Uma Análise do Tráfego de Controle de uma Nuvem IaaS Geodistribuída

Tatiana Sciammarella[1], Rodrigo S. Couto[2], Marcelo G. Rubinstein[2]
Miguel Elias M. Campista[1] e Luís Henrique M. K. Costa[1] *

[1]Universidade Federal do Rio de Janeiro - PEE/COPPE/GTA - DEL/POLI
[2]Universidade do Estado do Rio de Janeiro - FEN/DETEL/PEL

{tatiana,miguel,luish}@gta.ufrj.br, {rodrigo.couto,rubi}@uerj.br

Abstract. *Geodistributed clouds are composed of several virtual machine servers scattered over sites interconnected by Wide Area Networks. In this scenario, the traffic generated by control messages can be prohibitive, because of the high bandwidth costs of these networks. This paper evaluates the potential of this type of traffic as an obstacle to increase the number of servers in the cloud. From the results obtained with the OpenStack orchestrator, we estimate that 100 servers and 15 virtual machines per server generates an average traffic about 2.7 Mb/s towards the infrastructure controller. To this average, we can add traffic bursts produced upon creation and destruction of virtual machines, further aggravating the problem. These results point out that, if the IaaS cloud is not well designed, the control traffic can become a concern for smaller clouds, such as collaborative and university ones.*

Resumo. *As nuvens geodistribuídas são compostas por diversos servidores de máquinas virtuais espalhados em sítios interconectados por redes de longa distância. Nesse cenário, o tráfego gerado por mensagens de controle pode ser proibitivo, visto que o custo de banda passante nessas redes é elevado. Este trabalho avalia o potencial desse tipo de tráfego como um obstáculo ao aumento do número de servidores da nuvem. A partir dos resultados obtidos com o orquestrador OpenStack, estima-se que com 100 servidores e 15 máquinas virtuais por servidor gera-se um tráfego médio de cerca de 2,7 Mb/s para o controlador da infraestrutura. A essa média pode-se somar os picos de tráfego gerados durante a criação e a destruição de máquinas virtuais, agravando o problema. Esses resultados mostram que, caso a nuvem IaaS não seja bem planejada, o tráfego de controle pode se tornar uma preocupação para nuvens de menor porte, como as colaborativas e universitárias.*

1. Introdução

A computação em nuvem vem conquistando popularidade nos mais diversos ramos produtivos por permitir a terceirização dos serviços de tecnologia da informação. Diferentes modelos de serviço podem ser oferecidos, como é o caso do IaaS (*Infrastructure-as-a-Service* - Infraestrutura como Serviço) [Jennings e Stadler, 2015], reduzindo os gastos com manutenção e administração de recursos computacionais. No modelo IaaS, o

*Este trabalho foi realizado com recursos da RNP, CNPq, FAPERJ e CAPES.

provedor de infraestrutura disponibiliza, sob demanda, máquinas virtuais (*Virtual Machines* - VMs) aos seus clientes para uso personalizado. Tais VMs podem ser acessadas facilmente pela Internet através de um navegador *web*, por exemplo. Algumas nuvens IaaS são geodistribuídas, ou seja, possuem uma infraestrutura dividida em sítios interconectados por redes de longa distância (*Wide Area Network* - WAN), o que tem reflexos em termos de resiliência e desempenho [Develder et al., 2012].

As grandes nuvens geodistribuídas são compostas por sítios autônomos interconectados, com centenas ou milhares de servidores cada um [Develder et al., 2012]. Esses servidores, cujo papel é hospedar as VMs dos clientes, são gerenciados por um orquestrador de nuvem, como o OpenStack [OpenStack, 2015] e o CloudStack [Apache, 2015]. O orquestrador executa diversas tarefas de controle da infraestrutura, como a escolha dos servidores hospedeiros de um determinado conjunto de VMs, a criação e a destruição de VMs, a autenticação de usuários e a coleta de estatísticas de uso. Geralmente, cada sítio de uma nuvem geodistribuída possui uma ou mais máquinas que atuam como controladores, executando os módulos principais de orquestradores como o OpenStack ou o CloudStack. Eventualmente, para levar em conta aspectos da arquitetura geodistribuída, os módulos do controlador de um sítio trocam mensagens com os outros sítios da nuvem.

Um cenário de utilização importante de uma nuvem IaaS geodistribuída é o compartilhamento de recursos computacionais entre diferentes entidades. Por exemplo, universidades ou centros de pesquisa possuem recursos computacionais que podem ser compartilhados através de uma única nuvem IaaS [Couto et al., 2015]. Esse tipo de serviço é denominado nuvem IaaS colaborativa, na qual diferentes entidades com interesses comuns compartilham seus recursos computacionais [Endo et al., 2011]. Uma característica das nuvens IaaS colaborativas, diferente de outras nuvens geodistribuídas, é que o número de servidores de VMs por sítio pode ser pequeno, enquanto o número de sítios pode ser muito grande. O cenário é de múltiplas instituições com recursos mais modestos em vez de uma grande corporação com centros de dados distribuídos. Sendo assim, não é desejável a instalação de um controlador para cada sítio, como ocorre em grandes nuvens geodistribuídas. Dessa forma, um único controlador centralizado pode ser utilizado para controlar os recursos de todos os sítios através de redes WAN [Couto et al., 2015]. Esse tipo de comunicação centralizada introduz por outro lado desafios na implementação de uma infraestrutura de nuvem colaborativa, principalmente em relação à escalabilidade do sistema. O tráfego de controle gerado se concentra próximo ao controlador, podendo sobrecarregá-lo. Além disso, essa maior carga de controle pode elevar os custos com banda passante, que são mais altos em redes WAN.

