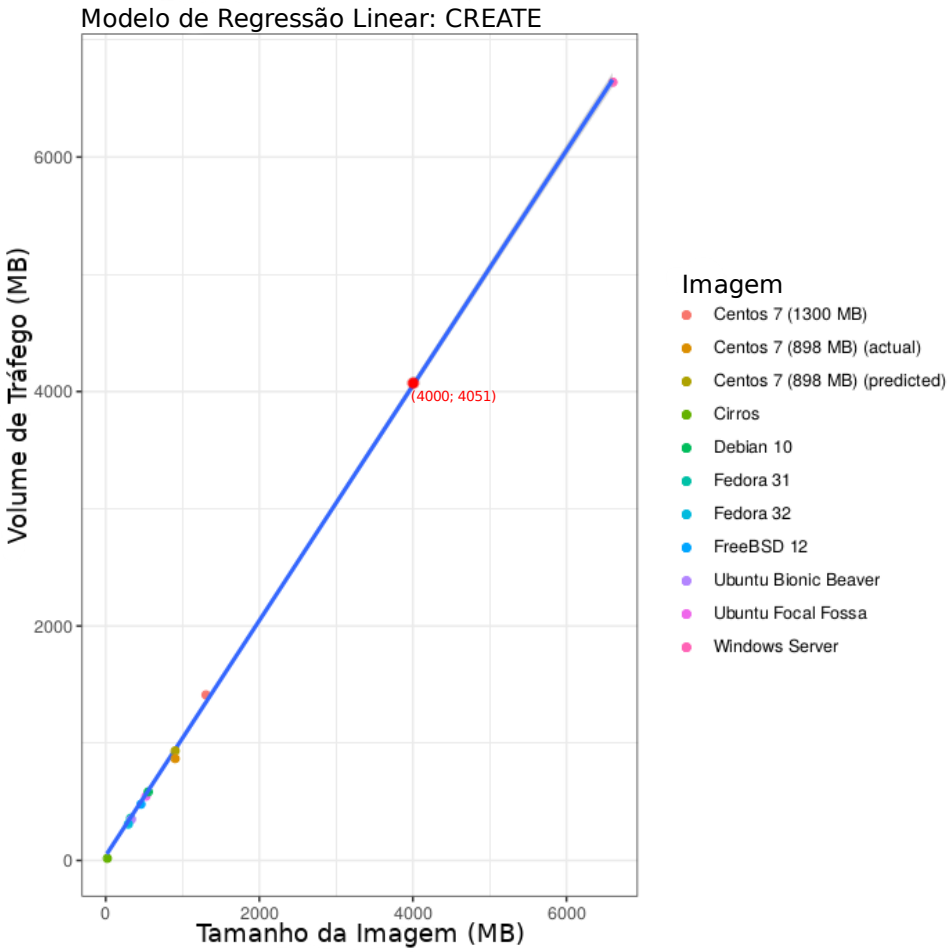


Fonte: O próprio autor.

A estimativa de volume de tráfego total necessário para criar (operação CREATE) uma instância de MV, por exemplo, pode ser calculada com o uso da *baseline* de tráfego para uma imagem do tamanho escolhido. Por exemplo, ao criar cinco instâncias do MS-Windows Server pela primeira vez, caso cada uma das instâncias seja alocada em um nó de computação diferente, será gerado na rede um tráfego total de $33227,91$ MB ($5 * 6645,582$, Tabela 4). Ou seja, aproximadamente $32,45$ GB de tráfego de rede somente com a criação da instância. Também é possível utilizar da predição de tráfego para fazer estimativas com imagens de tamanhos não contemplados pelos experimentos. Para uma imagem de SO com 4000 MB, aplicando na equação de regressão linear da operação CREATE ($y = 40,700864 + 1,002707x$, Subseção 5.3, Figura 21), tem-se $y = 4051,5289$ MB. A Figura 24 mostra a predição aplicada na reta. Portanto, ao criar 3 instâncias simultaneamente pela primeira vez e em servidores diferentes, mede-se aproximadamente $11,87$ GB de tráfego. As estimativas podem ser feitas em cenários mais complexos e também podem ser feitas com o SHELVE.

Informações sobre o perfil de uso dos consumidores (principais tarefas que realizam na nuvem, operações em MVs, imagens e *snapshots* utilizados, bem como os horários de uso) associadas à *baseline* de tráfego das operações em MVs possibilita uma gestão otimizada da rede da nuvem. Os administradores podem controlar a largura de banda da rede, por exemplo, de acordo com as necessidades identificadas

Figura 24 – Aplicando a regressão linear da operação CREATE para uma imagem de 4000 MB. $y = 40,700864 + 1,002707x$, na qual $x = 4000$. Portanto, $y = 4051,5289$.



Fonte: O próprio autor.

através das estimativas de volume de tráfego. Além disso, questões referentes à escalabilidade da nuvem podem ser melhor planejadas, como a necessidade de uma rede dedicada a armazenamento.

5.5 CONSIDERAÇÕES PARCIAIS

Durante o capítulo, são apresentados os resultados obtidos pelos experimentos de análise do tráfego por imagens do SO, análise do tráfego baseado no tipo de *flavor* e predição de tráfego. Além disso, é feita uma caracterização geral do tráfego administrativo do OpenStack. Os resultados mostram que o tráfego medido é resultante principalmente da conversação entre os serviços do OpenStack. Também, a maior parte do tráfego coletado diz respeito à transmissão das imagens/*snapshots* durante as operações CREATE e SHELVE. Por fim, a *baseline* do volume de tráfego produzido por operação ajuda no planejamento e gerenciamento do domínio administrativo da nuvem.

6 CONSIDERAÇÕES & TRABALHOS FUTUROS

Inicialmente, seguindo o escopo do projeto, o objetivo geral proposto pelo trabalho, de classificação/caracterização do tráfego da rede administrativa do OpenStack durante a mudança de estados das MVs em um ciclo de vida induzido, pode ser considerado totalmente atingido, visto que todos os experimentos propostos foram realizados. A Tabela 10 apresenta os objetivos específicos envolvidos.

