UMA ANÁLISE DO TRÁFEGO DE REDE EM CENÁRIOS DE COMPUTAÇÃO EM NUVEM GEODISTRIBUÍDOS

Tatiana Sciammarella

Projeto de Graduação apresentado ao Curso de Engenharia Eletrônica e de Computação da Escola Politécnica, Universidade Federal do Rio de Janeiro, como parte dos requisitos necessários à obtenção do título de Engenheiro.

Orientadores:
Miguel Elias Mitre Campista
Luís Henrique M. K. Costa

Rio de Janeiro
Setembro de 2015

UMA ANÁLISE DO TRÁFEGO DE REDE EM CENÁRIOS DE COMPUTAÇÃO EM NUVEM GEODISTRIBUÍDOS

Tatiana Sciammarella

PROJETO DE GRADUAÇÃO SUBMETIDO AO CORPO DOCENTE DO CURSO DE ENGENHARIA ELETRÔNICA E DE COMPUTAÇÃO DA ESCOLA POLITÉCNICA DA UNIVERSIDADE FEDERAL DO RIO DE JANEIRO COMO PARTE DOS REQUISITOS NECESSÁRIOS PARA A OBTENÇÃO DO GRAU DE ENGENHEIRO ELETRÔNICO E DE COMPUTAÇÃO

Autor:

Tatiana Sciammarella

Orientador:

Prof. Miguel Elias Mitre Campista, D.Sc.

Co-Orientador:

Prof. Luís Henrique Maciel Kosmalski Costa, Dr.

Examinador:

Prof. Pedro Braconnot Velloso, Dr.

Examinador:

Prof. Rodrigo de Souza Couto, D.Sc.

Rio de Janeiro

Setembro de 2015

UNIVERSIDADE FEDERAL DO RIO DE JANEIRO
Escola Politécnica - Departamento de Eletrônica e de Computação
Centro de Tecnologia, bloco H, sala H-217, Cidade Universitária
Rio de Janeiro - RJ CEP 21949-900

Este exemplar é de propriedade da Universidade Federal do Rio de Janeiro, que poderá incluí-lo em base de dados, armazenar em computador, microfilmar ou adotar qualquer forma de arquivamento.

É permitida a menção, reprodução parcial ou integral e a transmissão entre bibliotecas deste trabalho, sem modificação de seu texto, em qualquer meio que esteja ou venha a ser fixado, para pesquisa acadêmica, comentários e citações, desde que sem finalidade comercial e que seja feita a referência bibliográfica completa.

Os conceitos expressos neste trabalho são de responsabilidade do(s) autor(es).

DEDICATÓRIA

A Deus.

AGRADECIMENTO

Primeiramente, agradeço a toda minha família por tanto carinho. Em especial, aos meus pais, Elyane e Carmino, e meu irmão, Felipe, que me suportaram nos momentos mais difíceis. À minha tia, Diana, pelas palavras de conforto e orações. Aos meus padrinhos, Vinicius e Fabiana, que sempre estiveram presentes.

Aos Profs. Miguel e Luís Henrique cuja orientação e incentivo foram fundamentais para a conclusão deste trabalho. Ao Prof. Rubi pela contribuição no início da pesquisa. Ao Rodrigo pela paciência e supervisão. Aos colegas do Grupo de Teleinformática e Automação pela excelente companhia por tantos anos.

A todos os meus amigos, especialmente às minhas amigas da vida toda, Lunna e Dandara, e às que eu conheci na UFRJ, Luiza e Thais, pelos grandes e pequenos momentos. Muitos ainda estão por vir.

Por fim, agradeço a todos que contribuíram direta ou indiretamente para a minha formação. Este trabalho é o resultado de todo o investimento e confiança em mim depositados.

RESUMO

O investimento em serviços de computação em nuvem vem crescendo ano a ano. Diversas soluções estão disponíveis no mercado para criação de nuvens públicas e privadas. Uma dessas soluções, o OpenStack, é utilizada neste trabalho para gerenciar os recursos virtualizados de uma nuvem geodistribuída. Essa arquitetura geodistribuída foi proposta pelo GT-PID (Grupo de Trabalho - Plataforma IaaS Distribuída) vinculado à RNP (Rede Nacional de Ensino e Pesquisa) para integrar de forma colaborativa os recursos computacionais de universidades e centros de pesquisa. Nessa nuvem, o nó controlador e os servidores de máquinas virtuais (VMs - *Virtual Machines*) estariam espalhados geograficamente e se comunicariam através de uma rede de longa distância, que apresenta, tipicamente, características como latência e banda passante piores que as redes locais. O objetivo deste projeto é avaliar o tráfego estabelecido entre o nó controlador da nuvem e os servidores, para avaliar a escalabilidade e viabilidade dessa infraestrutura em relação às limitações de rede. Os resultados mostram que a cada Servidor de VMs e Discos adicionado ao sistema a taxa de transmissão média aumenta 15 kb/s, enquanto que cada VM em repouso contribui com 0,8 kb/s. Esses dados mostram que, para uma versão de produção com centenas de servidores e milhares de VMs, o tráfego contínuo entre o nó controlador e os servidores pode atingir níveis consideráveis para uma rede WAN. Esse fato é agravado pelos picos gerados durante atividades como a criação de máquinas virtuais. Portanto, enquanto os resultados mostram que a arquitetura é escalável, mostram também por outro lado que parâmetros como a carga de servidores e os casos de uso do sistema devem ser levados em consideração no planejamento da plataforma de computação em nuvem geodistribuída.

Palavras-Chave: computação em nuvem, IaaS, escalabilidade, orquestrador, OpenStack.

ABSTRACT

Investiments in cloud computing are growing every year. Many solutions are available in the market for creation of both public and private clouds. One such solution, OpenStack, is used in this work to manage the virtualized resources of a geo-distributed cloud. This geo-distributed architecture was proposed by GT-PID (Grupo de Trabalho - Plataforma IaaS Distribuída) linked to the RNP (Rede Nacional de Ensino e Pesquisa) to integrate computational resources from universities and research centers in a collaborative way. In this cloud, the controller node and virtual machine (VM) servers would be spread geographically and communicate through a wide area network (WAN), that typically displays worse latency and bandwidth than local area networks (LAN). The objective of this work is to analyze the traffic between the controller node and VM servers to evaluate the scalability and viability of this architecture with the network limitations in sight. Results show that for each Disk and VM server, overall traffic increases by 15 kb/s, while for each idle VM it increases by 0,8 kb/s. These results show that in a production environment with hundreds of servers and thousands of VMs, the continuous traffic between the controller node and servers can achieve considerable levels for a WAN. This scenario gets worse due to traffic peaks generated during actions such as creation of a virtual machine. Therefore, while results show that the architecture is indeed scalable, they also show that parameters such as server load and system use cases should be taken into consideration while planning the geo-distributed cloud computing platform.

Key-words: cloud computing, IaaS, scalability, orchestrator, OpenStack.

SIGLAS

AMQP - *Advanced Message Queuing Protocol*

API - *Application Programming Interface*

BPaaS - *Business Process as a Service*

GT-PID - Grupo de Trabalho - Plataforma IaaS Distribuída

HTTP - *Hypertext Transfer Protocol*

HTTPS - *Hypertext Transfer Protocol Secure*

IaaS - *Infrastructure as a Service*

LAN - *Local Area Network*

NFS - *Network File System*

NIST - *National Institute of Standards and Technology*

PaaS - *Platform as a Service*

REST - *Representational State Transfer*

RNP - Rede Nacional de Ensino e Pesquisa

RPC - *Remote Procedure Call*

SaaS - *Software as a Service*

SLA - *Service Level Agreement*

SQL - *Structured Query Language*

UFRJ - Universidade Federal do Rio de Janeiro

URI - *Uniform Resource Identifier*

VM - *Virtual Machine*

VMRC - *Virtual Machine Remote Control*

VNC - *Virtual Network Computing*

VPN - *Virtual Private Network*

WAN - *Wide Area Network*

Sumário

1 Introdução **1**

1.1 Computação em Nuvem 1

1.2 Serviços da Computação em Nuvem 2

1.2.1 Software as a Service - SaaS 2

1.2.2 Platform as a Service - PaaS 3

1.2.3 Infrastructure as a Service - IaaS 4

1.3 IaaS Geodistribuída 4

1.4 Objetivos do Projeto 6

1.5 Organização do Texto 7

2 O Orquestrador de Nuvem OpenStack **8**

2.1 Serviços 8

2.1.1 OpenStack Dashboard 9

2.1.2 OpenStack Compute 10

2.1.3 OpenStack Image Service 11

2.1.4 OpenStack Block Storage 12

2.1.5 OpenStack Identity 13

2.2 Comunicação dos Serviços do OpenStack 14

2.2.1 Comunicação via filas de mensagens 14

2.2.2 Comunicação via APIs do OpenStack 19

3 Trabalhos Relacionados **22**

3.1 Estudos de Desempenho de Rede 22

3.2 Estudos de Escalabilidade 23

4 Avaliação Experimental **26**

4.1 A Plataforma de Testes 26

4.1.1 Arquitetura Lógica 26

4.1.2 Arquitetura Física 27

4.2 Experimentos 28

4.2.1 Impacto do número de Servidores de VMs e Discos 28

4.2.2 Impacto do número de VMs por Servidor de VMs e Discos . . 31

4.2.3 Impacto da criação de uma VM em um Servidor de VMs e Discos 33

4.2.4 Impacto da criação e exclusão de múltiplas VMs 36

5 Conclusões **40**

Bibliografia **42**

Lista de Figuras

1.1 Modelo hierárquico de serviços da computação em nuvem. 3
1.2 Arquitetura da Plataforma IaaS Geodistribuída proposta pelo GT-PID. . . 5

2.1 Divisão em Projetos do OpenStack. 9
2.2 Componentes do Nova. 10
2.3 Componentes do Glance. 12
2.4 Componentes do Cinder. 13
2.5 Componentes do Keystone. 14
2.6 Troca de mensagens utilizando o RabbitMQ. 16
2.7 Sequência de eventos de uma rpc.cast. 18
2.8 Sequência de eventos de uma rpc.call. 19

3.1 Página HTML de relatório gerada pelo Rally. 25

4.1 Distribuição dos componentes do OpenStack na arquitetura original do GT-PID. 27
4.2 Ambiente de testes utilizado nos experimentos. 28
4.3 Amostra do tráfego de rede entre o Servidor de VMs e Discos e o Controlador. 29
4.4 Distribuição do tráfego entre a comunicação dos componentes do servidor com o RabbitMQ e o MySQL. 30
4.5 Tráfego na rede com o aumento do número de Servidores de VMs e Discos. 31
4.6 Impacto no número de descritores de arquivo com o aumento do número de Servidores de VMs e Discos. 32
4.7 Tráfego na rede com o aumento do número de VMs por Servidor de VMs e Discos. 33
4.8 Impacto no número de descritores de arquivo com o aumento do número de VMs alocadas por servidor de VMs e Discos. 34

4.9 Total de dados transferidos pela rede durante a criação da primeira e segunda instância de uma máquina virtual com inicialização a partir da mesma imagem e sem volume. 35
4.10 Total de dados transferidos pela rede durante a criação da primeira e segunda instância de uma máquina virtual com inicialização a partir da mesma imagem e com volume. 35
4.11 Exemplo de tráfego gerado pela criação e exclusão de 1 máquina virtual. . 37
4.12 Exemplo de tráfego gerado pela criação e exclusão de 10 máquinas virtuais. 38
4.13 Tráfego médio gerado na criação e exclusão de múltiplas instâncias. 38
4.14 Tráfego médio gerado nas criações e deleções concorrentes de instâncias. . 39

Lista de Tabelas

4.1 Configuração das máquinas utilizadas nos experimentos 27

Capítulo 1

Introdução

1.1 Computação em Nuvem

O modelo de computação em nuvem tornou-se possível após o surgimento de novas tecnologias de armazenamento e processamento que contribuíram para a redução do custo de aquisição de recursos computacionais [1]. Neste modelo, o cliente pode acessar um conjunto de recursos como servidores e aplicações disponíveis em centros de dados remotos através da Internet [2]. Entre as principais vantagens desse paradigma destacam-se a flexibilidade de alocação de recursos computacionais e a redução do custo de aquisição e manutenção da infraestrutura. Estudos divulgados pelo Gartner preveem que em 2015 haverá um aumento no gasto mundial com serviços de infraestrutura de nuvem de 32,8% em relação a 2014 [3].

O NIST *(National Institute of Standards and Technology)*, agência não regulatória da administração de tecnologia do Departamento de Comércio dos Estados Unidos, atribui cinco características essenciais à computação em nuvem [2]: alocação sob demanda dos recursos computacionais pelo cliente sem a necessidade de intervenção de terceiros; amplo acesso aos recursos pela rede, inclusive através de dispositivos *thin client* como celulares e tablets; alocação dos recursos entre os diversos clientes de forma transparente, ou seja, o cliente não sabe exatamente onde seus recursos foram alocados, apenas em um nível mais alto de abstração (país, estado ou centro de dados); rápida elasticidade dos recursos, os quais podem ser alocados e liberados conforme a demanda do cliente; e o monitoramento dos recursos pelo provedor e

clientes.

A nuvem também é caracterizada em relação à sua administração e público-alvo, podendo ser privada, comunitária, pública ou híbrida [2]. Uma nuvem privada tem sua infraestrutura operada por uma organização particular, já a nuvem comunitária é compartilhada entre organizações com interesses em comum. Ambas se destinam a atender às demandas de uma comunidade específica de usuários. Em contrapartida, nuvens públicas são administradas por centros acadêmicos, empresas ou organizações governamentais que disponibilizam sua infraestrutura para o público em geral. Por fim, existem as nuvens híbridas, que são formadas da combinação das anteriores.

Atualmente, diversas empresas oferecem soluções proprietárias para a nuvem como a Microsoft [4], a Amazon [5] e o Google [6]. Ao mesmo tempo, estão sendo desenvolvidas soluções abertas para comunidade como o projeto Eucalyptus [7], o CloudStack [8] e o OpenStack [9]. Essas soluções podem ser categorizadas de acordo com o foco do serviço em relação à disponibilização de infraestrutura física, plataforma ou software aos clientes.