Assim, este trabalho analisa o impacto da comunicação entre o nó controlador da nuvem e os nós servidores de VMs na realização de tarefas de controle em uma nuvem OpenStack. O objetivo é verificar até que ponto o tráfego de controle representa um obstáculo ao crescimento de uma nuvem colaborativa. As contribuições deste trabalho são a identificação dos elementos que possam representar um gargalo na infraestrutura geodistribuída e a análise das trocas de mensagens entre o nó controlador e os servidores de VMs. A partir dos resultados obtidos é possível concluir, por exemplo, que cada servidor de VMs adicionado à infraestrutura pode gerar aproximadamente um aumento médio de 15 kb/s no tráfego da rede, enquanto que cada VM em repouso contribui com 0,77 kb/s. Consequentemente, estima-se que em uma nuvem com 100 servidores e 15 máquinas vir-

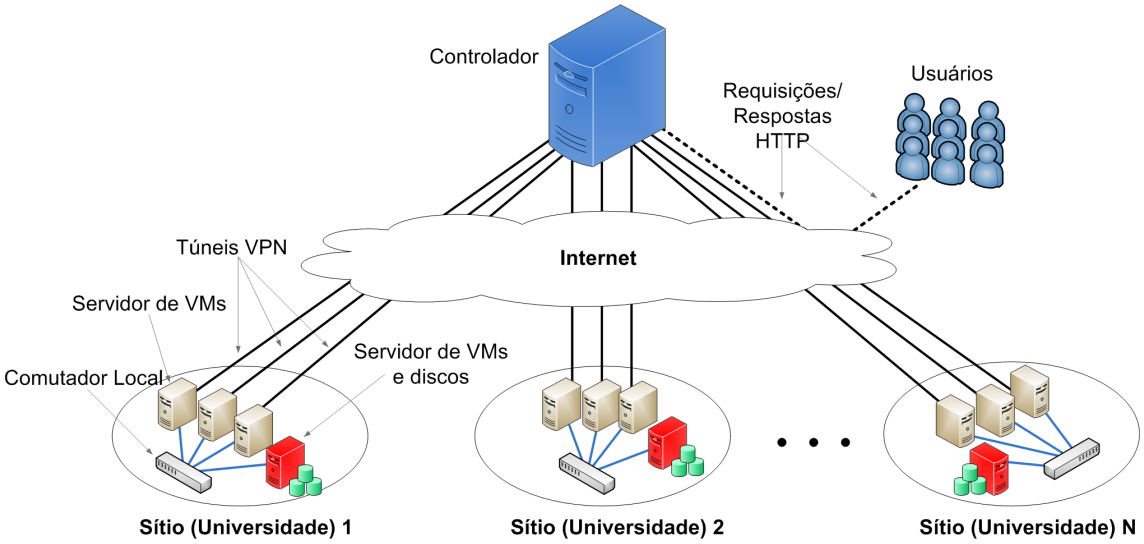


Figura 1. Arquitetura geodistribuída da nuvem colaborativa utilizada neste trabalho.

tuais por servidor, um tráfego médio de cerca de 2,7 Mb/s seria gerado para o controlador da infraestrutura. Apesar de esse valor ser pequeno para redes WANs comumente instaladas em grandes provedores de nuvem, esse tráfego pode rapidamente se tornar uma preocupação para nuvens menores, como as colaborativas e universitárias.

Este trabalho está organizado da seguinte forma. A Seção 2 apresenta a arquitetura da nuvem colaborativa. A Seção 3 descreve os detalhes relevantes do orquestrador de nuvens adotado, o OpenStack. A Seção 4 descreve a plataforma experimental utilizada para a avaliação de desempenho e analisa os resultados obtidos. A Seção 5 descreve os trabalhos relacionados, destacando a originalidade da análise realizada neste trabalho. Finalmente, a Seção 6 conclui o trabalho e indica possíveis trabalhos futuros.

2. Arquitetura da Nuvem Colaborativa

Este trabalho utiliza uma arquitetura em nuvem geodistribuída entre universidades e centros de pesquisa, proposta em [Couto et al., 2015]. O objetivo fundamental é compartilhar recursos computacionais entre as instituições envolvidas. Assim, em momentos de pico de utilização da infraestrutura local de uma instituição, seus usuários podem utilizar recursos computacionais ociosos de outras instituições, compartilhados através da nuvem. Por um lado, as instituições aumentam o seu poder computacional de maneira intrínseca ao modelo em nuvem; por outro lado, devem oferecer parte dos seus próprios recursos para que a nuvem funcione. Essa é a ideia da colaboração, calcada na possibilidade de uso intenso dos próprios recursos por parte das instituições de maneira assíncrona.

A arquitetura de nuvem colaborativa, apresentada na Figura 1, consiste em sítios, instalados em universidades ou centros de pesquisa, que são conectados a um Controlador centralizado. Cada sítio possui máquinas dos tipos Servidor de VMs e Discos e Servidor de VMs, conectadas entre si por um comutador local. Ambos os tipos de máquinas são utilizados para hospedar VMs, utilizando o KVM [Kivity et al., 2007] como hipervisor. O Servidor de VMs e Discos, por sua vez, possui a funcionalidade adicional de centralizar o disco virtual de todas as VMs de um sítio, permitindo assim a migração ao vivo de

VMs entre as máquinas do mesmo sítio [Couto et al., 2015]. O Controlador, além de realizar todas as funções de gerenciamento das máquinas dos sítios, é responsável por receber e atender requisições de criação de VMs pelos usuários. Além disso, fornece interfaces de acesso a essas VMs para os usuários. Tanto os Servidores de VMs como os Servidores de VMs e Discos estão conectados ao Controlador através de túneis VPN (*Virtual Private Network*) para atravessar a Internet. Para o gerenciamento da arquitetura da Figura 1, tanto o nó Controlador quanto as máquinas dos sítios executam o orquestrador OpenStack [OpenStack, 2015], detalhado na próxima seção.

O cenário previsto suporta a existência de sítios pequenos (p.ex., laboratórios), com disponibilidade possivelmente de apenas um servidor. Nesse caso, não é satisfatório executar uma solução completa de nuvem em cada sítio da infraestrutura, justificando a existência de um único controlador. Além disso, muitas vezes o parceiro de outra universidade não tem experiência de administração e todo procedimento pode ser feito remotamente. Note que a nuvem proposta pode ser integrada a outras soluções de colaboração e geodistribuição, como federação de nuvens [Barros et al., 2015], pois a infraestrutura completa é vista com uma única nuvem OpenStack. O uso de federação, por exemplo, pode ser uma solução utilizada para aumentar o número de sítios suportados.