Tabela 10 – Quadro de objetivos específicos envolvidos no trabalho. Mostra quanto atingiu-se de cada objetivo.

Objetivo específico (OE)	**Nível atingido**
1. Identificar sobre o funcionamento do OpenStack e como suas redes e serviços estão organizados	Totalmente
2. Identificar e usar uma arquitetura base de implantação do OpenStack	Totalmente
3. Coletar dados do tráfego da rede administrativa	Totalmente
4. Analisar dos dados coletados	Totalmente
5. Caracterizar/classificar o tráfego coletado por MV operação de mudança de estado	Totalmente

Analisando a Tabela 10, o **OE1** foi totalmente atingido. Um dos principais fatores que levaram à realização deste objetivo foi a documentação do OpenStack, que é muito completa e bem organizada (por vezes confusa, mas ainda assim de grande ajuda). Sobre o **OE2**, a documentação do OpenStack, leitura de artigos da área e o CloudLab, que conta com um perfil de implantação de nuvens OpenStack, foram os principais fatores contribuintes para que uma arquitetura base pudesse ser escolhida e utilizada durante as experimentações. Devido às técnicas de coleta de tráfego (*e.g.*, orientada a fluxos) identificou-se um método para medir o tráfego da rede administrativa do OpenStack (**OE3**). Em relação à análise dos dados coletados, **OE4**, o objetivo foi totalmente realizado. O tráfego de rede foi analisado no sentido de interpretar e dar significado aos dados coletados levando em consideração todo o cenário e contexto no qual os dados estavam inseridos. Uma das principais ferramentas que auxiliaram na análise dos dados foi a linguagem R, que foi utilizada para visualização, organização e estatísticas gerais dos dados. Por fim, o **OE5** também é classificado como totalmente atendido. Um dos maiores desafios na caracterização/classificação do tráfego coletado foi o RabbitMQ. As chamadas RPC assíncronas dificultaram bastante o uso do método baseado em portas durante a classificação dos pacotes. Outro desafio também superado foi a identificação com precisão do término da operação na MV.

O trabalho com nuvens OpenStack traz experiências que mostram a complexidade do funcionamento conjunto dos elementos que compõem um sistema distribuído. Foram feitas diversas implantações do OpenStack em máquinas disponibilizadas pela Universidade do Estado de Santa Catarina (UDESC), no Laboratório de

Processamento Paralelo Distribuído (LabP2D)[1], as quais envolvem desde a preparação do ambiente (*e.g.*, nós de computação, controle e equipamentos de rede) até a configuração dos *softwares* necessários. Neste sentido, são colocados em prática diversos conceitos aprendidos durante o Programa de Pós-Graduação em Computação Aplicada (PPGCA), principalmente relacionados à computação distribuída avançada e redes de computadores avançadas.

Em relação à execução dos experimentos, também foram encontradas algumas dificuldades como a identificação da rede administrativa, considerando que as configurações de rede do ambiente são feitas no perfil do experimento do OpenStack[2] no CloudLab, que foi superada acompanhando a documentação do perfil de experimento. Além disso, em alguns casos, *e.g.* nas 30 execuções do experimento com a imagem do MS-Windows Server 2012 R2, o volume de dados coletados (arquivos *.pcap*) excedia o espaço máximo de 100 GB na partição do servidor controlador. Neste caso, foi necessário capturar o tráfego, executar a análise dos pacotes e, em seguida, apagar o arquivo *.pcap*. Também houveram casos nos quais a duração do experimento com diversas execuções excedeu o limite de tempo padrão do CloudLab, que é de 16 horas. A coleta de dados para 30 execuções com a imagem do MS-Windows Server 2012 R2 necessitou a requisição de tempo extra na duração do experimento, levando em torno de 20 horas. Cada execução completa de um experimento com o MS-Windows Server leva em torno de 40 minutos, com o GNU/Linux Ubuntu Server (Bionic Beaver) uma execução completa leva em torno de 10 minutos. Aqui, uma execução completa do experimento considera todo o ciclo de vida induzido, coleta e análise do tráfego. Na análise, são armazenadas informações relevantes de cada pacote, além de cruzar dados de portas TCP com o mapa de portas do RabbitMQ.

O autor e seus orientadores agradecem o apoio do LabP2D, FAPESC e CloudLab.

6.1 TRABALHOS FUTUROS

Para trabalhos complementares/futuros, seria possível fazer medições de tráfego baseado no tipo de *flavor* aplicando cargas de RAM e/ou disco. Bem como testes com *caching* de imagens, que são viáveis para comparação do volume de tráfego e tempo de execução da criação da instância. Também é válida uma expansão do ciclo de vida induzido, visto que o atual ciclo cobre somente as operações mais comuns. O levantamento de um *dataset* maior (com mais imagens de SO de tamanhos variados) também impactaria positivamente na predição de tráfego, além de trazer mais dados para comparação durante uma análise geral. No demais, otimizações técnicas, tanto

[1] <http://labp2d.joinville.udesc.br/>
[2] <gitlab.flux.utah.edu/johnsond/openstack-build-ubuntu>

no ONM quanto nos *scripts* R auxiliares na análise dos dados, podem reduzir o tempo de experimentação e geração de relatórios dos dados, otimizando o crescimento do *dataset*.

6.2 CONTRIBUIÇÕES

As principais contribuições deste trabalho giram em torno do entendimento da rede administrativa do OpenStack. Portanto, tem-se:

- A análise e caracterização do tráfego no ambiente de rede mais interno do OpenStack é o produto final obtido, de acordo com o ciclo de vida induzido das MVs, que compreende as operações CREATE, SUSPEND, RESUME, STOP e SHELVE das insâncias;
- O monitor de rede utilizado, ONM[3], bem como as ferramentas intermediárias, o *lsof mapper*[4], para criar o mapa de portas usadas pelo RabbitMQ e os *scripts* em R[5], para análise e interpretação dos dados, também são partes da contribuição de todo o trabalho desempenhado; e
- A viabilidade na predição de volume de tráfego principalmente na criação de instâncias de MVs.