1.2 Serviços da Computação em Nuvem

Apesar da grande diversidade de serviços de computação em nuvem oferecidos, é possível classificá-los como parte das seguintes categorias: Software como Serviço (*Software as a Service* - SaaS), Plataforma como Serviço (*Platform as a Service* - PaaS) e Infraestrutura como Serviço (*Infrastructure as a Service* - IaaS). Por exemplo, o Processo de Negócios como Serviço (*Business Process as a Service* - BPaaS) é considerado parte do SaaS [10]. Um estudo da ontologia da nuvem [11] distribui as três categorias citadas em camadas hierárquicas onde uma camada depende dos serviços das camadas inferiores. Esses serviços são representados em uma pilha hierárquica, como visto na Figura 1.1.

1.2.1 Software as a Service - SaaS

A utilização de Software como Serviço permite que usuários finais acessem aplicações na nuvem através da Internet. Desta forma, mesmo usuários com dispositivos *thin*

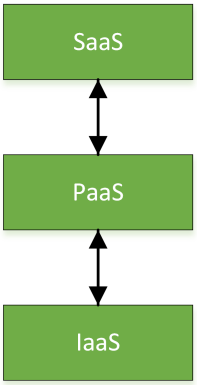

Figura 1.1: Modelo hierárquico de serviços da computação em nuvem.

client podem usufruir do serviço. Além disso, no SaaS, o usuário não administra ou controla a infraestrutura subjacente como rede, servidores, sistemas operacionais e armazenamento, ou mesmo as características da aplicação, exceto determinadas configurações [2]. A utilização desse tipo de serviço traz benefícios adicionais para os clientes, pois dispensa a compra de licenças, preocupações com manutenção e suporte do software e contribui para a redução dos custos da infraestrutura local. Dentre os exemplos de empresas que oferecem soluções de SaaS estão o Google com a suíte Google Apps [12] e a Salesforce [13] que oferece serviços de gestão de relacionamento com o cliente online.

1.2.2 Platform as a Service - PaaS

O modelo de Plataforma como Serviço oferece um ambiente pronto para desenvolvimento e testes de aplicações para a Web. No entanto, os desenvolvedores ficam restritos às linguagens de programação, bibliotecas, serviços e ferramentas suportadas pelo provedor. Diferente do SaaS, entretanto, o cliente controla as aplicações implantadas e, possivelmente, as configurações das aplicações hospedadas na infraestrutura [2]. Alguns exemplos de destaque desse modelo de serviço são a plataforma Microsoft Azure [4] da Microsoft e o GoogleAppEngine [6] do Google.

1.2.3 Infrastructure as a Service - IaaS

Neste modelo de serviço o provedor de infraestrutura disponibiliza recursos computacionais sob demanda para seus clientes. Graças à virtualização é possível criar máquinas virtuais (VMs) isoladas que compartilham recursos como memória e processamento de uma mesma máquina física. Dessa forma, diversos usuários podem alocar simultaneamente VMs personalizadas e acessá-las pela Internet. A oferta da infraestrutura como serviço (IaaS) contribui para a redução de custos de aquisição e manutenção de equipamentos por parte dos usuários. A Amazon se destaca neste mercado com a Amazon Elastic Compute Cloud [5]. Porém, estão disponíveis também diversas soluções de código aberto como o Eucalyptus [7], o CloudStack [8] e o OpenStack [14].

1.3 IaaS Geodistribuída

As implementações tradicionais de IaaS concentram seus recursos computacionais em grandes centros de dados. Em contrapartida, o cenário abordado neste trabalho prevê a criação de uma plataforma IaaS geodistribuída utilizando-se o OpenStack para integrar e orquestrar os recursos computacionais de diferentes universidades e centros de pesquisa. Essa arquitetura, proposta pelo GT-PID (Grupo de Trabalho - Plataforma IaaS Distribuída) [15], vinculado à RNP (Rede Nacional de Ensino e Pesquisa), é ilustrada na Figura 4.1. O GT-PID objetiva aumentar a capacidade global do sistema através da agregação dos recursos de cada instituição assim como melhorar a sobrevivência a falhas da nuvem. Tratando-se de uma arquitetura geodistribuída, em caso de pane ou catástrofes pontuais, o serviço poderia continuar disponível.

Na arquitetura da Figura 4.1 as universidades ou centros de pesquisa são denominados sítios. Os Servidores de Máquinas Virtuais em cada sítio se destinam à hospedagem das VMs dos usuários da infraestrutura. Já os Servidores de Máquinas Virtuais e Discos servem para, além de hospedar VMs, armazenar também os discos virtuais das VMs. Os servidores de um mesmo sítio são interligados por um Comutador local, possibilitando as comunicações de VMs hospedadas em diferentes

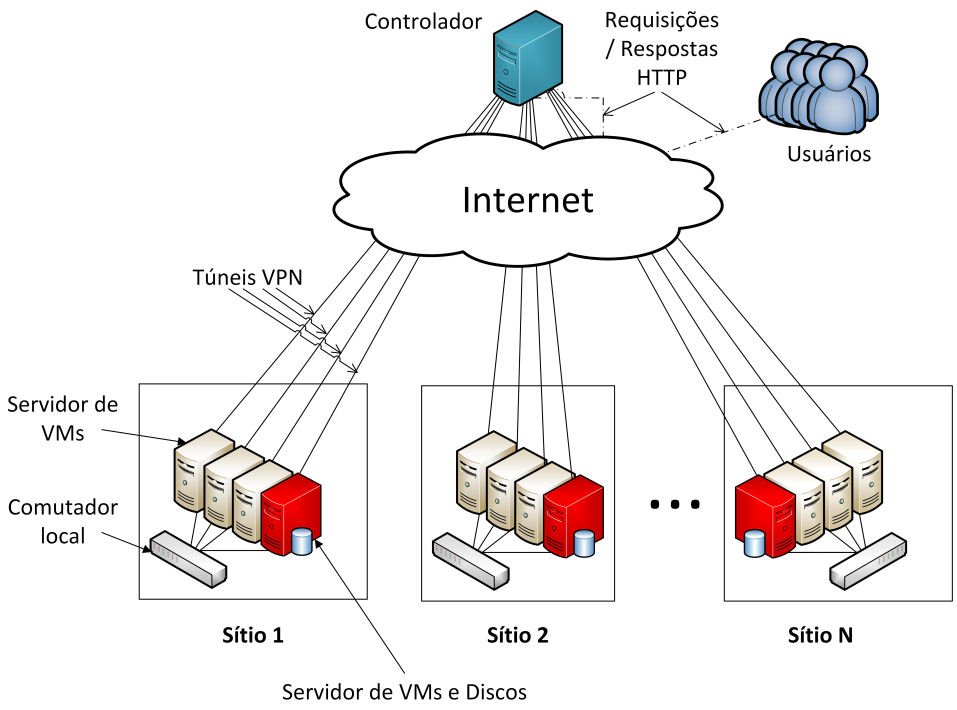

Figura 1.2: Arquitetura da Plataforma IaaS Geodistribuída proposta pelo GT-PID.

servidores e também operações de disco através do NFS (*Network File System*). Além disso, os servidores se comunicam com o nó controlador da nuvem através de túneis VPN (*Virtual Private Network*) estabelecidos pela Internet. O Controlador permite que os usuários acessem a interface web do sistema onde é possível, por exemplo, criar VMs, acessar seus consoles e controlar seus ciclos de vida.

Como pode ser observado na Figura 4.1, a comunicação entre os sítios e o Controlador se dá através de uma rede de longa distância (*Wide Area Network* - WAN). Entre os desafios da implementação da infraestrutura geodistribuída estão as limitações em termos de latência e banda passante de uma WAN em comparação com uma rede local (*Local Area Network* - LAN). Essas limitações podem impactar o desempenho da nuvem visto que a comunicação com o Controlador é crítica para o tempo de resposta das aplicações que executam na nuvem.

1.4 Objetivos do Projeto

Este projeto de fim de curso analisa o impacto da comunicação entre o nó controlador e os nós servidores de máquinas virtuais no tráfego da rede WAN entre esses componentes. O objetivo geral é realizar um estudo para determinar a viabilidade e escalabilidade desta infraestrutura em relação às limitações de rede. Desta forma, tem-se como objetivos específicos: a identificação de elementos que possam representar um gargalo na infraestrutura proposta; e a realização de estudos e testes para a modelagem da troca de mensagens entre o nó controlador e os Servidores de VMs.

Para isso, primeiramente é realizado um estudo do OpenStack. Esse orquestrador foi escolhido para operar a nuvem geodistribuída por ser modular, escalável horizontalmente e apresentar uma grande comunidade de desenvolvimento [9]. Na arquitetura estudada, seus componentes são distribuídos entre o nó controlador e os nós servidores e essa distribuição define o papel de cada máquina do sistema. Após o estudo da comunicação entre os componentes, é realizada uma abordagem experimental utilizando-se ferramentas de análise de tráfego em rede como o tcpdump [16] e Wireshark [17] para avaliar a troca de mensagens entre o nó controlador e os nós servidores em diferentes situações.

A partir dos resultados obtidos foi possível, por exemplo, estimar que cada Servidor de VMs e Discos adicionado à infraestrutura gera um aumento de 15kbps no tráfego contínuo da rede, enquanto que para cada VM instanciada o tráfego de controle aumenta em 0,8kbps. Esses resultados devem ser considerados no planejamento da implementação da infraestrutura e atividades permitidas aos usuários.

1.5 Organização do Texto

Este trabalho está organizado da seguinte forma. Os principais conceitos da plataforma utilizada para orquestração da nuvem são apresentados no Capítulo 2, onde é dada ênfase à arquitetura do OpenStack e a comunicação entre seus componentes. O Capítulo 3 apresenta os trabalhos relacionados. As especificações da plataforma de testes e os experimentos são apresentados no Capítulo 4. Por fim, no Capítulo 5, são apresentadas as considerações finais com base nos experimentos realizados e são feitas propostas de trabalhos futuros.

Capítulo 2

O Orquestrador de Nuvem OpenStack

2.1 Serviços

O OpenStack é um software de código aberto responsável pelo gerenciamento dos recursos computacionais de nuvens públicas e privadas, principalmente para aquelas que oferecem infraestrutura como serviço. Essa plataforma, que originou-se do trabalho da Rackspace (grande provedor de infraestrutura americano) e da NASA (agência espacial americana), atualmente se destaca pela sua grande comunidade desenvolvedora e por sua lista de patrocinadores, que incluem empresas como Intel, IBM e HP [18]. A arquitetura modular do OpenStack, que é subdividida em projetos com serviços especializados, favorece a customização desse software para atuar em diferentes cenários de computação em nuvem. Além disso, essa característica contribui também para escalabilidade horizontal da nuvem, visto que para adicionar novos servidores à infraestrutura, basta replicar os serviços nessas máquinas. Neste trabalho é utilizado o projeto OpenStack Compute para gerenciar o ciclo de vida das VMs, o OpenStack Block Storage para fornecer armazenamento persistente virtual, o OpenStack Image Service para fornecer as imagens de sistema, o OpenStack Identity para realizar o gerenciamento de identidade para usuários e projetos e o OpenStack Dashboard para fornecer a interface web do OpenStack. A seguir, esses projetos e seus serviços são apresentados em maiores detalhes.

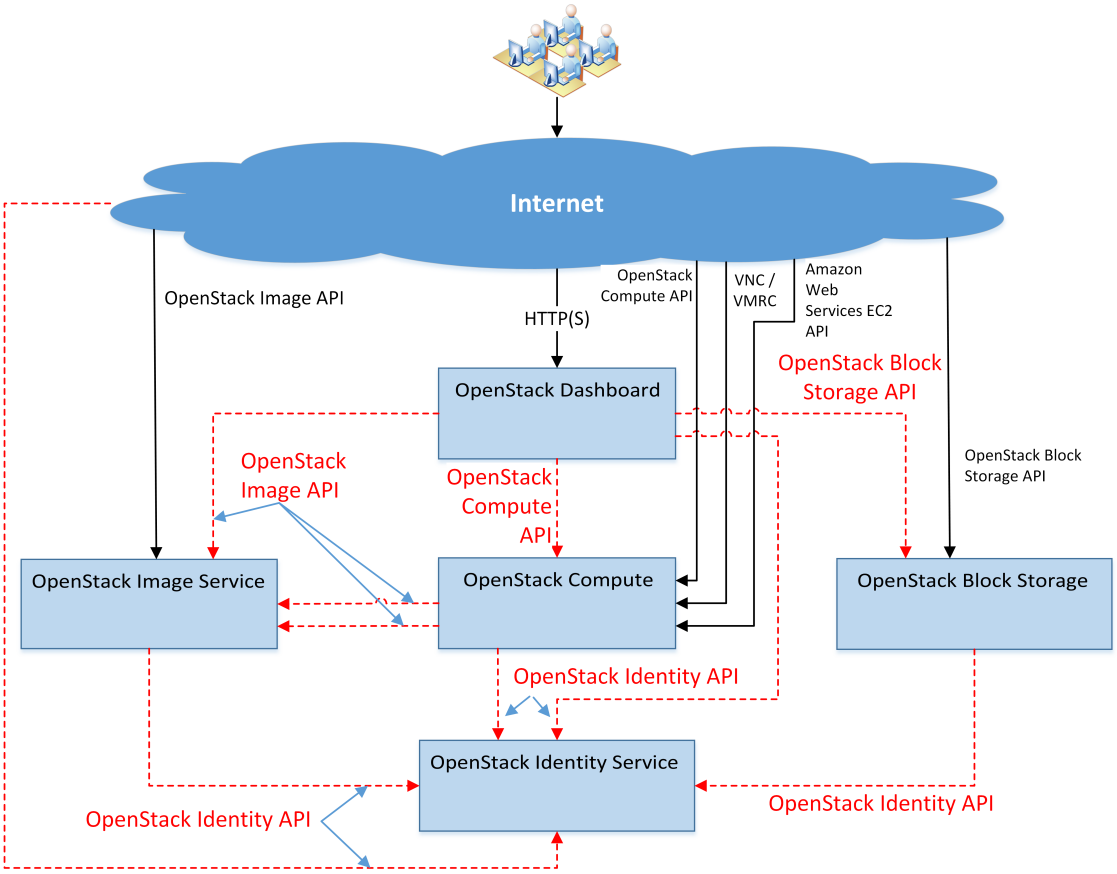

Figura 2.1: Divisão em Projetos do OpenStack.