3. OpenStack

O OpenStack é um software de código aberto responsável pelo gerenciamento dos recursos computacionais de infraestruturas de nuvens. Essa plataforma se destaca pela sua grande comunidade desenvolvedora e por sua lista de patrocinadores, que incluem empresas como Google, Intel, IBM e HP [OpenStack, 2015]. O OpenStack possui arquitetura modular, subdividida em projetos, favorecendo sua personalização para diferentes cenários. Esses projetos são serviços especializados para cada tarefa de gerenciamento da nuvem. Neste trabalho utiliza-se a versão Juno do OpenStack.

3.1. Projetos

A arquitetura do OpenStack é dividida em projetos com diferentes finalidades. Nesta seção, são considerados apenas os projetos relacionados aos serviços necessários à uma nuvem IaaS, caso da nuvem colaborativa utilizada cuja arquitetura está ilustrada na Figura 1. Esses projetos estão ilustrados na Figura 2 e são detalhados a seguir:

- ***Dashboard***: Esse projeto, também conhecido como Horizon, fornece uma interface *web* de gerenciamento do sistema. Para tal, esse projeto se comunica por meio de APIs (*Application Programming Interfaces*). Através da interface *web* os usuários podem criar, gerenciar e acessar suas VMs, enquanto o administrador pode executar tarefas como criação e especificação dos limites de alocação de recursos computacionais para os usuários.
- ***Compute***: Também conhecido como Nova, esse serviço é responsável pela interação com o hipervisor, permitindo o provisionamento e o gerenciamento de VMs. O Nova é composto por diversos módulos que se comunicam entre si e estão distribuídos pelas máquinas da infraestrutura, como mostrado mais adiante.
- ***Image Service***: Esse projeto, também conhecido como Glance, oferece um serviço de descoberta, registro, recuperação e armazenamento de imagens para inicialização de novas VMs. Uma imagem pode ser uma nova instalação de sistema operacional ou até mesmo uma cópia do disco de uma VM já configurada. O próprio usuário pode submeter suas imagens para o repositório do Glance.

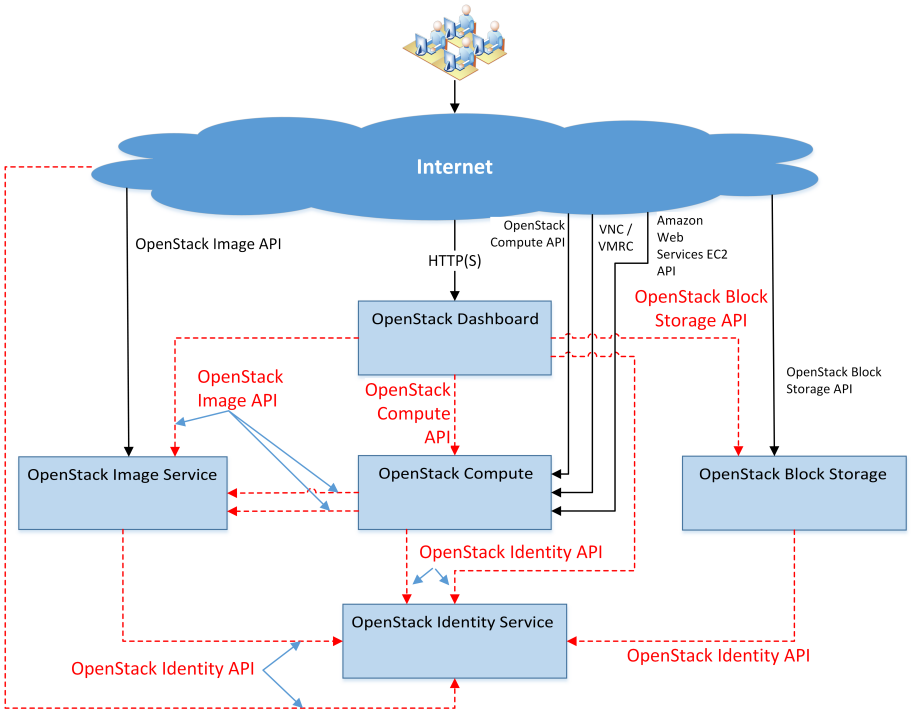


Figura 2. Interação dos projetos no OpenStack. As caixas azuis representam os projetos *Dashboard, Image Service, Compute, Block Storage* e *Identity Service*.

- ***Block Storage***: Conhecido também como Cinder, esse projeto fornece um serviço de armazenamento persistente para as VMs, em unidades denominadas volumes. Esses volumes representam discos virtuais e podem ser acessados quando estão associados às instâncias de VMs. Quando contém uma imagem, um volume pode ser uma unidade de inicialização (*boot*) para uma VM.
- ***Identity Service***: Também conhecido como Keystone, esse projeto é responsável pelo gerenciamento de identidades dos usuários e controle de acesso aos serviços.

Alguns dos projetos do OpenStack são divididos em módulos. Dentro de cada projeto, os módulos se comunicam utilizando filas de mensagens implementadas pelo *middleware* RabbitMQ [RabbitMQ, 2015], detalhado mais adiante. Além disso, um banco de dados MySQL é utilizado para armazenar informações referentes a cada projeto.