O ONM pode ser empregado em qualquer versão do OpenStack. Contudo, são necessárias adaptações de acordo com o cenário de utilização. Além disso, o ONM permite que o experimentador crie diferentes ciclos de vida para suas instâncias de MV, possibilitando até mesmo uma expansão do ciclo de vinda induzido atual. No mais, a coleta de tráfego também pode ser feita por outra ferramenta. Contudo, o ONM automatiza a transição de estados das MVs, além de armazenar os dados em um formato que é utilizado como entrada aos *scripts* em R para estudo dos dados.

Por fim, também contribui-se com a identificação da *baseline* de tráfego gerada pelas operações em MVs CREATE, SUSPEND, RESUME, STOP e SHELVE. Também é provado viável a predição de tráfego produzida pela operação CREATE com imagens cujos tamanhos não foram testados na prática. O volume de tráfego predito ajuda administradores de nuvens computacionais na melhor gestão da arquitetura de rede da nuvem. Além disso, também é possível fazer predição do tráfego de rede gerado pela operação SHELVE, embora com menor confiabilidade. De modo geral, o trabalho traz ganhos significativos no entendimento do comportamento da rede administrativa do OpenStack durante a execução de operações comuns em MVs.

[3] <github.com/Adnei/openstack_monitor>
[4] <github.com/Adnei/service_identifier>
[5] <github.com/Adnei/openstack_plots>

6.3 PRODUÇÕES

As seguintes publicações já foram realizadas:

- DONATTI, A. W.; MIERS, C. C. ; KOSLOVSKI, G. P. ; PILLON, M. A. . Network Traffic Characterization in the Control Network of OpenStack based on Virtual Machines State Changes. In: CLOSER - 10th International Conference on Cloud Computing and Services Science, 2020, Praga. CLOSER2020. Lisbon/Portugal: INSTICC, 2020. p. 1-8. Artigo. Qualis A2, Qualis CC/2016.
- DONATTI, A.; KOSLOVSKI, G.; PILLON, M. A.; MIERS, C. "Analysis and characterization of network traffic in the OpenStack management domain based on the virtual machine life cycle". SBRC 2019, LANCOMM (Latin American Student Workshop on Data Communication Networks). 2019. Gramado/RS - Brasil. Resumo. Qualis B4, Qualis CC/2016.
- DONATTI, A.; MIERS, C.; KOSLOVSKI, G. "Proposta de análise e caracterização do tráfego de rede no domínio de controle do OpenStack com foco no ciclo de vida das máquinas virtuais". 19a ERAD/RS (Escola Regional de Alto Desempenho / Região Sul), - Fórum de Pós-Graduação. Anais do ERAD/RS 2019. Três de Maio/RS - Brasil. Artigo.
- MIERS, C. C.; KOSLOVSKI, G. P.; PILLON, M. A.; DONATTI, A. W. "Análise do tráfego de máquinas virtuais na rede de controle de nuvens computacionais baseadas em OpenStack. 17a ERRC/RS (Escola Regional de Redes de Computadores/Região Sul). Anais do ERRC/RS 2019. 2019. Alegrete/RS - Brasil. Capítulo de Livro.
- DONATTI, A. W.; MIERS, C. C. ; KOSLOVSKI, G. P. . Caracterização do tráfego de rede no domínio de controle do OpenStack produzido pela mudança de estado de máquinas virtuais. In: XX Escola Regional de Alto Desempenho da Região Sul - ERAD/RS, 2020, Santa Maria. Anais do XX ERAD/RS. Porto Alegre RS: SBC, 2020. p. 1-2. Artigo.
- DONATTI, A. W.; MIERS, C. C. ; KOSLOVSKI, G. P. . Classificação do tráfego de rede do serviço de mensageria do OpenStack. In: XVIII Escola Regional de Redes de Computadores - ERRC, 2020. Anais do XVIII ERRC. Artigo.

REFERÊNCIAS

Aishwarya. K; Sankar, S. Traffic analysis using hadoop cloud. In: **2015 International Conference on Innovations in Information, Embedded and Communication Systems (ICIIECS)**. [S.l.: s.n.], 2015. p. 1–6.

ALENEZI, M.; ALMUSTAFA, K.; MEERJA, K. A. Cloud based sdn and nfv architectures for iot infrastructure. **Egyptian Informatics Journal**, v. 20, n. 1, p. 1 – 10, 2019. ISSN 1110-8665. Disponível em: <http://www.sciencedirect.com/science/article/pii/S1110866517303523>.

BOHN, R. B. et al. Nist cloud computing reference architecture. **2011 IEEE World Congress on Services**, p. 594–596, 2011.

BRAUN, H.-W.; CLAFFY, K. C. Web traffic characterization: an assessment of the impact of caching documents from ncsa's web server. **Computer Networks and ISDN Systems**, v. 28, n. 1, p. 37 – 51, 1995. ISSN 0169-7552. Selected Papers from the Second World-Wide Web Conference. Disponível em: <http://www.sciencedirect.com/science/article/pii/016975529500105X>.

Bruneo, D. A stochastic model to investigate data center performance and qos in iaas cloud computing systems. **IEEE Transactions on Parallel and Distributed Systems**, v. 25, n. 3, p. 560–569, March 2014.

CALLEGATI, F. et al. Performance of network virtualization in cloud computing infrastructures: The openstack case. In: . [S.l.: s.n.], 2014.

CHANDRAMOULI, R. **Security recommendations for hypervisor deployment**. [S.l.]: US Department of Commerce, National Institute of Standards and Technology, 2014.

Chaudhary, R. et al. Optimized big data management across multi-cloud data centers: Software-defined-network-based analysis. **IEEE Communications Magazine**, v. 56, n. 2, p. 118–126, Feb 2018.