2.1.1 OpenStack Dashboard

O projeto OpenStack Dashboard (codinome Horizon) fornece a interface gráfica do sistema. Trata-se de um aplicativo web extensível que se comunica com as APIs (*Application Programming Interfaces*) dos serviços do OpenStack e é acessível via conexões HTTP (*Hypertext Transfer Protocol*) ou HTTPS (*Hypertext Transfer Protocol Secure*). Através da interface web os usuários podem criar e gerenciar suas instâncias de máquinas virtuais e o administrador pode executar tarefas como criação e especificação dos limites de alocação de recursos computacionais para os usuários.

Conforme a Figura 2.1, além do acesso via interface web do Horizon, os usuários da nuvem podem acessar os recursos dos projetos diretamente através das APIs nativas do OpenStack ou através da API compatível da Amazon. É possível também acessar a interface gráfica das VMs, presentes nos servidores, remotamente utilizando-se os protocolos VNC (*(Virtual Network Computing*) e VMRC (*Virtual Machine Remote*

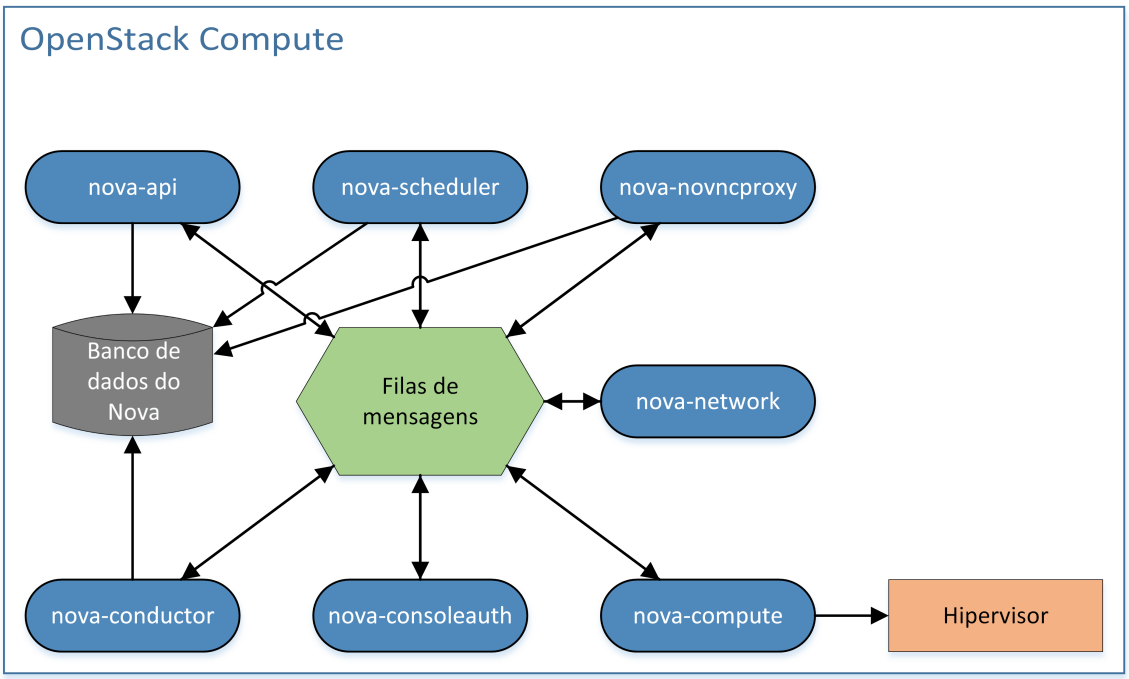

Figura 2.2: Componentes do Nova.

Control).

2.1.2 OpenStack Compute

O projeto OpenStack Compute (codinome Nova) é o elemento principal de uma nuvem OpenStack pois é o responsável pelo provisionamento e gerenciamento de máquinas virtuais. A Figura 2.2 ilustra seus componentes, que se encontram distribuídos entre o nó controlador e os servidores de VMs, desempenhando papéis distintos durante as atividades de criação e manutenção do ciclo de vida das VMs conforme a descrição abaixo.

- **nova-api -** é o componente central do Nova. Ele recebe e responde chamadas do usuário, além de iniciar as atividades de orquestração de VMs. Este componente, portanto, recebe e envia constantemente requisições via HTTP, e utiliza um sistema de filas de mensagens para se comunicar com outros componentes do Nova.

- **nova-compute -** é o elemento responsável pela criação e término das VMs. Ele realiza essas tarefas comunicando-se com o hipervisor, o qual controla os

dispositivos de hardware e fornece a abstração de máquinas virtuais.

- **nova-scheduler -** determina em qual Servidor de VMs (máquina física) uma instância de VM (máquina virtual) é criada.
- **nova-network -** componente que manipula a rede dentro do Nova. Realiza tarefas como configuração de interfaces de redes virtuais e alterações das regras de firewall do *iptables*.
- **nova-novncproxy -** fornece um proxy para acessar as VMs por um console VNC, o qual permite que o usuário acesse a interface gráfica da máquina remotamente.
- **nova-consoleauth -** realiza a autenticação dos usuários ao nova-novncproxy, fornecendo fichas para acessar o proxy.
- **banco de dados do Nova -** banco de dados SQL (*Structured Query Language*) que armazena dados relativos às instâncias.
- **nova-conductor -** é um mediador das interações entre o nova-compute e o banco de dados. Ele aumenta a segurança promovendo o isolamento entre esses dois componentes.
- **filas de mensagens -** elemento que intermedeia a troca de mensagens entre os componentes do Nova. Neste trabalho o sistema de filas é implementado com o RabbitMQ [19].

2.1.3 OpenStack Image Service

O projeto OpenStack Image Service (codinome Glance) oferece um serviço de descoberta, registro, recuperação e armazenamento de imagens para inicialização de novas máquinas virtuais. Uma imagem pode ser um simples sistema operacional ou até mesmo uma cópia do disco de uma máquina virtual existente. O próprio usuário pode submeter suas imagens para o repositório gerenciado pelo Glance. No cenário estudado, todos os componentes do Glance, ilustrados na Figura 2.3 e descritos abaixo, encontram-se no nó controlador da nuvem.

- **glance-api -** recebe chamadas de API para o Glance.

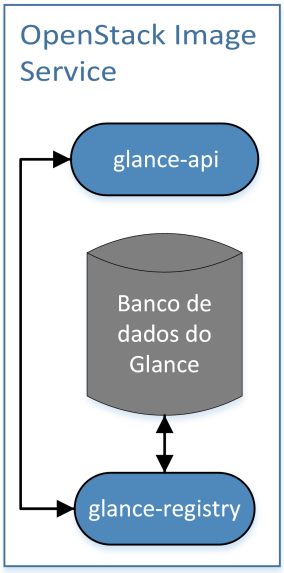

Figura 2.3: Componentes do Glance.

- **glance-registry -** componente que registra, processa e recupera metadados das imagens (tamanho, tipo, etc.).
- **banco de dados do Glance -** armazena metadados das imagens.
- **repositório de armazenamento do Glance -** armazena as imagens.

2.1.4 OpenStack Block Storage

O projeto OpenStack Block Storage (codinome Cinder) fornece um serviço de armazenamento persistente para as máquinas virtuais. Os dados ficam armazenados em unidades denominadas volumes na terminologia do OpenStack. Esses volumes representam discos virtuais e podem ser acessados quando estão ligados às instâncias de VMs. Sendo assim, a instância pode estar ligada a vários volumes, no entanto um volume só pode servir uma instância. Além disso, um volume pode ser uma unidade de inicialização (*boot*) para uma instância quando contém uma imagem de máquina virtual. A Figura 2.4 ilustra a arquitetura lógica deste projeto. Assim como o Nova, o Cinder utiliza um sistema de filas de mensagens para comunicação entre seus componentes e um banco de dados para armazenar informações relativas aos volumes. Os demais componentes são descritos abaixo.

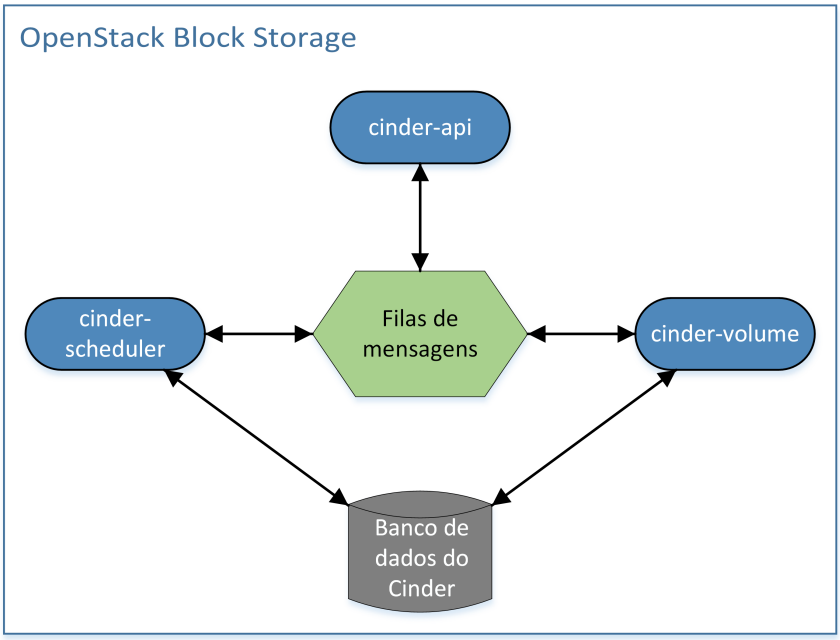

Figura 2.4: Componentes do Cinder.

- **cinder-api -** recebe chamadas de API e as encaminha para o componente cinder-volume de um determinado servidor de VMs e Discos.
- **cinder-volume -** administra os volumes e atualiza o banco de dados do Cinder com o estado dos volumes.
- **cinder-scheduler -** determina em qual Servidor de VMs e Discos o volume da VM é instanciado. Neste trabalho, esse componente sempre cria o volume no Servidor de VMs e Discos do sítio escolhido pelo nova-scheduler.

2.1.5 OpenStack Identity

O projeto OpenStack Identity (codinome Keystone) realiza o gerenciamento de identidades da nuvem. Através de operações de consulta e atualizações de suas tabelas, ele fornece os serviços de verificação e administração de fichas (*tokens*), que são utilizadas nas requisições de autenticação após a verificação das credenciais dos usuários, descoberta de terminais de comunicação para os serviços, autorização e validação de credenciais. Esse projeto é ilustrado na Figura 2.5.

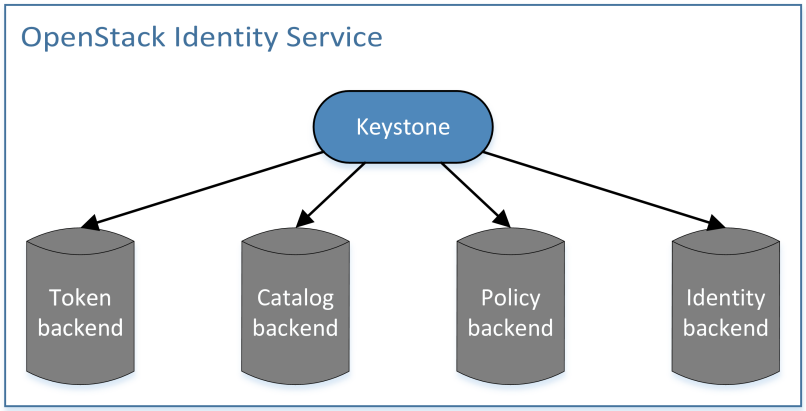

Figura 2.5: Componentes do Keystone.

- **Keystone -** elemento que recebe e responde chamadas de APIs para realizar o gerenciamento de identidades da nuvem.
- **Token backend -** tabela que armazena fichas.
- **Catalog backend -** tabela que contém o registro dos serviços disponíveis e os terminais de comunicação de suas APIs.
- **Policy backend -** tabela que armazena informações relativas às permissões de acesso de usuários e projetos aos serviços do OpenStack.
- **Identity backend -** tabela que armazena as credenciais de usuários e projetos.

2.2 Comunicação dos Serviços do OpenStack

Nesta seção são abordadas duas estratégias de comunicação do OpenStack. Na primeira, utiliza-se um sistema de filas de mensagens para estabelecer a comunicação interna entre os componentes de projetos como o Cinder e o Nova. Na segunda, realizam-se chamadas de APIs para acessar os serviços dos projetos do OpenStack.

2.2.1 Comunicação via filas de mensagens

Projetos como o Cinder e o Nova podem ser escalados horizontalmente, ou seja, seus componentes podem estar presentes em diversos servidores de máquinas vir-

tuais além do nó controlador, de forma a aumentar a capacidade dos serviços oferecidos. Para casos assim, o OpenStack utiliza o protocolo de filas de mensagens AMQP*(Advanced Message Queuing Protocol*) aliado a um *middleware* de mensagens, como o RabbitMQ [19] e o ZeroMQ [20], para intermediar a comunicação entre esses componentes.

O AMQP é um protocolo aberto cujo objetivo é promover a interoperabilidade entre *middlewares* orientados a mensagens definindo não somente o protocolo de rede, como também uma representação para os dados de envelopamento da mensagem e a semântica básica dos *middlewares*. Devido à característica geodistribuída da nuvem estudada, as mensagens encapsuladas nos servidores são enviadas ao *middleware*, presente no Controlador, utilizando-se o protocolo TCP/IP.

Neste projeto utiliza-se o *middleware* RabbitMQ para implementar o modelo *publish/subscribe* de troca de mensagens. Seguindo este modelo, as aplicações que geram mensagens (produtores) enviam as mensagens para o RabbitMQ, que utiliza um sistema de trocas (*exchanges* na terminologia do RabbitMQ) para encaminhá-las para as filas de mensagens apropriadas. Essas mensagens ficam armazenadas até o momento que as aplicações que registraram interesse de receber as mensagens dessas filas (consumidores) estejam prontas para consumi-las.