Os projetos Nova, Cinder, Glance e Keystone, todos utilizados na nuvem colaborativa deste trabalho, são exemplos de projetos do OpenStack divididos em módulos. Apesar das descrições detalhadas dos módulos serem omitidas neste trabalho por questões de concisão, a Figura 3 mostra a distribuição dos módulos entre os diversos tipos de máquina da arquitetura da Figura 1. Note que o Controlador hospeda todos os serviços relacionados a APIs, além do banco de dados e das filas de mensagens. Observe ainda que módulos responsáveis pela comunicação com hipervisores (nova-compute) ou por oferecer funcionalidades de redes para as VMs (nova-network) só existem nos nós servidores. Esses módulos não estão, por exemplo, presentes no controlador já que as VMs não são hospedadas neste nó. Por outro lado, módulos de interação com o usuário, por exemplo para acesso à VM (nova-novncproxy) ou para autenticação (nova-consoleauth e o projeto Keystone) são exclusivos do controlador. A principal diferença entre o Servidor de VMs e o Servidor de VMs e Discos é o módulo cinder-volume, utilizado para gerenciamento

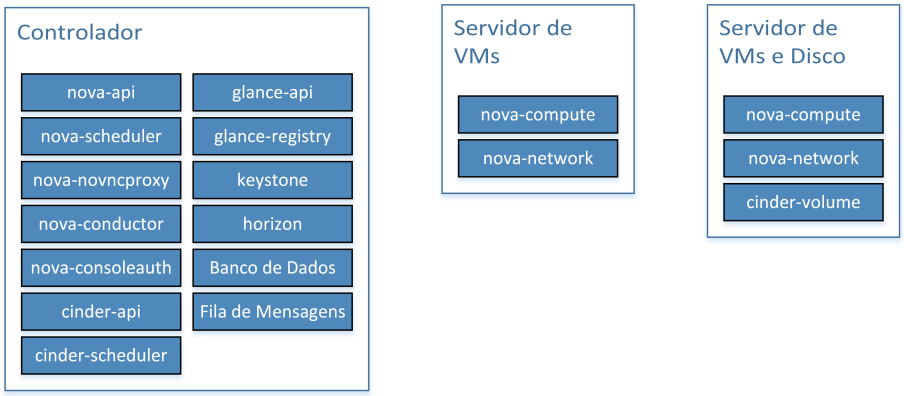
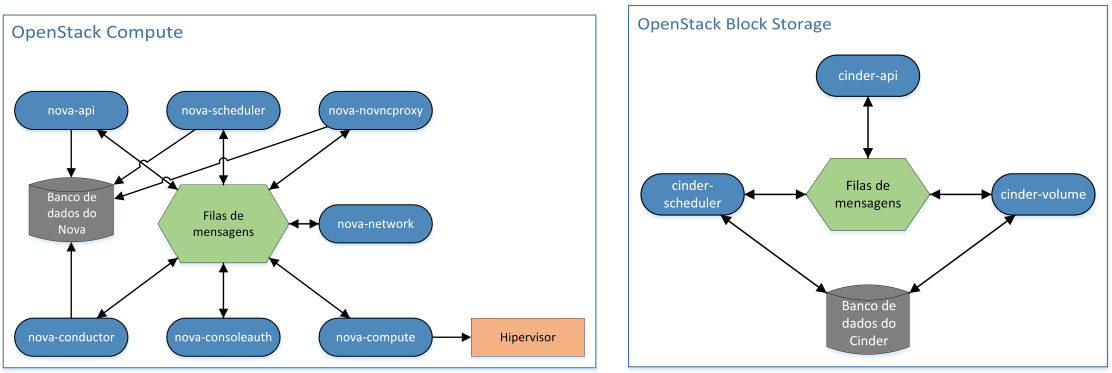


Figura 3. Distribuição dos módulos do OpenStack na arquitetura utilizada. A distribuição é realizada conforme os serviços oferecidos pela nuvem e a consequente função de cada um dos módulos na arquitetura.



(a) Componentes do Nova. (b) Componentes do Cinder.

Figura 4. Exemplo de projetos do OpenStack que utilizam comunicação direta entre os módulos e indireta com o banco de dados.

de volumes, e a hospedagem de um serviço NFS (*Network File System*), não mostrado na figura, utilizado para o armazenamento de discos virtuais.

3.2. Comunicação entre Serviços e Módulos

No OpenStack, os módulos e serviços podem se comunicar indiretamente por meio de filas de mensagens, ou diretamente por meio de APIs. Além disso, os módulos se comunicam com o banco de dados a partir de comandos MySQL. A fila de mensagens é utilizada exclusivamente para comunicação entre os módulos de um determinado projeto, como mostram as Figuras 4(a) e 4(b). Essas ilustram os módulos do Nova e do Cinder, respectivamente, e a comunicação entre eles a partir da fila de mensagens. A comunicação direta é utilizada para comunicação entre módulos de diferentes projetos e interação com o banco de dados que, na arquitetura proposta, está centralizado no Controlador.

A fila de mensagens utiliza o protocolo AMQP *(Advanced Message Queuing Protocol)* [ISO/IEC, 2014], implementado pelo *middleware* de mensagens RabbitMQ [RabbitMQ, 2015]. O AMQP é um protocolo aberto, cujo objetivo é promover a troca de mensagens entre aplicações diferentes, independentemente de como elas são implementadas. O RabbitMQ implementa um modelo *publish/subscribe* de troca de men-

Tabela 1. Configuração das máquinas utilizadas nos experimentos.

Máquina	CPU	RAM
Controlador	Intel Core i7 CPU 860 @ 2,80 GHz	8 GB
Servidor 1	Intel Core i7-4930K CPU @ 3,40 GHz	32 GB
Servidor 2	Intel Core i7 CPU 860 @ 2,80 GHz	8 GB
Servidor 3	Intel Xeon CPU E3-1241 v3 @ 3,50 GHz	32 GB
Servidor 4	Intel Core2 Quad CPU Q9400 @ 2,66 GHz	6 GB

sagens. Os módulos que geram mensagens enviam-nas para o *middleware*, que armazena as mensagens na fila apropriada. Essas mensagens ficam armazenadas até o momento em que os módulos que registraram interesse em recebê-las estejam prontos para consumi-las. Devido à característica geodistribuída da nuvem colaborativa, as mensagens geradas nos servidores são enviadas à fila RabbitMQ hospedada no Controlador.

Para comunicação direta, 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*, utilizando o padrão REST (*REpresentational State Transfer*). Dessa forma, os serviços do OpenStack podem ser acessados através da Internet, como ilustrado na Figura 2. Além de serem usadas para interação com componentes externos à infraestrutura, as APIs são utilizadas para comunicação entre módulos de projetos diferentes. Como os serviços de API são hospedados no Controlador da infraestrutura, essa máquina também centraliza o tráfego desse tipo de comunicação. Além disso, o Controlador centraliza o tráfego MySQL, já que essa máquina hospeda o banco de dados utilizado em todos os projetos.