Chen, D.; Zhao, H. Data security and privacy protection issues in cloud computing. In: **2012 International Conference on Computer Science and Electronics Engineering**. [S.l.: s.n.], 2012. v. 1, p. 647–651.

CHOWDHURY, N. M. K.; BOUTABA, R. A survey of network virtualization. **Computer Networks**, Elsevier, v. 54, n. 5, p. 862–876, 2010.

Claise, B.; Trammell, B.; Aitken, P. **Specification of the IP Flow Information Export (IPFIX) Protocol for the Exchange of Flow Information**. [S.l.], 2013. Disponível em: <https://www.ietf.org/rfc/rfc7011.txt>.

CLOUDLAB. 2019. Disponível em: <cloudlab.us>.

Dainotti, A.; Pescape, A.; Claffy, K. C. Issues and future directions in traffic classification. **IEEE Network**, v. 26, n. 1, p. 35–40, January 2012.

DAINOTTI, A.; PESCAPE, A.; VENTRE, G. A packet-level characterization of network traffic. In: **2006 11th International Workshop on Computer-Aided Modeling, Analysis and Design of Communication Links and Networks**. [S.l.: s.n.], 2006. p. 38–45. ISSN 2378-4865.

DAWSON, M. Red Hat Global Customer Tech Outlook 2019, **Red Hat Global Customer Tech Outlook 2019: Automation, cloud, & security lead funding priorities**. 2018. Disponível em: <https://www.redhat.com/en/blog/red-hat-global-customer-tech-outlook-2019-automation-cloud-security-lead-funding-priorities>.

DIAMANTI. 2018 container adoption benchmark survey. 2018.

DIAMANTI. 2019 container adoption benchmark survey. 2019.

Finsterbusch, M. et al. A survey of payload-based traffic classification approaches. **IEEE Communications Surveys Tutorials**, v. 16, n. 2, p. 1135–1156, Second 2014.

FLITTNER, M.; BAUER, R. Trex: Tenant-driven network traffic extraction for sdn-based cloud environments. In: **2017 Fourth International Conference on Software Defined Systems (SDS)**. [S.l.: s.n.], 2017. p. 48–53.

GILL, P. et al. Youtube traffic characterization: A view from the edge. In: **Proceedings of the 7th ACM SIGCOMM Conference on Internet Measurement**. New York, NY, USA: ACM, 2007. (IMC '07), p. 15–28. ISBN 978-1-59593-908-1. Disponível em: <http://doi.acm.org/10.1145/1298306.1298310>.

GUSTAMAS, R. G.; SHIDIK, G. F. Analysis of network infrastructure performance on cloud computing. In: **2017 International Seminar on Application for Technology of Information and Communication (iSemantic)**. [S.l.: s.n.], 2017. p. 169–174.

He, M. et al. Evaluating the control and management traffic in openstack cloud with sdn. In: **2019 IEEE 20th International Conference on High Performance Switching and Routing (HPSR)**. [S.l.: s.n.], 2019. p. 1–6.

Hofstede, R. et al. Flow monitoring explained: From packet capture to data analysis with netflow and ipfix. **IEEE Communications Surveys Tutorials**, v. 16, n. 4, p. 2037–2064, Fourthquarter 2014.

IBM GLOBAL SERVICES. The next-generation data center. Somers, NY, USA, set. 2016.

JADEJA, Y.; MODI, K. Cloud computing - concepts, architecture and challenges. In: **2012 International Conference on Computing, Electronics and Electrical Technologies (ICCEET)**. [S.l.: s.n.], 2012. p. 877–880.

Kreutz, D. et al. Software-defined networking: A comprehensive survey. **Proceedings of the IEEE**, v. 103, n. 1, p. 14–76, Jan 2015. ISSN 0018-9219.

KRUTZ, R. L.; VINES, R. D. **Cloud security: A comprehensive guide to secure cloud computing**. [S.l.]: Wiley Publishing, 2010.

Landfeldt, B.; Sookavatana, P.; Seneviratne, A. The case for a hybrid passive/active network monitoring scheme in the wireless internet. In: **Proceedings IEEE International Conference on Networks 2000 (ICON 2000). Networking Trends and Challenges in the New Millennium**. [S.l.: s.n.], 2000. p. 139–143.

Lin, P. et al. Profiling and accelerating string matching algorithms in three network content security applications. **IEEE Communications Surveys Tutorials**, v. 8, n. 2, p. 24–37, Second 2006.

MCGRATH, M. G.; RAYCROFT, P.; BRENNER, P. R. Intercloud networks performance analysis. In: **2015 IEEE International Conference on Cloud Engineering**. [S.l.: s.n.], 2015. p. 487–492.

MCKEOWN, N. et al. Openflow: Enabling innovation in campus networks. **SIGCOMM Comput. Commun. Rev.**, ACM, New York, NY, USA, v. 38, n. 2, p. 69–74, mar. 2008. ISSN 0146-4833. Disponível em: <http://doi.acm.org/10.1145/1355734.1355746>.

MELL, P. M.; GRANCE, T. Sp 800-145. the nist definition of cloud computing. National Institute of Standards & Technology, 2011.

NGUYEN, T. T. T.; ARMITAGE, G. A survey of techniques for internet traffic classification using machine learning. **IEEE Communications Surveys Tutorials**, v. 10, n. 4, p. 56–76, Fourth 2008. ISSN 1553-877X.

OPENSTACK. 2018 openstack user survey report. 2018. Disponível em: <https://www.openstack.org/user-survey/2018-user-survey-report/>.

OPENSTACK. Designing an openstack network. 2018. Disponível em: <https://docs.openstack.org/arch-design/design-networking/design-networking-design.html>.

OPENSTACK. Networking concepts. 2018. Disponível em: <https://docs.openstack.org/arch-design/design-networking/design-networking-concepts.html#traffic-flow>.

OPENSTACK. Provision an instance. 2018. Disponível em: <https://docs.openstack.org/operations-guide/ops-customize-provision-instance.html>.