Conforme a Figura 2.6, no RabbitMQ, cada *exchange* está relacionado a um conjunto de filas de mensagens. Esse relacionamento é chamado de ligação e significa que uma fila quer receber mensagens de um determinado *exchange*. Cada ligação possui o parâmetro `chave de ligação`. Quando uma mensagem é enviada para o RabbitMQ, define-se o *exchange* de destino e uma `chave de roteamento` que indica para qual fila do *exchange* a mensagem deve ser enviada.

A política de roteamento de mensagens de cada *exchange* é determinada pelo seu tipo. Um *direct exchange* encaminha mensagens para filas cuja chave de ligação é idêntica à chave de roteamento da mensagem. Um *fanout exchange* encaminha as mensagens para todas as filas ligadas àquele *exchange* independentemente da chave de roteamento. Já um *topic exchange* permite realizar o roteamento baseado em

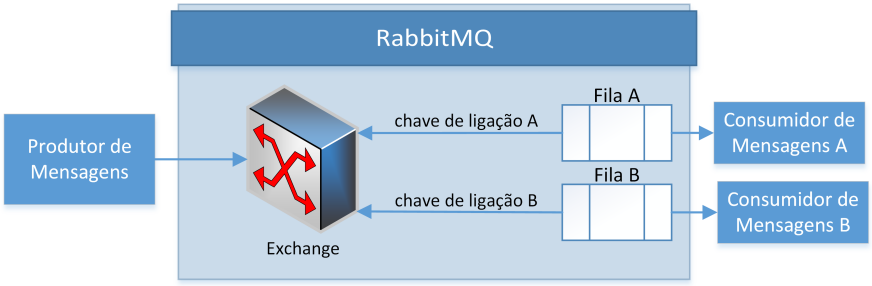

Figura 2.6: Troca de mensagens utilizando o RabbitMQ.

múltiplos critérios. Uma mensagem enviada para esse tipo de *exchange* pode conter uma chave de roteamento formada por uma lista de palavras separadas por pontos como "topic.host". Assim como o *direct exchange*, a mensagem será enviada para uma fila se a chave de ligação combinar com a chave de roteamento. No entanto, existem dois casos especiais. No primeiro, utiliza-se o caractere "*" para substituir uma palavra da chave de ligação como "topic.*". Nesse caso, por exemplo, essa chave de ligação pode ser combinada com a chave de roteamento "topic.host1" ou com "topic.host2". No segundo caso, utiliza-se o caractere "#" para substituir zero ou mais palavras da chave de ligação. Dessa forma, uma chave de ligação do tipo "topic.#" poderá ser combinada, por exemplo, com uma chave de roteamento do tipo "topic", "topic.host1" ou "topic.host2".

Esse sistema de troca de mensagens é utilizado para realizar chamadas remotas de procedimentos. Tais chamadas (*Remote Procedure Call*- RPC) invocam um procedimento em outro espaço de endereçamento da rede. No OpenStack, um produtor pode realizar uma `rpc.call` (chamada remota de procedimento que envia uma mensagem e espera um retorno) ou uma `rpc.cast` (apenas envia uma mensagem). Para realizar esses procedimentos, os componentes do Nova e do Cinder criam instâncias de objetos para enviar mensagens (produtores) e para recebê-las (consumidores). A descrição desses objetos encontra-se abaixo.

- ***Topic Publisher*** **-** esse produtor é instanciado quando ocorre uma rpc.call ou rpc.cast. Esse tipo de objeto se conecta sempre com o mesmo *topic exchange*. O ciclo de vida é limitado pela entrega da mensagem.

- ***Topic Consumer*** **-** esse consumidor é criado quando um novo componente

como o nova-compute é instanciado, e possui duração igual ao ciclo de vida desse mesmo componente. Todo componente tem dois *topic consumers*: um que se conecta com um *topic exchange* via uma fila compartilhada entre outros *topic consumers* quando ocorre um rpc.cast e um segundo que se conecta a uma fila exclusiva quando ocorre um rpc.call.

- ***Direct Publisher -*** esse produtor é instanciado apenas quando ocorre uma rpc.call. Seu propósito é retornar uma mensagem de resposta ao componente que realizou um rpc.call. Esse objeto se conecta com um *direct exchange* para enviar a mensagem.

- ***Direct Consumer -*** esse consumidor é instanciado apenas quando é feita uma rcp.call. Ele se conecta com um único tipo de *direct exchange* via uma fila exclusiva. Seu ciclo de vida é limitado pelo recebimento da mensagem de resposta.

A integração desses componentes durante uma rpc.cast e rcp.call com o RabbitMQ é exemplificada a seguir.

RPC Cast

A Figura 2.7 ilustra a sequência de eventos que ocorrem durante uma rcp.cast para o nova-compute presente no Servidor 1.

1. Envio da mensagem pelo *Topic Publisher*: o nova-api instancia um *Topic Publisher* para enviar uma mensagem para o sistema de filas.

2. Recepção da mensagem pelo *Topic Consumer*: depois que a mensagem passa pelo *exchange* chamado "nova" que é do tipo *topic*, ela chega na fila cuja chave de ligação seja "compute". O *Topic Consumer* que estiver inscrito nessa fila receberá a mensagem e passará para o componente responsável pela tarefa requerida.

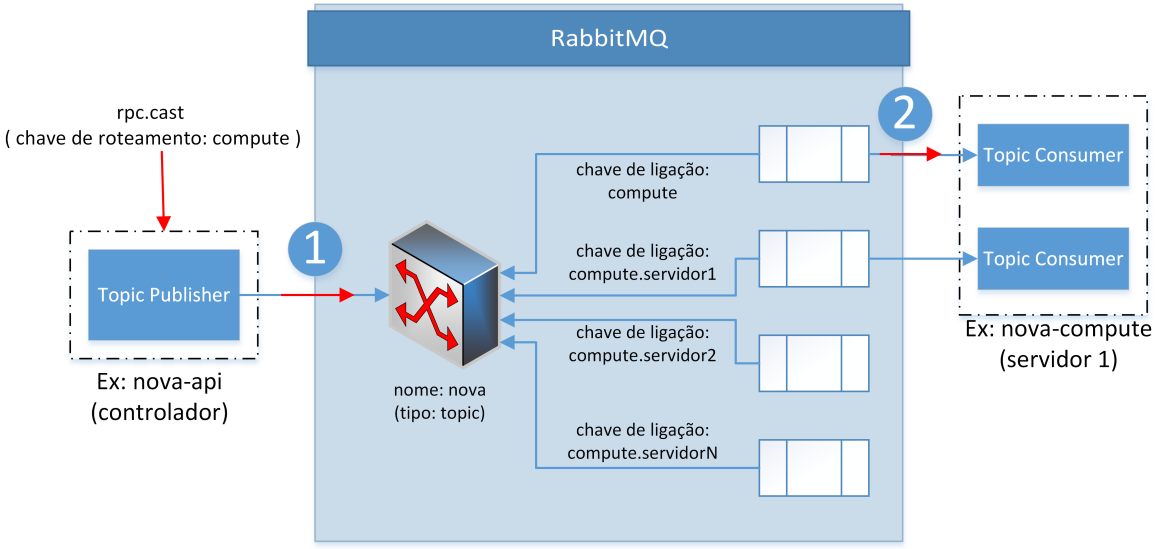

Figura 2.7: Sequência de eventos de uma rpc.cast.

RPC Calls

A Figura 2.8 ilustra a sequência de eventos que ocorrem durante uma rcp.call do nova-api para o nova-compute presente no Servidor 1.

1. Envio da mensagem pelo *Topic Publisher*: o nova-api instancia um *Topic Publisher* para enviar uma mensagem para o RabbitMQ e um *Direct Consumer* para esperar a resposta.

2. Recepção da mensagem pelo *Topic Consumer*: a mensagem passa pelo *exchange* "nova", que é do tipo *topic*, e é encaminhada para a fila cuja chave de ligação seja do tipo "compute.servidor1". O *Topic Consumer* inscrito nessa fila recebe a mensagem e passa a requisição para o nova-compute do Servidor 1.

3. . Envio da mensagem pelo *Direct Publisher*: assim que a ação é cumprida, um *Direct Publisher* é instanciado para enviar a resposta para o sistema de filas.

4. Recepção da mensagem pelo *Direct Consumer*: a mensagem de resposta passa por um *direct exchange* e é encaminhada para a fila cuja chave de ligação corresponde ao identificador da mensagem. A mensagem então chega ao *Direct Consumer* do nova-api.

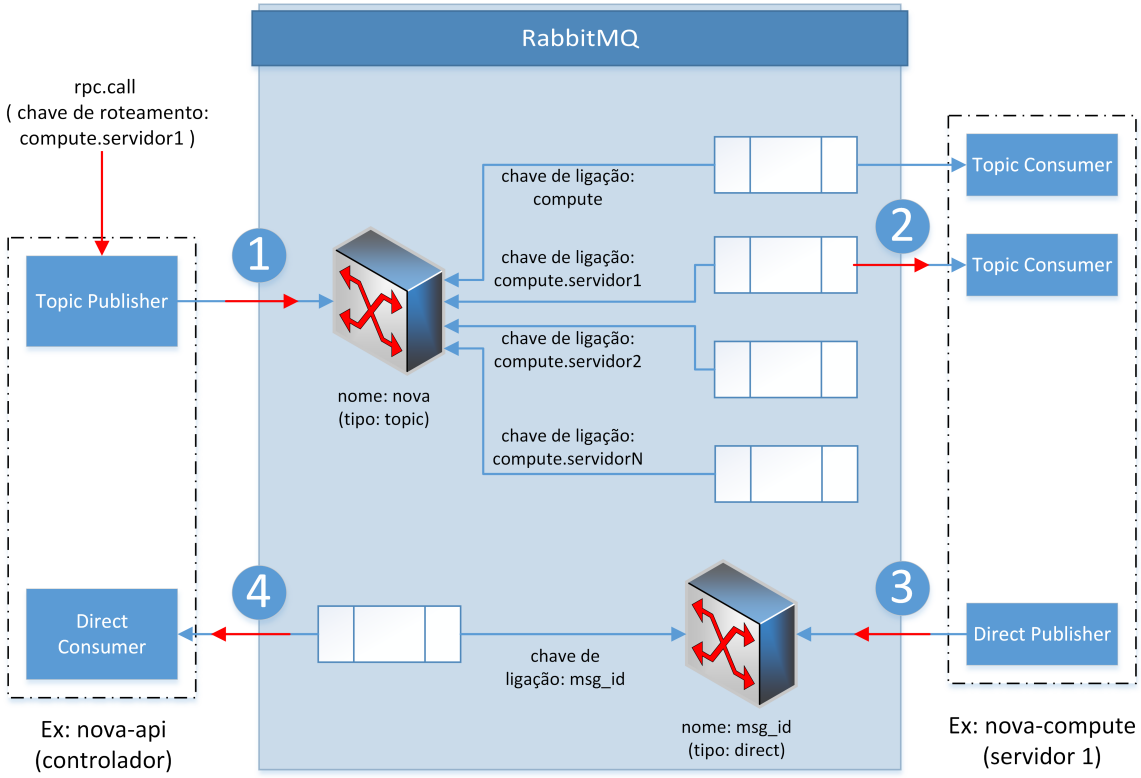

Figura 2.8: Sequência de eventos de uma rpc.call.

Observando-se os exemplos acima percebe-se que a utilização de um *middleware* de mensagens como o RabbitMQ traz benefícios aos sistemas distribuídos como o desacoplamento espacial, pois as mensagens são enviadas para um intermediário e não para qualquer destinatário, ou seja, o emissor não precisa conhecer o receptor, e o desacoplamento temporal, pois as mensagens ficam armazenadas até o momento de seu consumo. Como mencionado anteriormente, esse sistema de filas é utilizado para a comunicação interna dos componentes de um mesmo projeto como os do Nova e do Cinder, os quais costumam estar replicados em diversas máquinas da nuvem. No entanto, para a comunicação entre projetos do OpenStack ou entre um usuário e um projeto são realizadas chamadas de API como abordado a seguir.

2.2.2 Comunicação via APIs do OpenStack

Cada projeto do OpenStack possui uma API que fornece um conjunto padronizado de requisições para que outras aplicações possam acessar seus serviços. Essas APIs são implementadas como serviços web, tornando-as acessíveis através da Internet

como ilustrado na Figura 2.1.

As APIs do OpenStack são do tipo RESTful, ou seja, são baseadas no padrão REST (*Representational State Transfer*) de construção de serviços web. Esse tipo de API utiliza URIs (*Unified Resource Identifiers*) para identificar os recursos do sistema e formatos padrões como XML e JSON para representá-los. Os métodos `GET` , `PUT`, `POST` e `DELETE` do protocolo HTTP são utilizados para operar sobre esses recursos.

No exemplo abaixo, o Horizon faz uma requisição enviando a URI do tipo `/v2/{tenant\_id}/os-security-groups` juntamente como o método `GET` para a API do Nova (presente no Controlador), para receber a lista dos grupos de segurança existentes. O destino da requisição também é passado na URI identificando-se o endereço IP da VPN do Controlador (10.8.0.1), e a porta que a API do Nova escuta (8774).

```
REQ: curl -g -i 'http://10.8.0.1:8774/v2/c76cca8ea94347088404273e8a41d5b7/os-security-groups' -X GET -H "Accept: application/json-H "User-Agent: python-novaclient-H "X-Auth-Project-Id: c76cca8ea94347088404273e8a41d5b7-H "X-Auth-Token: SHA12c8af17db7195c525e494694b6d84739e0290150"
```

A reposta para essa requisição é mostrada abaixo. Primeiramente, o número "200" indica que a requisição foi concluída com sucesso. Depois vem a data, o tamanho e o tipo da resposta, que neste caso está no formato JSON, e um identificador para a requisição. O conteúdo em si, com as informações dos grupos de segurança, encontra-se no "RESP BODY".

```
RESP: [200] CaseInsensitiveDict('date': 'Wed, 12 Aug 2015 16:33:40 GMT', 'content-length': '139', 'content-type': 'application/json', 'x-compute-request-id': 'req-36639060-718f-4a28-9cf4-a11796f633a8') RESP BODY: "security_groups": ["rules": [], "tenant_id": "c76cca8ea94347088404273e8a41d5b7", "description": "default", "id": 1, "name": "default
]
```

O entendimento da arquitetura do OpenStack e suas estratégias de comunicação é importante para elaborar a implementação da nuvem. Além disso, diversas empresas e instituições também investem em pesquisas para avaliar o desempenho e escalabilidade dessa plataforma. Algumas dessas pesquisas são apresentadas no capítulo seguinte.