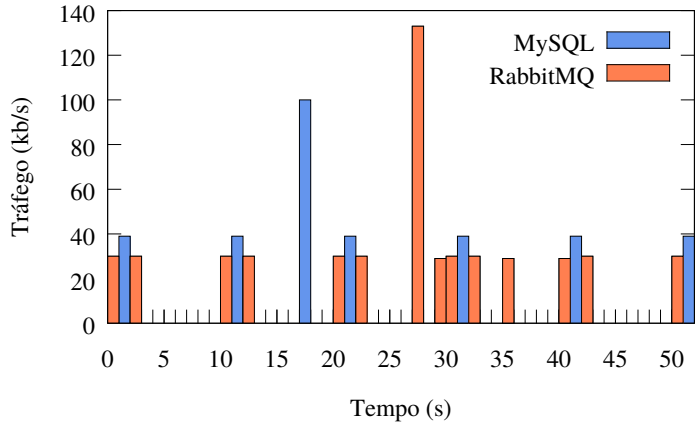
Como visto anteriormente, o Controlador é responsável por todas as interações entre os módulos, centralizando o tráfego de controle. Consequentemente, é importante estudar o tráfego gerado nessas interações, de forma a dimensionar melhor a capacidade de rede alocada ao nó Controlador.

4. Avaliação Experimental

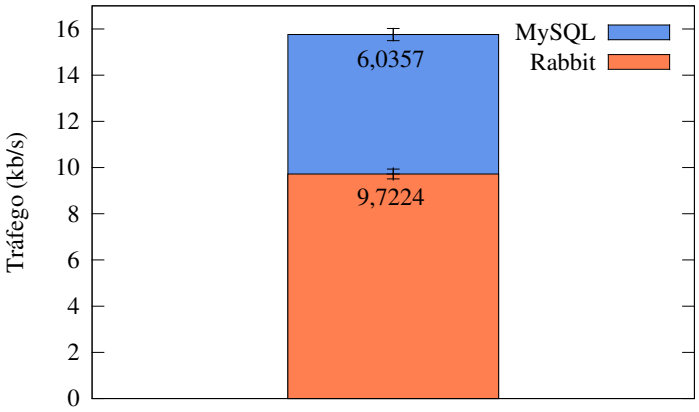
O objetivo da análise deste trabalho é avaliar o tráfego de rede entre o Controlador e os servidores sob diferentes condições. Para isso, os experimentos são realizados de acordo com a infraestrutura da Figura 1, capturando o tráfego na interface VPN do Controlador, que consiste no seu tráfego de rede WAN.

Na infraestrutura empregada nos experimentos, optou-se em utilizar somente Servidores de VMs e de Discos. Em comparação aos Servidores de VMs, esses servidores possuem um componente a mais (cinder-volume) que se comunica com o Controlador através do túnel VPN. Portanto, os Servidores de VMs e de Discos representam o pior caso para a análise proposta. Em alguns experimentos, são utilizados quatro destes servidores, caracterizando uma nuvem com quatro sítios, e em outros apenas um. Para garantir o controle sobre o cenário e isolamento de fatores externos, todos os sítios encontram-se na mesma localidade física e são conectados por uma rede local, ainda usando VPN, ao invés da Internet. A Tabela 1 contém a configuração de cada máquina da rede de testes.

Para todos os experimentos, à exceção das amostras apresentadas ao longo do



(a) Amostra do tráfego entre o Servidor de VMs e Discos e o Controlador ao longo do tempo.



(b) Distribuição do tráfego entre a comunicação dos componentes do servidor com o RabbitMQ e o MySQL.

Figura 5. Tráfego RabbitMQ e MySQL gerado a partir de um único servidor de VMs e Discos para o Controlador.

tempo, são calculadas médias e intervalos de confiança de 95%. Em algumas figuras, os intervalos não aparecem por serem muito pequenos. A partir das capturas, realizam-se diferentes análises apresentadas a seguir.

4.1. Impacto do número de Servidores de VMs e Discos

Neste experimento, inicialmente mediu-se o tráfego entre um Servidor de VMs e Discos, sem VMs instanciadas, e o Controlador. Vale notar que, neste experimento, nenhuma VM está instanciada nos servidores. A Figura 5(a) apresenta o tráfego gerado devido à troca de mensagens entre o RabbitMQ e os componentes nova-compute e nova-network e entre o banco de dados e o cinder-volume. Na infraestrutura utilizada, os componentes nova-compute e nova-network presentes nos Servidores de VMs e Discos enviam periodicamente atualizações para o nova-conductor, que executa 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

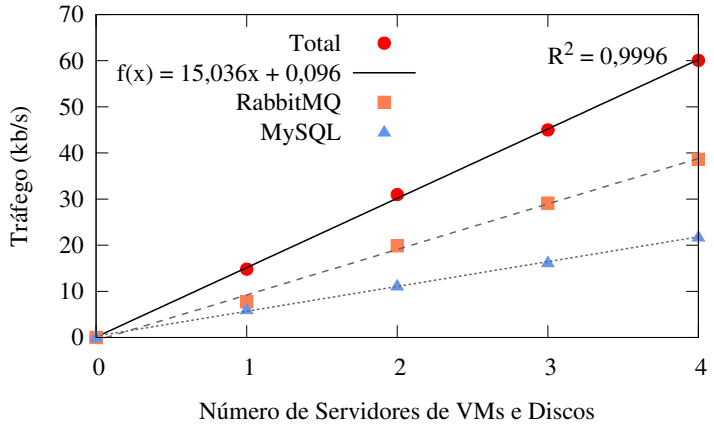


Figura 6. Tráfego na rede com o aumento do número de Servidores de VMs e Discos.

ao Controlador. No entanto, diferentemente do Nova, ele se comunica diretamente com o banco de dados do Cinder usando MySQL.