OPENSTACK. Openstack documentation. 2019. Disponível em: <https://docs.openstack.org>.

OPENSTACK. Openstack software: Compute starter kit. 2019. Disponível em: <https://www.openstack.org/software/sample-configs#compute-starter-kit>.

OPENSTACK. Openstack survey report. 2019. Disponível em: <https://www.openstack.org/analytics>.

OPENSTACK. Server concepts. 2019. Disponível em: <https://docs.openstack.org/api-guide/compute/server_concepts.html>.

OPENSTACK. Vision for openstack clouds. 2019. Disponível em: <https://governance.openstack.org/tc/reference/technical-vision.html?_ga=2.75225810.1135708675.1574873893-711996610.1570721857>.

OPENSTACK. What is openstack? 2019. Disponível em: <https://www.openstack.org/software>.

OPENSTACK. **AMQP and Nova**. 2020. <https://docs.openstack.org/nova/ussuri/reference/rpc.html>. Accessed: 2020-10-02.

OPENSTACK. **Firewalls and default ports**. 2020. <https://docs.openstack.org/install-guide/firewalls-default-ports.html>. Accessed: 2020-09-20.

OPENSTACK. **Manage Flavors**. 2020. <https://docs.openstack.org/horizon/stein/admin/manage-flavors.html>. Accessed: 2020-09-20.

OPENSTACK. **Message Queuing**. 2020. <https://docs.openstack.org/security-guide/messaging.html>. Accessed: 2020-09-20.

PANIZZON, G. et al. A Taxonomy of container security on computational clouds: concerns and solutions. **Revista de Informática Teórica e Aplicada**, v. 26, n. 1, p. 47–59, abr. 2019. ISSN 21752745. Disponível em: <https://seer.ufrgs.br/rita/article/view/RITA-VOL26-NR1-47>.

PISKAC, P.; NOVOTNY, J. Using of time characteristics in data flow for traffic classification. In: **AIMS**. [S.l.: s.n.], 2011.

PROJECT, A. C. Apache cloudstack open source cloud computing. 2019. Disponível em: <https://cloudstack.apache.org>.

Risso, F. et al. Lightweight, payload-based traffic classification: An experimental evaluation. In: **2008 IEEE International Conference on Communications**. [S.l.: s.n.], 2008. p. 5869–5875.

ROOZBEH, A. **Toward Next-generation Data Centers: Principles of Software-Defined "Hardware" Infrastructures and Resource Disaggregation**. 99 p. Tese (Doutorado) — KTH Royal Institute of Technology, Stockholm, Sweden, 04 2019.

SANKARI, S.; VARALAKSHMI, P.; DIVYA, B. Network traffic analysis of cloud data centre. In: **2015 International Conference on Computing and Communications Technologies (ICCCT)**. [S.l.: s.n.], 2015. p. 408–413.

SCARFONE, K.; SOUPPAYA, M.; HOFFMAN, P. Guide to security for full virtualization technologies. **NIST Special Publication**, v. 800, p. 125, 2011.

Scharf, M. et al. Network-aware instance scheduling in openstack. In: **2015 24th International Conference on Computer Communication and Networks (ICCCN)**. [S.l.: s.n.], 2015. p. 1–6.

SCIAMMARELLA, T. et al. Analysis of control traffic in a geo-distributed collaborative cloud. In: **2016 5th IEEE International Conference on Cloud Networking (Cloudnet)**. [S.l.: s.n.], 2016. p. 224–229.

SHARMA, D. et al. Hansel: Diagnosing faults in openstack. In: **Proceedings of the 11th ACM Conference on Emerging Networking Experiments and Technologies**. New York, NY, USA: ACM, 2015. (CoNEXT '15), p. 23:1–23:13. ISBN 978-1-4503-3412-9. Disponível em: <http://doi.acm.org/10.1145/2716281.2836108>.

Shete, S.; Dongre, N. Analysis amp; auditing of network traffic in cloud environment. In: **2017 International Conference on Intelligent Computing and Control Systems (ICICCS)**. [S.l.: s.n.], 2017. p. 97–100.

VENZANO, D.; MICHIARDI, P. A measurement study of data-intensive network traffic patterns in a private cloud. In: **Proceedings of the 2013 IEEE/ACM 6th International Conference on Utility and Cloud Computing**. Washington, DC, USA: IEEE Computer Society, 2013. (UCC '13), p. 476–481. ISBN 978-0-7695-5152-4. Disponível em: <https://doi.org/10.1109/UCC.2013.93>.

VILELA, G. S. **Caracterização de Tráfego Utilizando Classificação de Fluxos de Comunicação**. Dissertação (Mestre em Ciências em Engenharia de Sistemas e Computação) — Universidade Federal do Rio de Janeiro, Rio de Janeiro, Brasil, março 2006.

WANG, G.; NG, T. S. E. The impact of virtualization on network performance of amazon ec2 data center. In: **Proceedings of the 29th Conference on Information Communications**. Piscataway, NJ, USA: IEEE Press, 2010. (INFOCOM'10), p. 1163–1171. ISBN 978-1-4244-5836-3. Disponível em: <http://dl.acm.org/citation.cfm?id=1833515.1833691>.

WILLIAMSON, C. Internet traffic measurement. **IEEE Internet Computing**, v. 5, n. 6, p. 70–74, Nov 2001. ISSN 1089-7801.

ANEXO A – TABELA COM TODOS OS SERVIÇOS ANALISADOS.

Tabela 11 – Volume de Tráfego (MB) / Serviço (Média ± *sd*).