Capítulo 3

Trabalhos Relacionados

Este capítulo apresenta trabalhos que possuem relação com o projeto desenvolvido. Esses trabalhos realizam estudos de desempenho de rede e escalabilidade do OpenStack.

3.1 Estudos de Desempenho de Rede

Em [21] é realizado um estudo do Quantum, projeto do OpenStack especializado no fornecimento do serviço de rede para as máquinas virtuais e que veio a substituir e expandir as funcionalidades de rede do componente nova-network. Esse projeto é, atualmente, conhecido como Neutron. Nesse trabalho, o desempenho do Quantum é analisado em dois cenários: no primeiro, há apenas uma máquina especializada no fornecimento do serviço de rede, já no segundo esse serviço está distribuído entre os servidores de máquinas virtuais. Esse estudo aponta que no primeiro cenário, em que há apenas uma máquina fornecendo o serviço de rede, o risco de falha do serviço é maior, pois o nó de rede é um ponto único de falha. Além disso, em momentos de tráfego intenso, esse nó pode representar um gargalo no desempenho do sistema. No segundo cenário, entretanto, o tráfego é distribuído entre os servidores e a qualidade do serviço de rede torna-se maior, pois não há um gargalo e nem um ponto único de falha para rede. Nesses dois cenários são conduzidos testes de desempenho utilizando o software D-ITG para estimar o atraso e a perda de pacotes. Os resultados mostram que a utilização de diversos nós de rede é vantajosa em relação a um nó único pois, mesmo com o aumento da quantidade de dados transferidos, o atraso de pacotes

apresenta uma distribuição aproximadamente uniforme.

Em [22] Gebreyohannes também estuda o desempenho da rede entre VMs de uma implementação de nuvem OpenStack com o Neutron. Ele analisa parâmetros como taxa de transferência, perda e atraso de pacotes utilizando a ferramenta iperf [23] para os seguintes cenários: VMs no mesmo servidor e mesma sub-rede, VMs no mesmo servidor e sub-redes diferentes, VMs em servidores diferentes e sub-redes iguais e VMs em servidores e sub-redes diferentes. Os resultados indicam que VMs no mesmo servidor e mesma sub-rede apresentam um melhor desempenho de rede devido às menores rotas de transmissão de pacotes.

Os projetos abordados acima estudam o desempenho da rede entre máquinas virtuais. Neste trabalho, no entanto, é feita uma análise do tráfego entre os servidores e o nó controlador, identificando, através de ferramentas como o tcpdump, a contribuição de cada projeto do OpenStack neste tráfego.

3.2 Estudos de Escalabilidade

Na literatura existem diversos estudos sobre a escalabilidade do OpenStack. Recentemente, esses estudos estão sendo facilitados pela utilização do Rally [24], uma ferramenta de *benchmark* especialmente desenvolvida para a realização de testes de escalabilidade e desempenho em nuvens OpenStack. Essa ferramenta oferece uma série de cenários de testes pré-definidos que ajudam a simular diferentes cargas na nuvem. Os cenários são arquivos no formato JSON ou YAML, que contêm parâmetros configuráveis como `flavor`, para definir a quantidade de recursos de hardware alocados para uma VM, `image`, para definir a imagem da VM, `times`, para definir o número de iterações de uma ação do cenário (como o número de ciclos de criação de VMs), `concurrency`, para definir o nível de paralelização das requisições (concurrency), além de critérios para acordo de nível de serviço (*Service Level Agreement* - SLA) como o parâmetro `max_seconds_per_iteration` para definir o tempo máximo aceitável para execução de uma iteração. O exemplo abaixo é do arquivo boot-and-delete.json. A partir das configurações deste arquivo o Rally envia requisições para a API do Nova para criar e excluir instâncias.

```
{"NovaServers.boot_and_delete_server": [{
    "args": {
        "flavor": {
            "name": "m1.tiny"
        },
        "image": {
            "name": "^cirros.*uec$"
        },
        "force_delete": false
    },
    "runner": {
        "type": "constant",
        "times": 10,
        "concurrency": 2
        },
    "context": {
        "users": {
            "tenants": 3,
            "users_per_tenant": 2
        }
     }
     "sla": {
         "max_seconds_per_iteration": 15
            }
}]}
```

Após a execução dos testes, o Rally gera um relatório que contém informações como a duração total e individual das iterações e a quantidade de iterações mal-sucedidas. Depois de executar o cenário descrito acima, é possível visualizar uma página HTML de relatório como ilustra a Figura 3.1. Essa figura mostra que 100% das iterações foram bem-sucedidas pois todas foram executadas em um intervalo de

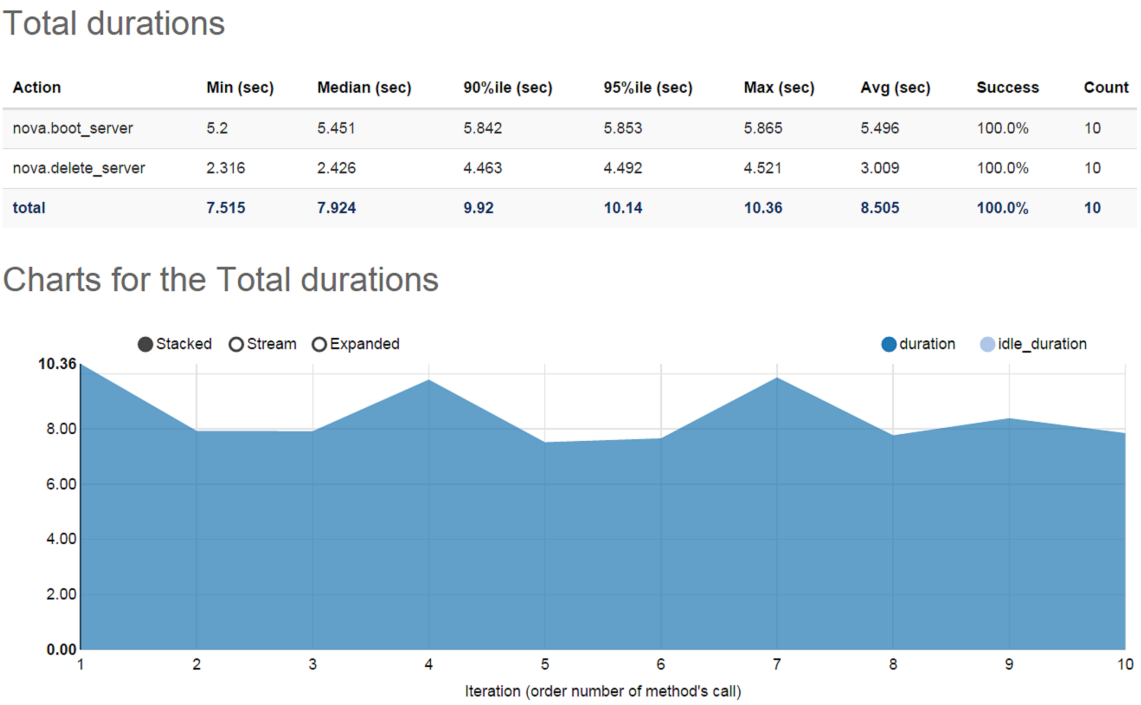

Figura 3.1: Página HTML de relatório gerada pelo Rally.

tempo menor que o definido pelo parâmetro `max_seconds_per_iteration`.

A Mirantis, um dos maiores contribuidores para o código-fonte do OpenStack e responsável pelo gerenciamento de nuvens OpenStack para mais de 100 clientes, em [25] utiliza o Rally para avaliar o desempenho de sua versão do OpenStack no centro de dados da IBM. A Cisco em [26] também realiza diversos experimentos para estressar os componentes do OpenStack e determinar, por exemplo, o tempo que uma instância leva para ser inicializada quando o nova-api recebe muitas requisições e o número máximo de servidores e VMs que o RabbitMQ suporta. Neste trabalho, diferentes cenários de criação e deleção de VMs do Rally são utilizados para gerar carga na rede da plataforma de testes apresentada no próximo capítulo.

Capítulo 4

Avaliação Experimental

Este capítulo apresenta a plataforma de testes utilizada e os experimentos realizados, que têm por objetivo avaliar o impacto na rede do número de Servidores de VMs e Discos ativos, do número de máquinas virtuais em execução por servidor e das atividades de criação e exclusão de máquinas virtuais. Esses resultados são importantes pois permitem dimensionar a infraestrutura de rede conforme o tamanho da nuvem.

4.1 A Plataforma de Testes

Nesta seção são apresentados detalhes do ambiente de testes utilizados como a distribuição dos componentes do OpenStack entre o Controlador e Servidores de VMs e Discos e a arquitetura física da plataforma de testes.

4.1.1 Arquitetura Lógica

Na arquitetura inicialmente proposta para o cenário geodistribuído as máquinas da infraestrutura se dividem entre Controlador, Servidores de VMs e Servidores de VMs e Discos. A distribuição dos componentes do OpenStack entre as máquinas nessa configuração encontra-se na Figura 4.1. Nos experimentos, optou-se por utilizar somente servidores do tipo VMs e Discos. Esses servidores possuem um maior número de componentes se comunicando com o Controlador através do túnel VPN e, portanto, representam o pior caso para o estudo proposto.

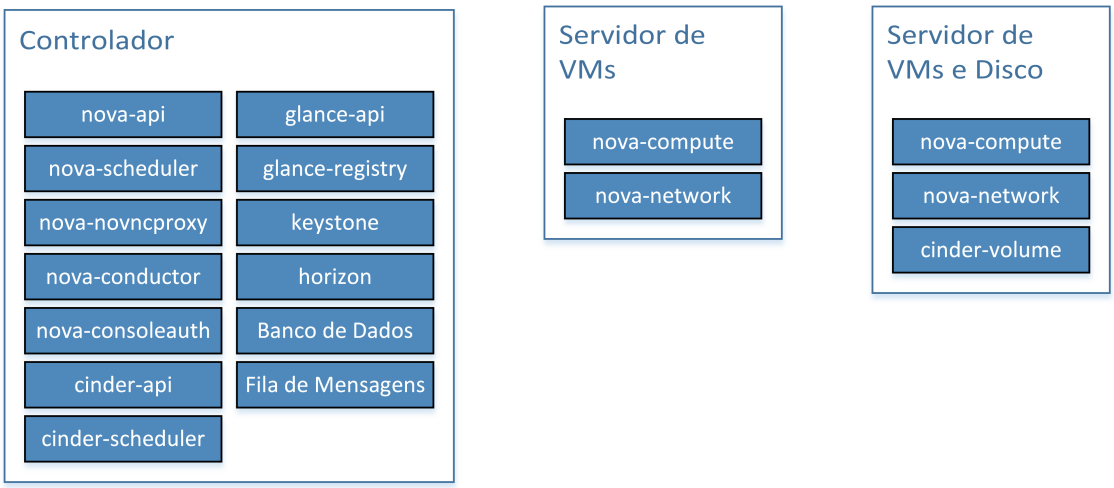

Figura 4.1: Distribuição dos componentes do OpenStack na arquitetura original do GT-PID.

4.1.2 Arquitetura Física

A arquitetura física utilizada nos experimentos é ilustrada na Figura 4.2. Neste cenário cada Servidor de VMs e Discos emula um sítio e está ligado ao Controlador através de um túnel VPN, uma rede privada virtual que utiliza tecnologia de tunelamento e criptografia para manter a segurança dos dados trafegados [27].

A Tabela 4.1 contém a configuração de cada máquina da rede de testes. Apenas no experimento que mapeia o comportamento da rede com o aumento do número de servidores são utilizadas todas as máquinas. Nos demais, é utilizado apenas o Controlador e o Servidor 1.

Tabela 4.1: Configuração das máquinas utilizadas nos experimentos

Máquina	**CPU**	**RAM**
Controlador	Intel(R) Core(TM) i7 CPU 860 @ 2,80GHz	8GB
Servidor 1	Intel(R) Core(TM) i7-4930K CPU @ 3,40GHz	32GB
Servidor 2	Intel(R) Core(TM) i7 CPU 860 @ 2,80GHz	8GB
Servidor 3	Intel(R) Xeon(R) CPU E3-1241 v3 @ 3,50GHz	32GB
Servidor 4	Intel(R) Core(TM)2 Quad CPU Q9400 @ 2,66GHz	6GB

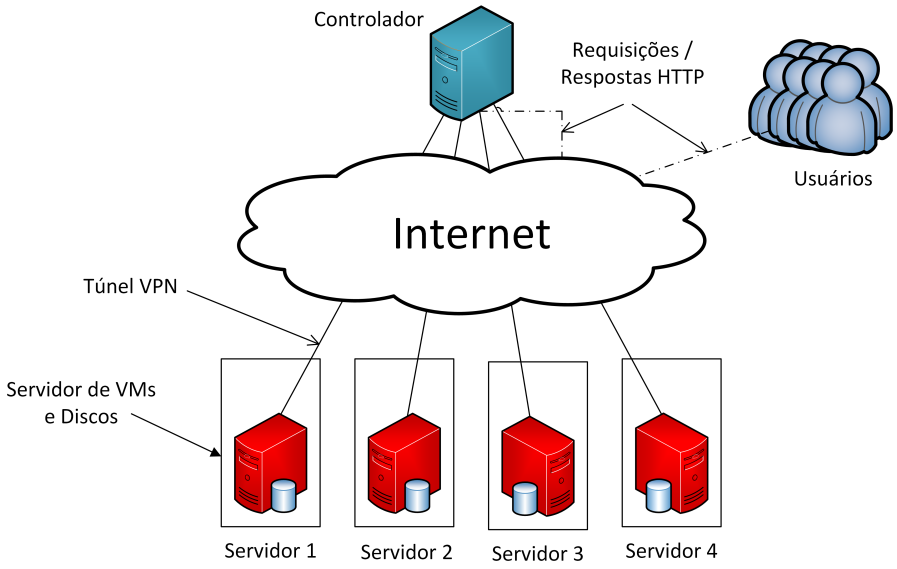

Figura 4.2: Ambiente de testes utilizado nos experimentos.