Observa-se na Figura 5(a) que o RabbitMQ se comunica com os componentes do servidor a cada 10 s, para atualizar o estado dos serviços, e que entre 20 e 30 s há um pico, correspondente à atualização da lista de instâncias dos servidores (essa atualização é realizada a cada 60 s). Da mesma forma, o cinder-volume (rótulo MySQL na figura) relata o seu estado a cada 10 s e entre 10 e 20 s há um pico, relativo à atualização das informações dos volumes (realizada periodicamente a cada 60 s). A Figura 5(b) mostra a contribuição desses dois tipos de comunicação para o tráfego médio na rede, calculado durante 60 s, em 25 experimentos diferentes, quando apenas o Servidor 1 está conectado ao Controlador. Pode-se observar que a maior parte do tráfego envolve o RabbitMQ, ou seja, o tráfego gerado pelo Nova. Além disso, é possível verificar que o tráfego total é de aproximadamente 15 kb/s.

Em seguida, incrementou-se o número de Servidores de VMs e Discos conectados ao Controlador, medindo o impacto no tráfego. A cada Servidor de VMs e Discos adicionado à infraestrutura amostrou-se o tráfego por 60 s. Este procedimento foi repetido 10 vezes. A Figura 6 apresenta o tráfego total na rede e também os tráfegos relacionados somente ao RabbitMQ e ao MySQL, em função do número de Servidores de VMs e Discos. Pode-se observar que o tráfego na rede aumenta com o crescimento do número de servidores. Em função da tendência linear observada nos resultados, foram realizados ajustes lineares dos pontos dos tráfegos que resultaram nas retas apresentadas na mesma figura. Para esse ajuste linear, o valor de R^2 indica o quão ajustado aos pontos é o modelo linearizado. Os valores de R^2 variam de 0 a 1, sendo o modelo exatamente linear para R^2 igual a 1. Para o tráfego total na rede, o valor encontrado para R^2 foi 0,9966. A partir da equação da reta obtida é possível concluir que cada servidor adiciona em média um tráfego de 15 kb/s à rede. Essa constatação coincide com o resultado da Figura 5(b). Assim, por extrapolação, em um cenário com 100 servidores, um tráfego de 1,5 Mb/s passaria pela interface de rede do Controlador.

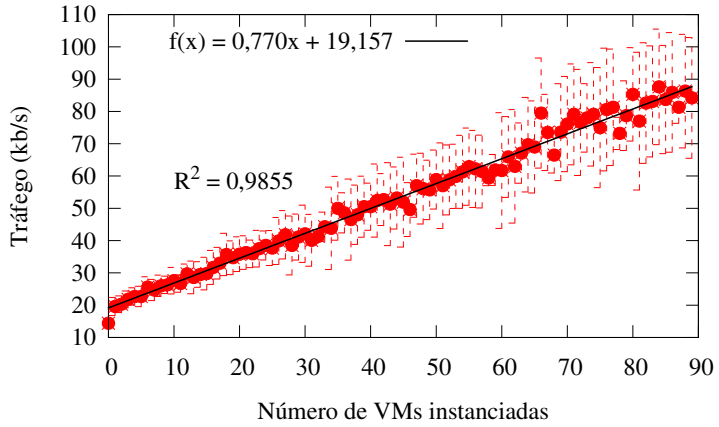


Figura 7. Tráfego na rede com o aumento do número de VMs.

4.2. Impacto do número de VMs por Servidor de VMs e Discos

Neste experimento, variou-se o número de VMs instanciadas em um único Servidor de VMs e Discos (Servidor 1) para analisar o impacto no tráfego da rede. 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, captura-se o tráfego gerado após a criação bem-sucedida de cada VM, ou seja, sem considerar o tráfego relativo ao processo de criação de VMs. Para cada número de VMs instanciadas, o tráfego é medido por 60 s. Esse experimento foi repetido 10 vezes para cada número de VMs. As VMs utilizadas no experimento e no restante deste trabalho possuem o sistema operacional CirrOS, 64 GB de RAM e 1 CPU virtual. Vale notar que as mensagens de controle não variam de acordo com a configuração das VMs e número de CPUs, já que as mensagens enviadas são informações genéricas como estado das VMs, id das VMs, etc. Consequentemente, a configuração das VMs não altera o comportamento observado nos experimentos a seguir.

A Figura 7 ilustra o resultado do experimento. Pode-se observar um comportamento aproximadamente linear, apesar da maior variabilidade do experimento conforme o número de VMs instanciadas aumenta. O ajuste linear dos dados para obter a equação da reta gerou um R^2 igual a 0,9855. Pela equação obtida, estima-se que cada VM gere uma taxa média de 0,77 kb/s. Em um cenário com 100 servidores contendo 15 VMs cada, totalizando 1500 VMs, a carga total gerada pelas VMs seria de 1,155 Mb/s. Neste cenário, somando-se a carga dessas VMs com a dos Servidores de VMs e Discos, analisada anteriormente na Seção 4.1, tem-se uma carga total de aproximadamente 2,7 Mb/s.

4.3. Impacto da criação e destruição de múltiplas VMs

Os experimentos desta seção mostram o comportamento do tráfego durante os processos de criação e de destruição de VMs. No OpenStack, é possível inicializar uma instância a partir de uma imagem armazenada em uma unidade de disco efêmera, que somente armazena os dados enquanto a instância associada existir; ou a partir de uma imagem armazenada em um volume do Cinder, que oferece armazenamento persistente. Quando o usuário solicita a inicialização de 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

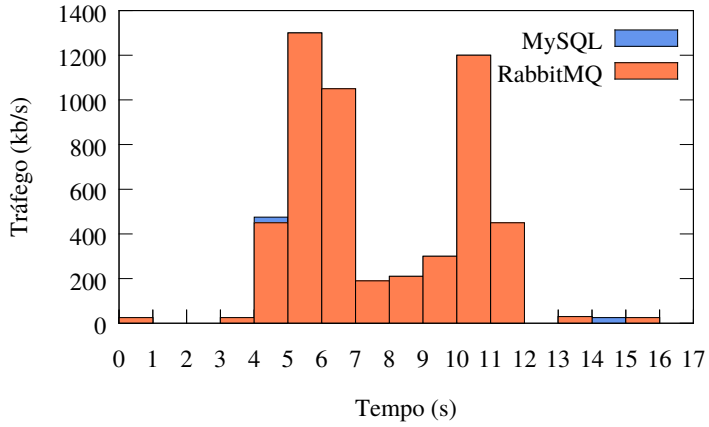


Figura 8. Tráfego gerado pela criação e destruição de uma VM.