Operação	Imagem	Ceilometer	Cinder	Designate	Glance	Heat	Keystone	Magnum	Manila	MISC	Neutron	Nova	RabbitMQ	Sahara	Trove
CREATE	Ubuntu Bionic Beaver	0,022 ± 0,004	0,008 ± 0,007	0,105 ± 0,012	346,61 ± 0,01	0,073 ± 0,087	0,14 ± 0,016	0,012 ± 0,019	0,007 ± 0,007	9,552 ± 1,605	0,345 ± 0,159	1,11 ± 0,194	0,249 ± NA	0 ± 0,001	0,011 ± 0,017
	Centos 7 (1300 MB)	0,022 ± 0,001	0,018 ± 0,015	0,195 ± 0,016	1398,295 ± 0,093	0,19 ± 0,186	0,195 ± 0,027	0,166 ± 0,345	0,006 ± 0,005	18,041 ± 0,919	0,902 ± 0,36	1,739 ± 0,449	0,257 ± NA	0,002 ± 0,002	0,158 ± 0,351
	Centos 7 (898 MB)		0,01 ± 0,006	0,142 ± 0,008	860,562 ± 0,123	0,223 ± 0,298	0,163 ± 0,027	0,245 ± 0,474	0,006 ± 0,007	13,229 ± 1,015	0,55 ± 0,293	1,362 ± 0,232	NA	0,001 ± 0,001	0,049 ± 0,055
	Cirros		0,003 ± 0,004	0,061 ± 0,004	15,784 ± 0,004	0,084 ± 0,142	0,145 ± 0,026	0,033 ± 0,049	0,001 ± 0,002	7,938 ± 0,404	0,434 ± 0,276	1,129 ± 0,04		0,001 ± 0,002	0,015 ± 0,031
	Debian 10		0,01 ± 0,007	0,137 ± 0,018	575,644 ± 0,005	0,362 ± 0,095	0,146 ± 0,023	0,033 ± 0,047	0,004 ± 0,004	14,482 ± 0,537	0,481 ± 0,164	1,366 ± 0,357	0,305 ± NA	0,002 ± 0,002	0,067 ± 0,061
	Fedora 31	0,021 ± 0	0,007 ± 0,006	0,116 ± 0,048	356,159 ± 0,017	0,1 ± 0,124	0,137 ± 0,027	0,032 ± 0,035	0,065 ± 0,146	9,547 ± 1,734	0,374 ± 0,236	1,025 ± 0,209	NA	0,001 ± 0,001	0,025 ± 0,034
	Fedora 32	0,021 ± 0,004	0,009 ± 0,009	0,339 ± 0,106	303,511 ± 0,011	0,09 ± 0,126	0,141 ± 0,023	0,028 ± 0,043	0,005 ± 0,007	11,6 ± 0,638	0,362 ± 0,135	0,988 ± 0,236	0,321 ± NA		0,041 ± 0,047
	Ubuntu Focal Fossa	0,022 ± 0,001	0,005 ± 0,005	0,127 ± 0,014	539,857 ± 0,003	0,295 ± 0,049	0,159 ± 0,033	0,018 ± 0,023	0,004 ± 0,004	13,408 ± 1,038	0,447 ± 0,177	1,094 ± 0,185	NA	0,001 ± 0,002	0,081 ± 0,066
	FreeBSD 12	0,023 ± 0,005	0,008 ± 0,005	0,092 ± 0,01	476,316 ± 0,003	0,06 ± 0,08	0,151 ± 0,037	0,019 ± 0,03	0,003 ± 0,003	8,58 ± 1,673	1,051 ± 0,155	0,828 ± 0,15	0,292 ± NA		0,014 ± 0,025
	MS Windows Server	0,022 ± 0,001	0,024 ± 0,006	0,452 ± 0,115	6615,876 ± 1,037	0,233 ± 0,178	0,243 ± 0,033	0,057 ± 0,041	0,012 ± 0,005	25,98 ± 1,927	1,16 ± 0,606	1,429 ± 1,006		0,004 ± 0,006	0,08 ± 0,132
SUSPEND	Ubuntu Bionic Beaver	0,009 ± 0	0,001 ± 0,002	0,024 ± 0,009	0,003 ± 0,002	0,025 ± 0,055	0 ± 0,001	0,009 ± 0,018	0,003 ± 0,002	1,775 ± 0,556	0,035 ± 0,042	0,156 ± 0,034	NA	0 ± 0	0,009 ± 0,023
	Centos 7 (1300 MB)		0,005 ± 0,006	0,033 ± 0,004		0,04 ± 0,092	0,012 ± 0,02	0,005 ± 0,005	0,004 ± 0,004	2,808 ± 0,327	0,127 ± 0,111	0,185 ± 0,119		0,001 ± 0,002	0,013 ± 0,034
	Centos 7 (898 MB)	0,011 ± 0,007	0,002 ± 0,003	0,026 ± 0,006		0,008 ± 0,005	0,004 ± 0,007	0,08 ± 0,077	0,002 ± 0,003	2,436 ± 0,345	0,059 ± 0,028	0,153 ± 0,021		0 ± 0	0,003 ± 0,003
	Cirros	0,009 ± 0	0,003 ± 0,004	0,014 ± 0,003	0,001 ± 0,001	0,009 ± 0,009	0,012 ± 0,017	0,002 ± 0,001	0,005 ± 0,003	1,599 ± 0,288	0,03 ± 0,008	0,181 ± 0,07		NA	0,007 ± 0,004
	Debian 10	0,009 ± 0,001	0,003 ± 0,005	0,021 ± 0,002	0,002 ± 0,002	0,108 ± 0,106	0,002 ± NA	0,008 ± 0,007	0 ± 0	2,237 ± 0,38	0,076 ± 0,126	0,104 ± 0,045		0 ± 0	0,001 ± 0,001
	Fedora 31	0,009 ± 0	0,001 ± 0,003		0,001 ± 0,001	0,03 ± 0,053	0,012 ± 0	0,012 ± 0,021	0,013 ± 0,037	1,694 ± 0,579	0,038 ± 0,041	0,145 ± 0,034			0,002 ± 0,002