4.2 Experimentos

O objetivo da análise deste trabalho é avaliar o desempenho da rede entre o Controlador e os Servidores de VMs e Discos sob diferentes condições. Para isso, o tráfego de pacotes que passa pela interface VPN do Controlador é capturado para análise, já que o tráfego da rede WAN passa por essa interface.

4.2.1 Impacto do número de Servidores de VMs e Discos

Neste experimento, variou-se o número de Servidores de VMs e Discos ligados ao Controlador observando-se o impacto no tráfego e no serviço do RabbitMQ.

4.2.1.1 Análise do tráfego

A Figura 4.3 ilustra o tráfego gerado por um Servidor de VMs e Discos em consequência da troca de mensagens entre o banco de dados e o cinder-volume e entre o RabbitMQ e os componentes nova-compute e nova-network. O cinder-volume reporta seu estado a cada 10 segundos e atualiza as informações dos volumes a cada 60 segundos (pico maior entre 10 e 20 segundos). Da mesma forma, o RabbitMQ se comunica com os componentes do servidor a cada 10 segundos para atualizar o

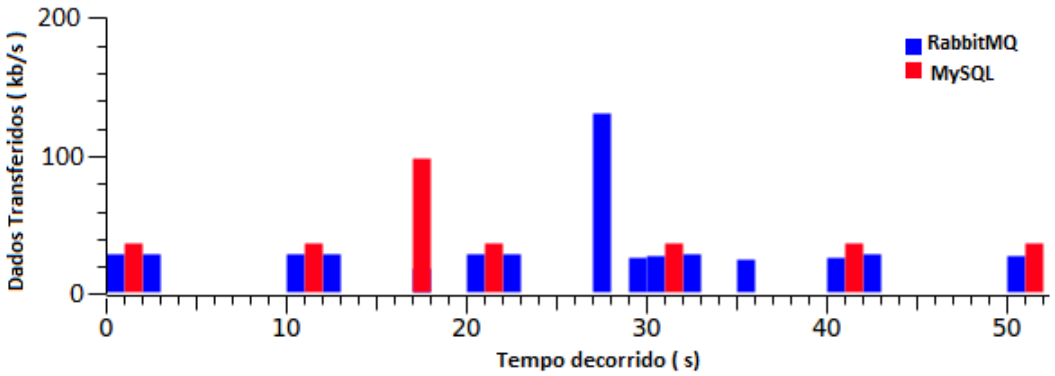

Figura 4.3: Amostra do tráfego de rede entre o Servidor de VMs e Discos e o Controlador.

estado dos serviços e a cada 60 segundos para atualizar a lista de instâncias dos servidores (pico maior entre 20 e 30 segundos). A Figura 4.4 mostra a contribuição desses dois tráfegos para o tráfego total na rede.

Na infraestrutura estudada, os componentes nova-compute e nova-network presentes nos Servidores de VMs e Discos enviam periodicamente atualizações sobre as instâncias para o nova-conductor, presente no Controlador, utilizando o sistema de filas de mensagens do RabbitMQ. O nova-conductor então insere essas informações no banco de dados do Nova. Da mesma forma, o componente cinder-volume, presente em cada Servidor de VMs e Discos, envia periodicamente atualizações sobre os volumes ao Controlador. No entanto, ele se comunica diretamente com o banco de dados do Cinder ao invés de utilizar o RabbitMQ e um componente especializado para agir sobre o banco de dados como o nova-conductor. Em um ambiente onde os servidores não possuam máquinas virtuais ou volumes, o tráfego gerado por cada servidor de VMs e Discos é semelhante, pois todos os componentes reportam a mesma condição.

Para determinar a influência do aumento do número de Servidores de VMs e Discos na rede foi realizado o seguinte experimento: a cada Servidor de VMs e Discos adicionado à infraestrutura amostrou-se o tráfego por 60 segundos. Este procedimento foi repetido 10 vezes. Com os resultados experimentais, realizou-se um ajuste linear dos pontos do tráfego total que resultou na equação da reta mostrada na Figura 4.5. O valor de R^2 indica o quão ajustado aos pontos é o

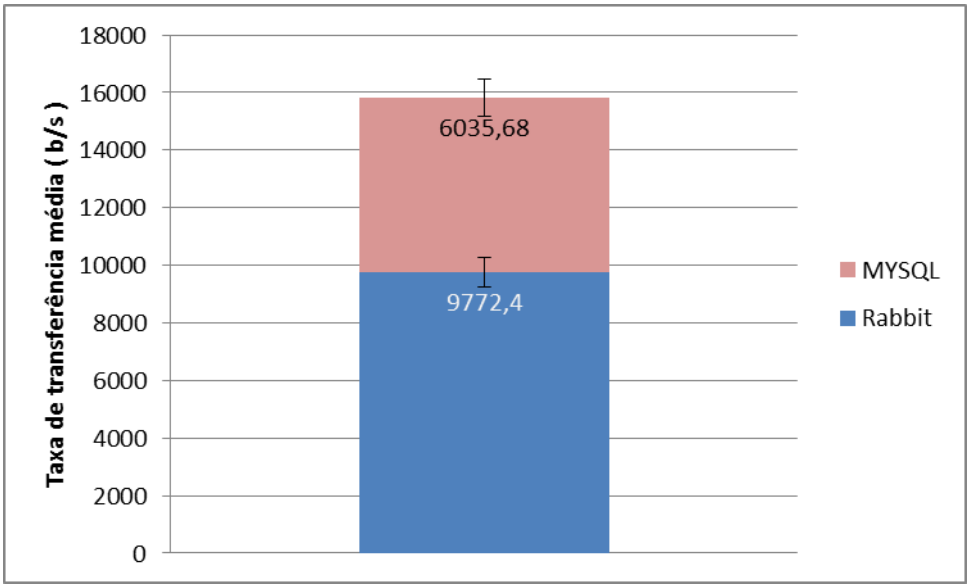

Figura 4.4: Distribuição do tráfego entre a comunicação dos componentes do servidor com o RabbitMQ e o MySQL.

modelo linearizado. Os valores de R^2 variam de 0 a 1, sendo o modelo exatamente linear para R^2 igual a 1. Nas medidas deste experimento o valor encontrado para R^2 foi 0,9966. A partir da equação da reta dada é possível concluir que cada servidor adiciona em média um tráfego de 15 kb/s à rede. Por extrapolação, em um cenário com 100 servidores, um tráfego contínuo de 1,5 Mb/s passaria pela interface do Controlador. Ou seja, considerando que a velocidade média mundial de enlaces de Internet é 3,9 Mb/s [28], apenas o tráfego dos servidores em um cenário sem máquinas virtuais representa mais de 35% desse valor.

4.2.1.2 Análise do RabbitMQ

Em um sistema de produção, o RabbitMQ precisa que o número de descritores de arquivos configurados para o sistema seja suficiente para lidar com múltiplas conexões concorrentes e filas. Caso o número de descritores chegue ao limite, os componentes não conseguirão se comunicar. Assim como no trabalho da Cisco [26], foi realizado um experimento para mapear o aumento do número de descritores de arquivos à medida que novos Servidores de VMs e Discos são adicionados à infraestrutura.

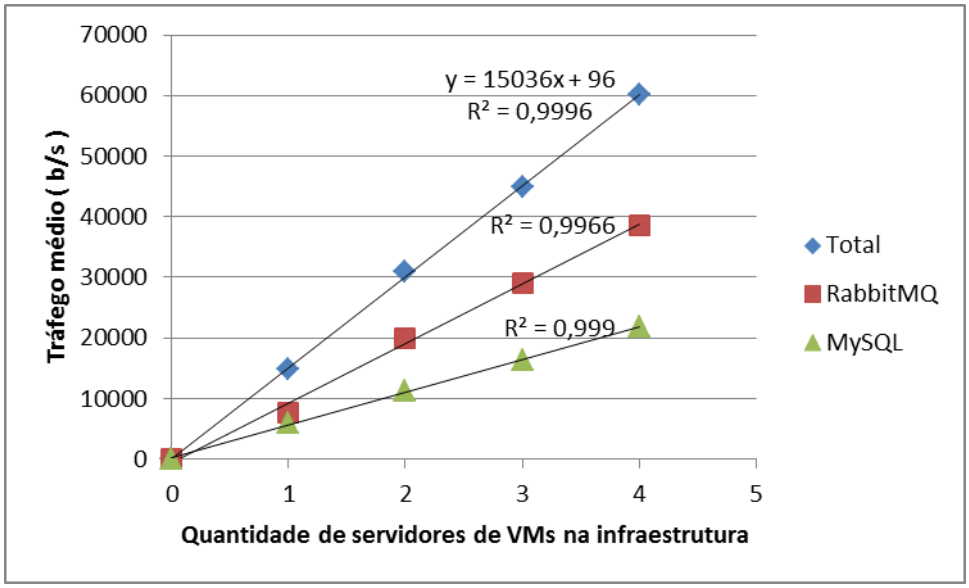

Figura 4.5: Tráfego na rede com o aumento do número de Servidores de VMs e Discos.

Neste experimento, o número de descritores de arquivos totais alocados pelo sistema para cada Servidor de VMs e Discos adicionados à infraestrutura é extraído. Essa medida foi repetida 10 vezes. A Figura 4.6 mostra o resultado desse teste e a reta gerada pelo ajuste linear. Substituindo-se o y da equação da reta pelo valor padrão de descritores de arquivos (1024) chega-se a um valor limite de 188 Servidores de VMs e Discos sem VMs instanciadas por Controlador. No entanto, é prática comum aumentar o número de descritores de arquivos disponíveis no sistema. Desta forma, aumentando o número de descritores de arquivos do usuário *rabbitmq* do sistema operacional para 65536, valor recomendado para ambientes de produção [29], chega-se a 12691 servidores.

4.2.2 Impacto do número de VMs por Servidor de VMs e Discos

Neste experimento variou-se o número de VMs em um único Servidor de VMs e Discos para analisar o impacto no tráfego da rede e no serviço do RabbitMQ.

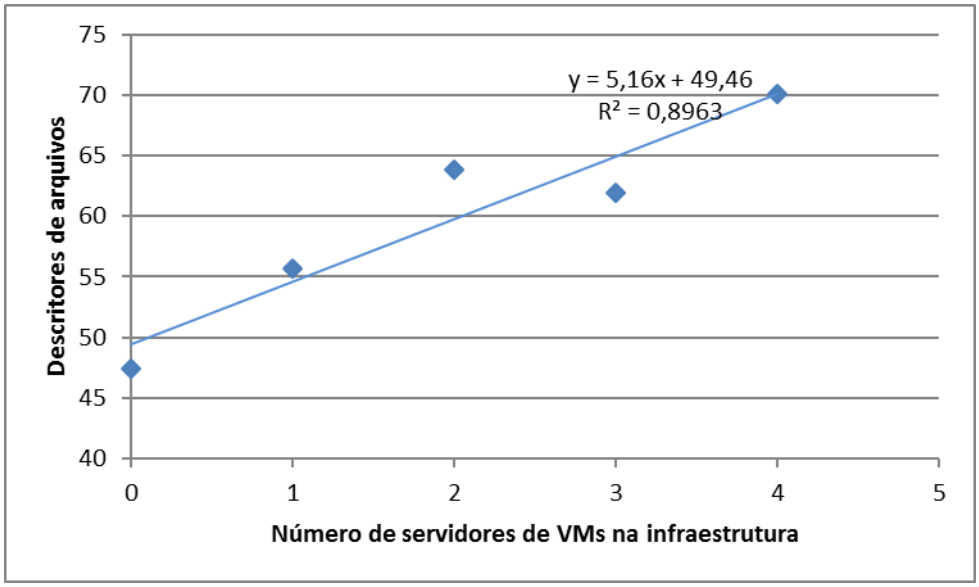

Figura 4.6: Impacto no número de descritores de arquivo com o aumento do número de Servidores de VMs e Discos.

4.2.2.1 Análise do tráfego

À medida que o número de instâncias de VMs e volumes aumenta, mais dados precisam ser enviados para o Controlador para atualizar os bancos de dados do Nova e do Cinder. Para avaliar o impacto do aumento do número de VMs instanciadas na rede, foi amostrado o tráfego após a criação de cada máquina virtual em um Servidor de VMs e Discos por 60 segundos. Esse experimento foi repetido 10 vezes para um total de 90 VMs por ciclo.

A Figura 4.7 ilustra o resultado do experimento. O ajuste linear dos dados gerou um R^2 igual a 0,9855 indicando um comportamento aproximadamente linear. Pela equação gerada, estima-se que cada VM gere uma taxa média 0,8 kb/s. Em um cenário com 100 servidores contendo 15 VMs cada, totalizando 1500 VMs, a carga total gerada pelas VMs seria de 1,2 Mb/s. Nesse cenário, somando-se a carga das VMs com o dos Servidores de VMs em Discos, analisado anteriormente, tem-se uma carga total de 2,7 Mb/s, valor que representa mais de 68% da velocidade média mundial de enlaces de Internet [28].