efêmera, também armazenada no servidor. 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 enviada novamente pela rede. No caso de instâncias com inicialização a partir de volume, primeiramente um volume vazio é criado, depois a imagem é transferida para o Servidor de VMs e Discos e copiada para o volume gerenciado pelo Cinder. Diferentemente do Nova, o Cinder não possui uma política de cache [Cinder, 2015]. Consequentemente, sempre que uma nova instância com inicialização a partir de volume é criada, a imagem é transferida pela rede.

O caso mais comum de criação de VMs em uma nuvem IaaS é a partir de disco efêmero (isto é, baseada em imagem), devido ao alto tráfego gerado na criação de VMs por volume. Portanto, os casos de uso do sistema devem ser pensados de forma a restringir a utilização de instâncias inicializadas a partir de volume de forma a não sobrecarregar a rede. Para avaliar apenas as mensagens de controle geradas, nesta seção analisa-se o tráfego de rede durante a criação e a destruição consecutivas de VMs baseadas em imagem. O tráfego foi avaliado após a existência de uma imagem de VM no cache do Glance. Dessa forma, os experimentos não são dependentes da configuração utilizada para as VMs. Primeiramente, é mostrado na Figura 8 o impacto no tráfego da rede quando um usuário realiza uma requisição para criar uma instância e, ao fim desse processo, realiza outra requisição para destruí-la. Note que o tráfego atinge um valor de pico de aproximadamente 1,3 Mb/s durante a criação da VM e 1,2 Mb/s na destruição.

Para analisar o tráfego gerado em mais detalhes, emprega-se um *benchmark* para diferentes situações de criação e destruição de VMs no Servidor 1. Dessa forma, utiliza-se a ferramenta Rally [Rally, 2015], que foi 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 simulam diferentes cargas na nuvem, sendo utilizada em trabalhos relacionados [Gelbukh, 2014, Cisco, 2014]. Neste experimento utiliza-se o cenário `boot-and-delete.json`, um dos mais simples presentes no Rally, para executar experimentos de criação e destruição de VMs e, a partir deles, calcular a média do tráfego gerado. Nesse cenário é definido um número de VMs a serem requisitadas à nuvem. O Rally gera uma requisição para criar uma instância, espera esta ser criada, e depois solicita sua destruição. Na sequência, faz um novo ciclo até atingir o

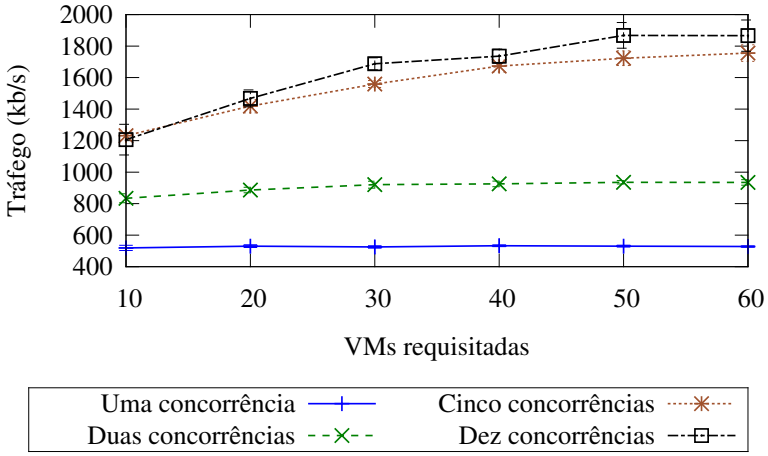


Figura 9. Tráfego gerado nas criações e destruições concorrentes de VMs.

total de instâncias requisitadas. Além disso, o Rally permite configurar a quantidade de requisições paralelas que são enviadas ao Nova API, representando as ações concorrentes de múltiplos usuários da nuvem. Para um valor de concorrência igual a cinco, por exemplo, são emitidas cinco 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 em paralelo. Nos experimentos realizados foram criadas VMs utilizando-se valores de concorrência de 1, 2, 5 e 10. O experimento foi repetido 10 vezes para cada número de VMs requisitadas e concorrências.

A Figura 9 mostra o tráfego gerado. Note que, mesmo para um número pequeno de usuários concorrentes considerado nos experimentos, o tráfego pode ser elevado. Por exemplo, para dez usuários executando o ciclo com dez VMs, o tráfego médio gerado durante o processo é de 1,2 Mb/s. Isso pode representar um alto impacto na rede para uma nuvem de pequeno porte, considerando também o tráfego já gerado continuamente pelas mensagens periódicas, como visto nas Seções 4.1 e 4.2. Ainda na Figura 9, observa-se que a variação de uma concorrência para duas concorrências faz o tráfego médio aproximadamente dobrar. No entanto, o tráfego gerado para cinco concorrências não é cinco vezes maior do que o tráfego com uma concorrência. Além disso, de cinco para dez concorrências, o tráfego praticamente não se altera, pois Servidor de VMs e Discos utilizado não suporta a criação paralela de todas as VMs requisitadas.

5. Trabalhos Relacionados

Alguns trabalhos da literatura abordam a escalabilidade de componentes do OpenStack. Gelbukh, em seu trabalho, avalia a escalabilidade de uma versão do OpenStack desenvolvida pela Mirantis, que é uma das maiores contribuidoras para o código-fonte do OpenStack [Gelbukh, 2014]. Mais especificamente, o trabalho avalia a quantidade de requisições simultâneas para a criação de VMs que uma infraestrutura OpenStack consegue atender. Os resultados mostram que a infraestrutura utilizada nesse trabalho consegue servir 250 requisições paralelas de criação de VMs. Além disso, foi possível instanciar 75.000 VMs na infraestrutura estudada. Apesar de os resultados serem dependentes do *hardware* dos servidores e da rede empregada na infraestrutura, os resultados mostram que é possível alcançar um alto nível de escalabilidade no OpenStack.