	Fedora 32		0,006 ± 0,007	0,108 ± 0,033	0,004 ± 0,005	0,022 ± 0,058	0,006 ± 0,01	0,016 ± 0,025	0,001 ± 0,003	2,937 ± 0,277	0,071 ± 0,085	0,116 ± 0,074		0,001 ± 0,001	0,016 ± 0,031
	Ubuntu Focal Fossa	0,011 ± 0,007	0,003 ± 0,004	0,044 ± 0,013	0,003 ± 0,003	0,153 ± 0,164	0,008 ± 0,014	0,036 ± 0,053	0,003 ± 0,003	3,815 ± 0,513	0,097 ± 0,119	0,155 ± 0,09		0,002 ± 0,002	0,006 ± 0,005
	FreeBSD 12	0,009 ± 0	0,002 ± 0,004	0,013 ± 0,004	0 ± 0	0,006 ± 0,005	0 ± 0	0,002 ± 0,002	0,002 ± 0,002	1,054 ± 0,425	0,174 ± 0,053	0,072 ± 0,025		0 ± 0	0,01 ± 0,021
	MS Windows Server			0,031 ± 0,031	0,003 ± 0,005	0,019 ± 0,05	0,001 ± 0,003	0,016 ± 0,025	0,002 ± 0,003	1,03 ± 0,422	0,085 ± 0,05	0,064 ± 0,015			0,004 ± 0,008
RESUME	Ubuntu Bionic Beaver	0,009 ± 0	0,001 ± 0,004	0,014 ± 0,005	0,001 ± 0,002	0,036 ± 0,065	0 ± 0	0,011 ± 0,021		1,46 ± 0,395	0,041 ± 0,057	0,2 ± 0,041		0,001 ± 0,002	0,01 ± 0,023
	Centos 7 (1300 MB)		0,009 ± 0,01	0,017 ± 0,006		0,088 ± 0,122	NA	0,044 ± 0,079	0 ± 0	1,626 ± 0,22	0,056 ± 0,021	0,207 ± 0,055		0,004 ± 0	0,097 ± 0,125
	Centos 7 (898 MB)	0,009 ± 0,001	0,003 ± 0,005	0,015 ± 0,004	0 ± 0	0,014 ± 0,022	0,001 ± NA	0,093 ± 0,106	0,002 ± 0,003	1,624 ± 0,238	0,093 ± 0,108	0,241 ± 0,11		0 ± 0	0,004 ± 0,005
	Cirros		0 ± 0		0,001 ± 0,001	0,009 ± 0,012	NA	0,032 ± 0,046	0,001 ± 0,002	1,733 ± 0,189	0,032 ± 0,017	0,194 ± 0,019		0 ± NA	0,004 ± 0,003
	Debian 10	0,009 ± 0	0,004 ± 0,004	0,016 ± 0,005	0,005 ± 0,009	0,093 ± 0,003		0,039 ± 0,06	0,006 ± 0,002	1,671 ± 0,147	0,052 ± 0,025	0,128 ± 0,046		0,001 ± 0,002	0,008 ± 0,006
	Fedora 31		0,003 ± 0,004	0,013 ± 0,003	0 ± 0	0,023 ± 0,027	0,006 ± 0,007	0,011 ± 0,02	0,002 ± 0,003	1,383 ± 0,397	0,051 ± 0,047	0,189 ± 0,033		0,001 ± 0,001	0,001 ± 0
	Fedora 32			0,104 ± 0,031	0,002 ± 0,004	0,008 ± 0,026	0,012 ± 0,011	0,012 ± 0,026	0,003 ± 0,003	1,646 ± 0,287	0,033 ± 0,018	0,106 ± 0,037		0,001 ± 0,002	0,004 ± 0,005
	Ubuntu Focal Fossa		0,002 ± 0,003	0,013 ± 0,005	0,002 ± 0,002	0,1 ± 0,013	0,037 ± NA	0,03 ± 0,049	0 ± NA	1,863 ± 0,309	0,035 ± 0,017	0,127 ± 0,087	NA	0 ± NA	0,003 ± 0,003
	FreeBSD 12		0,002 ± 0,004	0,014 ± 0,005	0 ± 0	0,008 ± 0,007	0,02 ± 0,033	0,004 ± 0,003	0,001 ± 0,002	1,345 ± 0,505	0,205 ± 0,05	0,094 ± 0,029		0,001 ± 0,002	0,004 ± 0,006
	MS Windows Server		0,002 ± 0,003	0,033 ± 0,038	0,002 ± 0,004	0,009 ± 0,01	0,013 ± 0,021	0,013 ± 0,024	0 ± 0	1,545 ± 0,345	0,123 ± 0,067	0,1 ± 0,033		0,002 ± 0,002	0,013 ± 0,016
STOP	Ubuntu Bionic Beaver	0,009 ± 0,001	0,003 ± 0,004	0,039 ± 0,004	0,002 ± 0,002	0,013 ± 0,011	0,016 ± 0,018	0,006 ± 0,014	0,004 ± 0,003	2,823 ± 0,91	0,053 ± 0,02	0,198 ± 0,068		0,001 ± 0,001	0,014 ± 0,026
	Centos 7 (1300 MB)		0,001 ± 0,002	0,016 ± 0,008	0,001 ± 0,001	0,048 ± 0,107	NA	0,021 ± 0,04	0,001 ± 0,002	1,253 ± 0,229	0,046 ± 0,046	0,141 ± 0,059		0 ± 0	0,007 ± 0,01
	Centos 7 (898 MB)	0,009 ± 0	0,004 ± 0,004	0,024 ± 0,003	0,002 ± 0,002	0,008 ± 0,005		0,072 ± 0,107	0,004 ± 0,003	2,195 ± 0,306	0,072 ± 0,129	0,192 ± 0,09			0,002 ± 0,002
	Cirros	0,011 ± 0	0,014 ± 0,005	0,247 ± 0,007	0,02 ± 0,003	0,16 ± 0,144	0,013 ± 0,001	0,085 ± 0,044	0,007 ± 0,004	20,769 ± 0,301	0,451 ± 0,166	0,795 ± 0,105		0,004 ± 0,001	0,08 ± 0,05