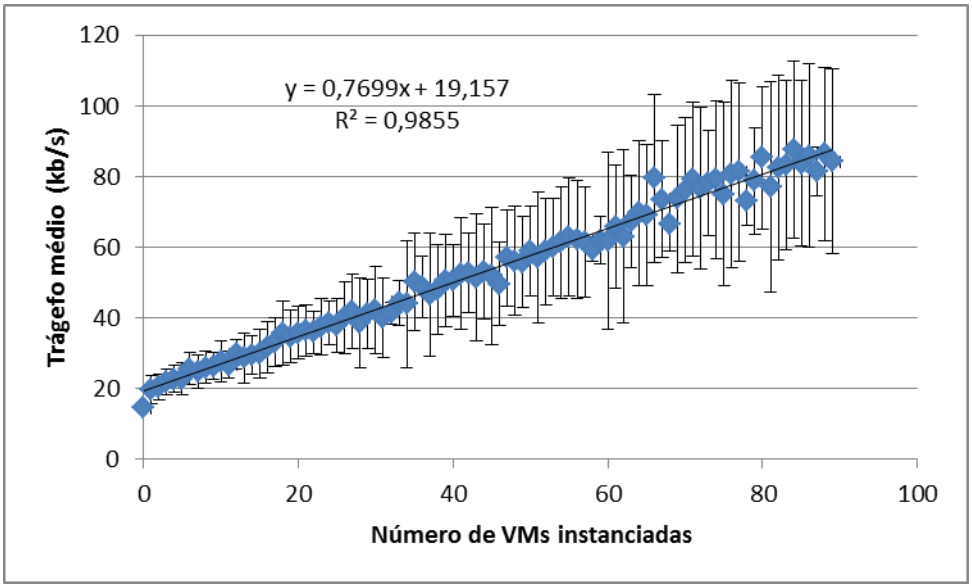

Figura 4.7: Tráfego na rede com o aumento do número de VMs por Servidor de VMs e Discos.

4.2.2.2 Análise do RabbitMQ

Neste experimento, o número total de descritores de arquivos alocados após a instanciação de cada máquina virtual em um Servidor de VMs e Discos é extraído, totalizando 90 VMs. A Figura 4.8 ilustra o resultado do experimento. O comportamento observado na figura se assemelha a um degrau. Na faixa de 80 máquinas virtuais instanciadas, o número de descritores de arquivos praticamente dobrou. A Cisco em [26], realiza um experimento semelhante extraindo o número de descritores alocados a cada servidor adicionado à infraestrutura após a criação de 20 máquinas virtuais em cada um, no entanto, seu resultado apresenta um comportamento linear. Essa diferença de resultados deve estar relacionada a diferenças na política de alocação de descritores de arquivos entre o Controlador desse trabalho e o da Cisco.

4.2.3 Impacto da criação de uma VM em um Servidor de VMs e Discos

Os experimentos a seguir têm o propósito de mostrar o comportamento do tráfego durante um processo de criação de uma máquina virtual.

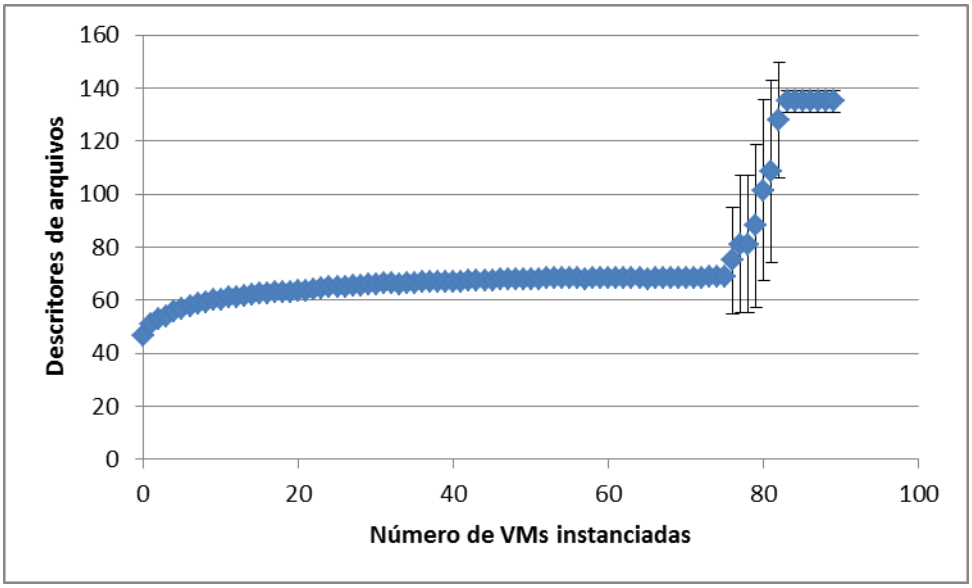

Figura 4.8: Impacto no número de descritores de arquivo com o aumento do número de VMs alocadas por servidor de VMs e Discos.

4.2.3.1 Análise do tráfego

No OpenStack, é possível inicializar uma instância a partir de uma imagem armazenada em uma unidade de disco efêmera, a qual armazena os dados enquanto a instância associada existir, ou a partir de uma imagem armazenada em um volume, que oferece armazenamento persistente. Quando o usuário pede para iniciar uma instância sem volume pela primeira vez, o Glance envia a imagem ao Servidor de VMs e Discos através do túnel VPN. A imagem é armazenada localmente no servidor e, então, copiada para uma unidade de disco efêmera. Neste caso, se uma nova instância de máquina virtual com a mesma imagem for criada no mesmo Servidor de VMs e Discos, a imagem não precisará ser passada novamente pela rede. No caso de instâncias com inicialização a partir de volume, primeiramente um volume vazio é criado, depois a imagem transferida para o Servidor de VMs e Discos é copiada para o volume. Diferentemente do Nova, o Cinder ainda não possui uma política de cache [30]. Dessa forma, sempre que uma nova instância com inicialização a partir de volume é criada, a imagem é transferida pela rede.

Para ilustrar o impacto na rede desses dois tipos de inicialização, foi amostrado o tráfego durante a criação da primeira e segunda instância de máquina virtual

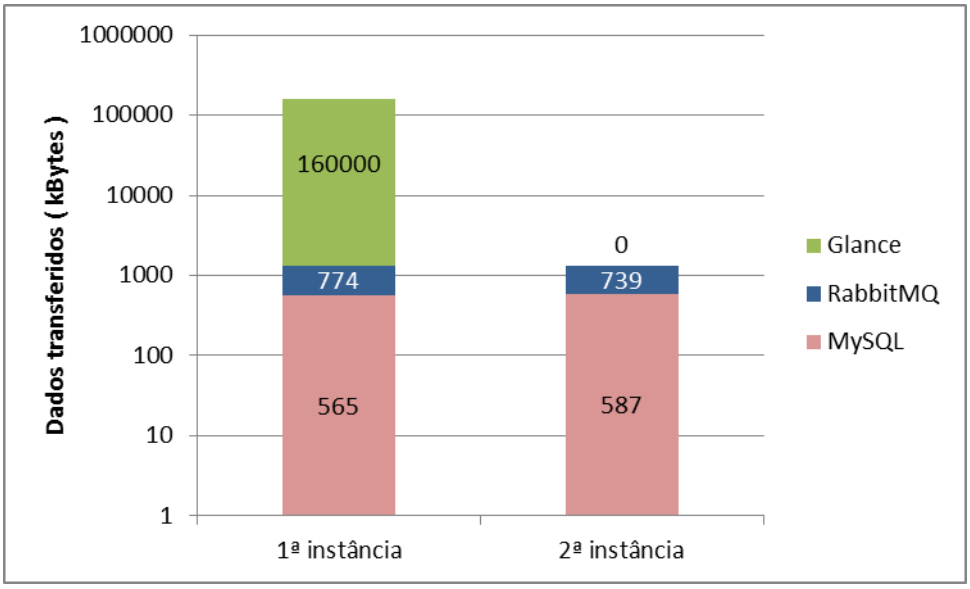

Figura 4.9: Total de dados transferidos pela rede durante a criação da primeira e segunda instância de uma máquina virtual com inicialização a partir da mesma imagem e sem volume.

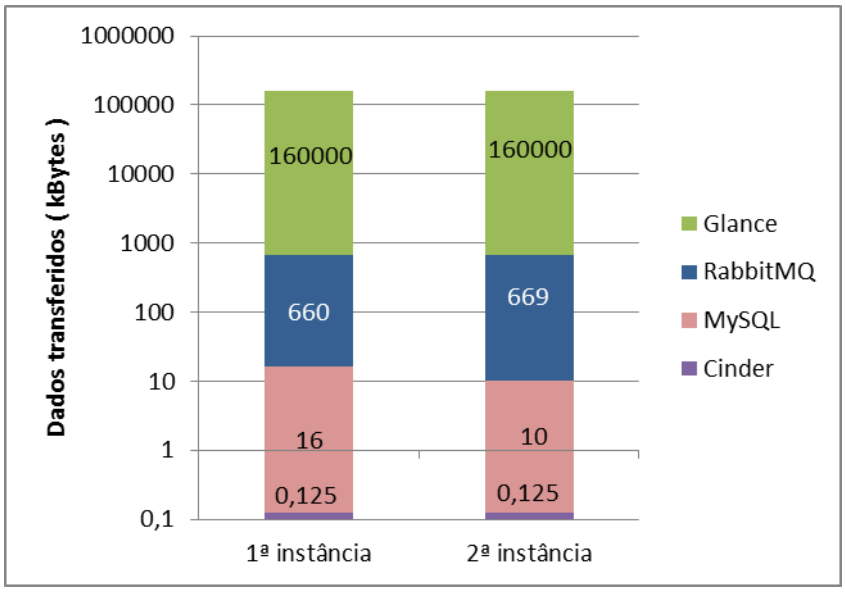

Figura 4.10: Total de dados transferidos pela rede durante a criação da primeira e segunda instância de uma máquina virtual com inicialização a partir da mesma imagem e com volume.

com e sem volume. Esse experimento foi repetido 10 vezes para cada caso utilizando-se sempre a imagem do sistema operacional Porteus de 160 MB [31] para inicializar as instâncias. A Figura 4.9 contém o total de dados transferidos durante a criação da primeira e segunda instância de uma máquina virtual sem volume. Nessa figura, observa-se que o total de dados amostrados para a primeira instância resulta da comunicação dos componentes do servidor com o RabbitMQ, o MySQL e o Glance. No entanto, a parcela do Glance não está presente para a segunda instância devido à utilização da imagem armazenada localmente. A Figura 4.10 ilustra a inicialização das instâncias com volume. Nesse caso, o total de dados amostrados resulta da comunicação dos componentes do servidor com o RabbitMQ, o MySQL, o Glance e o Cinder. Como esperado, o total de dados transferidos é semelhante para a primeira e segunda instância, pois a imagem sempre é transferida do Controlador para o Servidor de VMs e Discos. Pelos gráficos fica claro que a imagem passada pelo Glance é a parte mais significativa do tráfego. Quanto maior a imagem, maior o tempo de ocupação da banda. Portanto, os casos de uso do sistema devem ser pensados de forma a restringir a utilização de instâncias inicializadas a partir de volume para não sobrecarregar a rede.

4.2.4 Impacto da criação e exclusão de múltiplas VMs

Nos experimentos descritos a seguir, foi observado o tráfego de rede durante a criação e exclusão de múltiplas instâncias. Os experimentos são divididos em dois casos. No primeiro, o nova-api recebe uma requisição para criar múltiplas instâncias e em seguida recebe uma requisição para excluí-las. Esse caso representa a ação de um único usuário da nuvem. No segundo caso, o nova-api recebe requisições paralelas para criar e excluir instâncias. Esse caso representa as ações concorrentes de múltiplos usuários da nuvem.

4.2.4.1 Impacto da criação e exclusão de múltiplas VMs por um usuário

Este experimento ilustra o impacto no tráfego da rede quando um usuário realiza uma requisição para criar múltiplas instâncias e, ao fim desse processo, realiza outra requisição para excluí-las. Para automatizar esse experimento foi utilizado o cenário boot-and-delete-multiple.json do Rally para gerar as requisições para a API

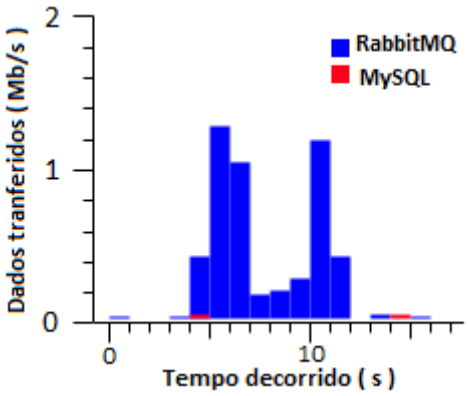

Figura 4.11: Exemplo de tráfego gerado pela criação e exclusão de 1 máquina virtual.

do Nova variando-se a quantidade de VMs solicitadas para valores entre 1 e 20. O experimento foi repetido 10 vezes. A Figura 4.11 ilustra o tráfego para criação e exclusão de uma VM e a Figura 4.12 para dez VMs. Na primeira, o tráfego atinge um valor de pico de aproximadamente 1,3 Mb/s durante o processo de criação da VM e 1,2 Mb/s na exclusão. Esses valores aumentam no segundo caso, ultrapassando 5 Mb/s. Além disso, o tempo entre o envio dos parâmetros para criação de VMs e o início da exclusão das máquinas aumenta consideravelmente. Esse comportamento indica um gargalo no Servidor de VMs para processar a criação de múltiplas instâncias.

A Figura 4.13 apresenta o tráfego de rede médio do experimento. Na faixa entre 5 e 10 instâncias o tráfego atinge um valor máximo de aproximadamente 1,2 Mb/s. Esses resultados mostram também que, durante o processo de criação e exclusão de máquinas virtuais para um único usuário, o tráfego médio de rede pode atingir um valor equivalente ao tráfego contínuo gerado por 1500 máquinas virtuais.

4.2.4.2 Impacto da criações e exclusão paralela de VMs por múltiplos usuários

Este experimento ilustra o impacto no tráfego de rede quando múltiplos usuários enviam simultaneamente requisições para criação e exclusão de instâncias. Para

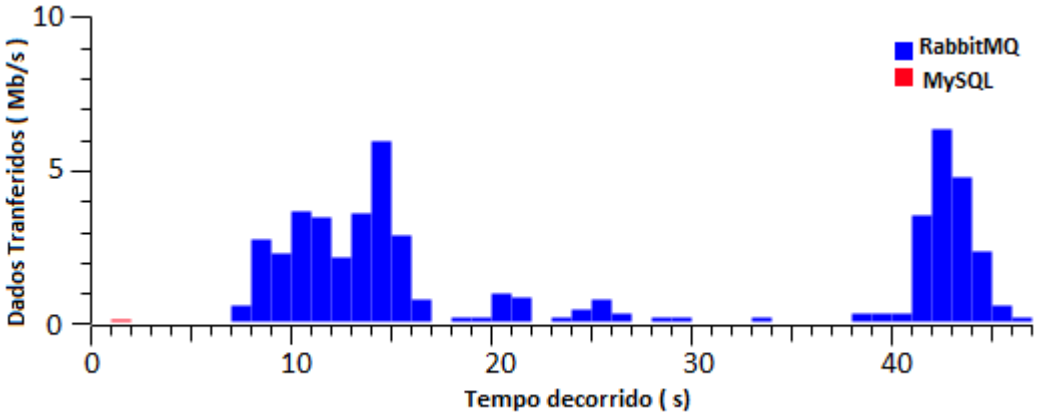

Figura 4.12: Exemplo de tráfego gerado pela criação e exclusão de 10 máquinas virtuais.

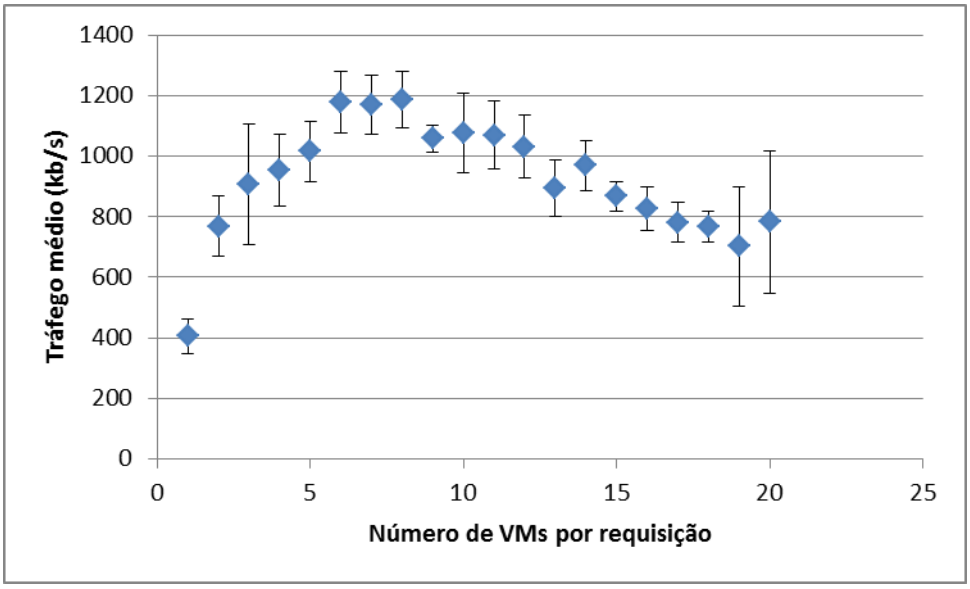

Figura 4.13: Tráfego médio gerado na criação e exclusão de múltiplas instâncias.

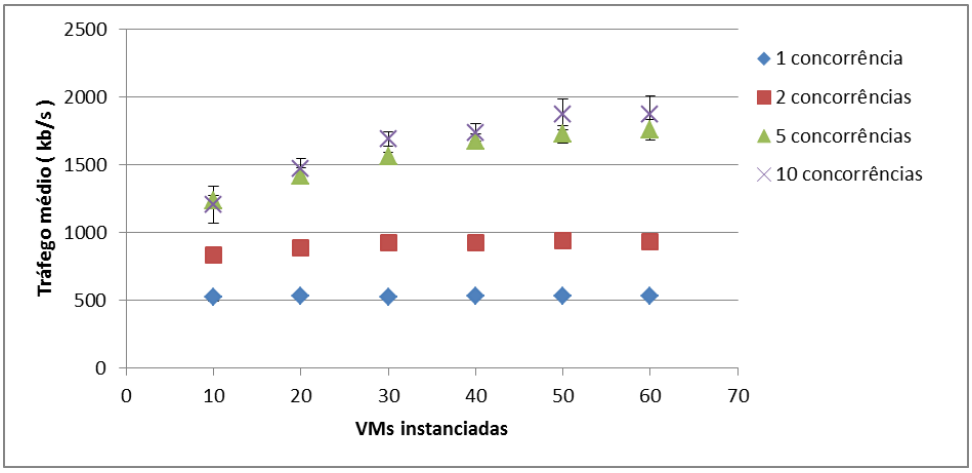

Figura 4.14: Tráfego médio gerado nas criações e destruições concorrentes de instâncias.

realizar este experimento, foi utilizado o cenário boot-and-delete.json do Rally. Neste cenário, um número máximo de instâncias que devem ser criadas é configurado. O Rally gera uma requisição para criar uma instância, espera essa ser criada e depois manda excluí-la. Na sequência faz um novo ciclo até atingir o total de iterações. Além disso, o Rally permite configurar a quantidade de requisições paralelas que são enviadas ao Nova API. Para um valor de concorrência igual a 5, por exemplo, são emitidas 5 requisições paralelas por vez e à medida que uma acaba, uma nova é enviada para manter o Nova API ocupado com um valor constante de requisições paralelas. Nos experimentos realizados foram criadas N VMs utilizando-se valores de concorrência de 1, 2, 5 e 10. O experimento foi repetido 10 vezes para cada número N de instâncias e concorrências.

A partir dos dados coletados gerou-se a Figura 4.14. Neste gráfico, observa-se que a variação da concorrência de 1 para 2 fez o tráfego médio dobrar, no entanto, o mesmo efeito não é observado de 5 para 10 concorrências, para os quais o valor do tráfego médio é praticamente igual. Assim como no experimento anterior, o Servidor de VMs não consegue otimizar a criação de mais de 5 instâncias simultaneamente. Desta forma, para mais de 5 concorrências, o tráfego médio máximo atinge um limite superior de aproximadamente 2 Mb/s.

Capítulo 5

Conclusões

O objetivo deste trabalho foi analisar a escalabilidade da nuvem OpenStack geo-distribuída proposta pelo GT-PID considerando-se as limitações de rede. Para isso, foram realizados estudos sobre a arquitetura do OpenStack e o modelo de comunicação adotado por seus componentes. Em seguida foram realizados experimentos para determinar a carga na rede gerada por servidores e VMs.

A análise do aumento do tráfego conforme o aumento do número de VMs e Servidores de VMs e Discos, mostraram que a carga contínua gerada na rede em uma arquitetura em produção, ou seja, com um número elevado de servidores e VMs, pode atingir valores significativos para uma rede WAN. Esse resultado é agravado quando consideram-se os picos de carga durante atividades de provisionamento e exclusão de máquinas virtuais. Conforme o experimento de criação e término de múltiplas instâncias, um usuário, por exemplo, pode solicitar a criação de cinco instâncias e gerar uma carga temporária na rede equivalente a 1500 máquinas virtuais em repouso. Dessa forma, no planejamento dos casos de uso dos usuários é preciso pensar em limitar a quantidade de VMs simultâneas que um usuário pode criar para não sobrecarregar a rede. Além disso, considerando-se a análise sobre os tipos de inicialização de VM, deve-se pensar em priorizar a criação de instâncias com disco efêmero e que podem estar ligadas a volumes para armazenamento persistente, sem que esse possua uma imagem.

O tráfego gerado nos experimentos realizados não é tão significativo para instalações padrões do OpenStack onde todos os servidores encontram-se na mesma

rede local. No entanto, para o cenário geodistribuído, a localização do Controlador deve ser cuidadosamente estudada para não gerar uma alta sobrecarga do enlace de acesso ao Controlador.

Como extensões do projeto, destacam-se alguns pontos. O primeiro é a realização de novos experimentos com um número maior de servidores para comparar com os resultados obtidos. Em seguida, a simulação de carga real nas VMs para determinar o impacto na rede das atividades dos usuários. Também é importante a realização de estudos para determinar um acordo de nível de serviço para os usuários da nuvem geodistribuída. Por fim, pode-se estender a análise realizada incluindo o Ceilometer e o Neutron, projetos do OpenStack recentemente incorporados à arquitetura do GT-PID, que servem respectivamente para o monitoramento de recursos e configurções avançadas de rede.

Referências Bibliográficas

[1] ZHANG, Q., CHENG, L., BOUTABA, R., "Cloud computing: state-of-the-art and research challenges", *Journal of internet services and applications*, v. 1, n. 1, pp. 7–18, 2010.

[2] MELL, P., GRANCE, T., "The NIST definition of cloud computing", *Computer Security Division, Information Technology Laboratory, National Institute of Standards and Technology Gaithersburg*, 2011.

[3] "Gartner Says Worldwide Cloud Infrastructure-as-a-Service Spending to Grow 32.8 Percent in 2015", Set. 2015. Disponível em: <*http* : *//www.gartner.com/newsroom/id/*3055225>.

[4] WILDER, B., *Cloud architecture patterns: using microsoft azure*. "O'Reilly Media, Inc.", 2012.

[5] CLOUD, A. E. C., "Amazon web services", *Retrieved November*, v. 9, pp. 2011, 2011.

[6] ZAHARIEV, A., "Google app engine"Helsinki University of Technology, 2009.

[7] NURMI, D., WOLSKI, R., GRZEGORCZYK, C., *et al.*, "The eucalyptus open-source cloud-computing system". In: *Cluster Computing and the Grid, 2009. CCGRID'09. 9th IEEE/ACM International Symposium on*, pp. 124–131, IEEE, 2009.

[8] KUMAR, R., JAIN, K., MAHARWAL, H., *et al.*, "Apache CloudStack: Open Source Infrastructure as a Service Cloud Computing Platform". In: *International Journal of advancement in Engineering technology, Management and Applied Science*, 2014.

[9] SEFRAOUI, O., AISSAOUI, M., ELEULDJ, M., "OpenStack: toward an open-source solution for cloud computing", *International Journal of Computer Applications*, v. 55, n. 3, pp. 38–42, 2012.

[10] "Cisco Global Cloud Index: Forecast and Methodology, 2013-2018", Set. 2015. Disponível em: $<http://www.cisco.com/c/en/us/solutions/collateral/service-provider/global-cloud-index-gci/Cloud_Index_White_Paper.pdf>$.

[11] YOUSEFF, L., BUTRICO, M., DA SILVA, D., "Toward a unified ontology of cloud computing". In: *Grid Computing Environments Workshop, 2008. GCE'08*, pp. 1–10, IEEE, 2008.

[12] SULTAN, N., "Cloud computing for education: A new dawn?", *International Journal of Information Management*, v. 30, n. 2, pp. 109–116, 2010.

[13] "CRM: coloque o cliente no centro de seus negócios", Set. 2015. Disponível em: $<https://www.salesforce.com/br/crm/>$.

[14] KUMAR, R., GUPTA, N., CHARU, S., *et al.*, "Open Source Solution for Cloud Computing Platform Using OpenStack", *International Journal of Computer Science and Mobile Computing*, v. 3, n. 5, pp. 89–98, 2014.

[15] COUTO, R. S., SCIAMMARELLA, T., BARRETO, H. F. S. S. M., *et al.*, "GTPID: Uma Nuvem IaaS Universitária Geograficamente Distribuída", *Quinta Conferencia de Directores de Tecnología de Informacion - TICAL 2015*, pp. 1–19, 2015.

[16] FUENTES, F., KAR, D. C., "Ethereal vs. Tcpdump: a comparative study on packet sniffing tools for educational purpose", *Journal of Computing Sciences in Colleges*, v. 20, n. 4, pp. 169–176, 2005.

[17] ASRODIA, P., PATEL, H., "Network traffic analysis using packet sniffer", *International Journal of Engineering Research and Applications*, v. 2, n. 3, pp. 854–856, 2012.

[18] "Companies Supporting The OpenStack Foundation", Set. 2015. Disponível em: $<https://www.openstack.org/foundation/companies/>$.

[19] JONES, B., LUXENBERG, S., MCGRATH, D., *et al.*, "RabbitMQ Performance and Scalability Analysis", *project on CS*, v. 4284, 2011.

[20] HINTJENS, P., *ZeroMQ: Messaging for Many Applications*. "O'Reilly Media, Inc.", 2013.

[21] ZHAO, S., LI, L., YANG, J., *et al.*, "Deployment and Performance Evaluation of Virtual Network based on OpenStack", *International Workshop on Cloud Computing and Information Security*, 2013.

[22] GEBREYOHANNES, M. B., "Network Performance study on OpenStack Cloud Computing", 2014, Dissertação de mestrado, University of Oslo.

[23] TIRUMALA, A., QIN, F., DUGAN, J., *et al.*, "Iperf: The TCP/UDP bandwidth measurement tool", Set. 2015. Disponível em: $<http://dast.nlanr.net/Projects>$.

[24] "Rally", Set. 2015. Disponível em: $<https://wiki.openstack.org/wiki/Rally>$.

[25] "Benchmarking OpenStack at megascale: How we tested Mirantis OpenStack at SoftLayer", Set. 2015. Disponível em: $<https://www.mirantis.com/blog/benchmarking-openstack-megascale-tested-mirantis-openstack-softlayer/>$.

[26] "OpenStack Havana Scalability Testing", Set. 2015. Disponível em: $<http://www.cisco.com/c/en/us/td/docs/solutions/Enterprise/Data_Center/OpenStack/Scalability/OHS.pdf>$.

[27] SEID, H. A., LESPAGNOL, A., "Virtual private network", Jun. 16 1998, US Patent 5,768,271.

[28] "AKAMAI'S STATE OF THE INTERNET, Q1 2014 REPORT", Set. 2015. Disponível em: $<https://www.akamai.com/us/en/multimedia/documents/secure/akamai-state-of-the-internet-report-q1-2014.pdf?tid=2F36A519595FA828BA75652D4238EEB3>$.

[29] “Controlling System Limits on Linux”, Set. 2015. Disponível em: <*https* : *//www.rabbitmq.com/install − debian.html*>.

[30] “Generic image cache functionality”, Set. 2015. Disponível em: <*http* : *//specs.openstack.org/openstack/cinder − specs/specs/liberty/image − volume − cache.html*>.

[31] “Porteus”, Set. 2015. Disponível em: <*http* : *//www.porteus.org/*>.