	Debian 10	0,011 ± 0,007	0 ± 0	0,028 ± 0,022	0,002 ± 0,002	0,067 ± 0,02	0,009 ± 0,014	0,002 ± 0,001	0,003 ± 0,004	2,086 ± 0,49	0,027 ± 0,012	0,1 ± 0,083		0 ± 0	0,025 ± 0,048
	Fedora 31	0,01 ± 0,004	0,001 ± 0,004	0,021 ± 0,005	0,001 ± 0,001	0,042 ± 0,068	0,009 ± 0,009	0,009 ± 0,02		1,607 ± 0,548	0,039 ± 0,043	0,138 ± 0,047		0,001 ± 0,001	0,015 ± 0,027
	Fedora 32	0,009 ± 0	0,004 ± 0,005	0,085 ± 0,027	0,002 ± 0,002	0,038 ± 0,097	0,007 ± 0,007	0,018 ± 0,034	0,001 ± 0,002	2,161 ± 0,367	0,032 ± 0,012	0,093 ± 0,051		0,002 ± 0,002	0,015 ± 0,034
	Ubuntu Focal Fossa	0,011 ± 0,007	0,004 ± 0,004	0,046 ± 0,015	0,001 ± 0,002	0,074 ± 0,035	0,009 ± 0,014	0,008 ± 0,006		3,94 ± 0,574	0,124 ± 0,144	0,098 ± 0,036		0 ± 0	0,013 ± 0,024
	FreeBSD 12	0,013 ± 0,007	0,014 ± 0,005	0,258 ± 0,009	0,004 ± 0,004	0,1 ± 0,099	0,018 ± 0,023	0,029 ± 0,034	0,007 ± 0,003	15,12 ± 0,663	1,939 ± 0,182	0,482 ± 0,205		0,001 ± 0,001	0,051 ± 0,04
	MS Windows Server	0,015 ± 0,009	0,016 ± 0,006	0,293 ± 0,067	0,018 ± 0,009	0,162 ± 0,126	0,024 ± 0,018	0,034 ± 0,035	0,009 ± 0,004	16,308 ± 0,771	0,664 ± 0,426	0,486 ± 0,123			0,054 ± 0,096
SHELVE	Ubuntu Bionic Beaver	0,033 ± 0,002	0,01 ± 0,006	0,151 ± 0,011	1113,557 ± 0,088	0,105 ± 0,099	0,072 ± 0,02	0,024 ± 0,032	0,01 ± 0,012	11,663 ± 1,583	0,307 ± 0,141	0,824 ± 0,114			0,025 ± 0,031
	Centos 7 (1300 MB)	0,035 ± 0,001	0,029 ± 0,028	0,233 ± 0,014	1401,211 ± 0,56	0,26 ± 0,171	0,067 ± 0,01	0,118 ± 0,105	0,008 ± 0,004	20,03 ± 0,874	0,79 ± 0,334	1,423 ± 0,537		0,002 ± 0,002	0,098 ± 0,151
	Centos 7 (898 MB)	0,034 ± 0,001	0,013 ± 0,013	0,169 ± 0,021	919,581 ± 0,527	0,179 ± 0,134	0,089 ± 0,014	0,077 ± 0,075	0,006 ± 0,003	14,844 ± 0,441	0,462 ± 0,242	0,898 ± 0,133			0,049 ± 0,055
	Cirros	0,033 ± 0,001	0,004 ± 0,005	0,026 ± 0,004	28,721 ± 0,027	0,138 ± 0,14	0,056 ± 0,008	0,003 ± 0,001	0,001 ± 0,002	4,087 ± 0,345	0,118 ± 0,082	0,587 ± 0,052			0,007 ± 0,008
	Debian 10	0,033 ± 0,002	0,018 ± 0,005	0,262 ± 0,021	1512,123 ± 0,106	0,61 ± 0,139	0,062 ± 0,011	0,093 ± 0,052	0,009 ± 0,004	24,048 ± 0,655	0,728 ± 0,214	0,971 ± 0,221		0,003 ± 0,002	0,09 ± 0,054
	Fedora 31	0,033 ± 0,001	0,01 ± 0,008	0,148 ± 0,06	979,696 ± 0,071	0,116 ± 0,113	0,064 ± 0,011	0,014 ± 0,023	0,028 ± 0,057	9,809 ± 1,647	0,269 ± 0,143	0,654 ± 0,1	0,273 ± NA	0,001 ± 0,002	0,014 ± 0,023
	Fedora 32	0,032 ± 0,006	0,015 ± 0,013	0,383 ± 0,145	836,53 ± 0,118	0,174 ± 0,161	0,07 ± 0,019	0,066 ± 0,053	0,006 ± 0,004	14,959 ± 0,759	0,339 ± 0,159	0,704 ± 0,285	0,264 ± NA	0,002 ± 0,002	0,046 ± 0,049
	Ubuntu Focal Fossa	0,033 ± 0,002	0,019 ± 0,008	0,263 ± 0,012	1365,056 ± 0,087	0,428 ± 0,126	0,081 ± 0,018	0,086 ± 0,061	0,007 ± 0,004	23,964 ± 0,848	0,697 ± 0,108	0,799 ± 0,25			0,13 ± 0,056
	FreeBSD 12	0,034 ± 0,001	0,003 ± 0,004	0,062 ± 0,005	476,908 ± 0,084	0,068 ± 0,093	0,058 ± 0,012	0,018 ± 0,03	0,002 ± 0,003	5,598 ± 1,214	0,669 ± 0,124	0,43 ± 0,122	NA	0,001 ± 0,001	0,007 ± 0,015
	MS Windows Server	0,035 ± 0,001	0,032 ± 0,004	0,659 ± 0,167	6406,869 ± 1208,735	0,319 ± 0,199	0,064 ± 0,016	0,09 ± 0,035	0,018 ± 0,004	37,046 ± 1,885	1,54 ± 0,951	222,094 ± 1209,865		0,004 ± 0,001	0,116 ± 0,197

Fonte: O próprio autor.