A Cisco também realiza diversos experimentos para estressar os componentes do OpenStack e determinar, por exemplo, qual é a limitação em relação ao número de servidores de VMs que um único nó controlador suporta e qual o número máximo de VMs que podem ser hospedadas em cada servidor [Cisco, 2014]. Os resultados mostram que o número de servidores de VMs é limitado pelo RabbitMQ, que consiste na solução de comunicação entre o controlador e os servidores detalhada na Seção 3.2. Além disso, os resultados mostram que o tempo necessário para criar uma VM aumenta de acordo com o número de VMs já instanciadas na infraestrutura, mesmo se essas VMs estiverem em repouso. Devido a esse comportamento e a outros fatores apontados no trabalho, o número de VMs que podem ser instanciadas é limitado, mesmo se ainda existir memória RAM disponível para sua criação. O estudo recomenda, empiricamente, utilizar em cada servidor apenas 40% de sua capacidade teórica de hospedagem de VMs, calculada em função da capacidade de memória RAM do servidor.

Apesar de os trabalhos citados anteriormente consistirem em experimentos em massa, estressando diversos componentes do OpenStack, esses não realizam uma análise da sobrecarga de rede gerada por mensagens de controle. Dessa forma, devido à importância dessa análise para nuvens geodistribuídas, este trabalho complementa a literatura, estudando a escalabilidade do OpenStack do ponto de vista da capacidade de rede.

6. Conclusões e Trabalhos Futuros

Este trabalho analisou impacto do tráfego de controle em uma nuvem colaborativa com controlador centralizado, que por sua vez se conecta a servidores de VMs através de uma WAN. Esse cenário reproduz uma configuração típica de uma nuvem colaborativa de pequeno porte. Os experimentos mostraram que, mesmo com as VMs em repouso, mensagens são continuamente trocadas pela WAN entre o controlador e os servidores de VMs. Na arquitetura IaaS considerada, cada comunicação entre um servidor e o controlador gera um tráfego médio de aproximadamente 15 kb/s. Além disso, quanto maior for o número de VMs instanciadas, maior é o tráfego para o controlador, mesmo se as VMs não realizarem tarefas de rede. Isso ocorre pois o servidor de VMs relata periodicamente os estados de suas VMs para o controlador. Os experimentos mostraram que cada VM adicionada gera, em repouso, um tráfego médio de aproximadamente 0,77 kb/s. Apesar de os valores individuais serem baixos, em uma nuvem IaaS de tamanho médio, com 100 servidores e 15 VMs por servidor, esses valores produzirão um tráfego médio, de controle, de 2,7 Mb/s. Esse resultado é agravado quando considera-se o aumento de carga na rede durante atividades de criação e destruição de máquinas virtuais. Por exemplo, a criação e posterior destruição de 10 VMs em paralelo, cada uma por um usuário diferente, gera um tráfego médio de controle de 1,2 Mb/s durante o processo. Assim, conclui-se que o tráfego de controle gerado deve ser absolutamente considerado no projeto de uma nuvem colaborativa geodistribuída, e que este deve ser estimado de acordo com o perfil esperado de utilização de máquinas virtuais pelos usuários.

Como trabalhos futuros, pretende-se a realização de experimentos considerando as VMs em uso, por exemplo, no caso de suas aplicações instaladas gerarem determinado padrão de tráfego na rede. Além disso, é possível estender a análise considerando a existência de outros projetos do OpenStack, como o Neutron (serviço avançado de rede) e o Ceilometer (serviço de monitoramento de recursos), o que aumentará o tráfego de controle. Por fim, como as redes WANs possuem enlaces com alta latência devido às

distâncias geográficas e possíveis períodos de congestionamento, é importante verificar o impacto da latência da rede nos serviços do OpenStack.

Referências

Apache (2015). *Apache CloudStack*. The Apache Software Foundation. http://cloudstack.apache.org - Acessado em Dezembro de 2015.

Barros, A., Brasileiro, F., Farias, G., Germano, F., Nóbrega, M., Ribeiro, A. C., Silva, I. e Teixeira, L. (2015). Using fogbow to federate private clouds. Em *Salão de Ferramentas do XXXIII SBRC*.

Cinder (2015). *Generic image cache functionality*. OpenStack Cinder Team. http://specs.openstack.org/openstack/cinder-specs/specs/liberty/image-volume-cache.html - Acessado em Dezembro de 2015.

Cisco (2014). *OpenStack Havana Scalability Testing*. Cisco Systems. http://www.cisco.com/c/en/us/td/docs/solutions/Enterprise/Data_Center/OpenStack/Scalability/OHS.pdf - Acessado em Dezembro de 2015.

Couto, R. S., Sciammarella, T., Barreto, H. F. S. S. M., Campista, M. E. M., Rubinstein, M. G., Costa, L. H. M. K., Vetter, F. e Marins, A. L. A. (2015). GT-PID: Uma nuvem IaaS universitária geograficamente distribuída. Em *Quinta Conferencia de Directores de Tecnología de Información - TICAL 2015*, p. 1–19. Redclara.

Develder, C., De Leenheer, M., Dhoedt, B., Pickavet, M., Colle, D., De Turck, F. e Demeester, P. (2012). Optical networks for grid and cloud computing applications. *Proceedings of the IEEE*, 100(5):1149–1167.

Endo, P. T., de Almeida Palhares, A. V., Pereira, N. N., Gonçalves, G. E., Sadok, D., Kelner, J., Melander, B. e Mangs, J. (2011). Resource allocation for distributed cloud: concepts and research challenges. *IEEE Networks*, 25(4):42–46.

Gelbukh, O. (2014). *Benchmarking OpenStack at megascale: How we tested Mirantis OpenStack at SoftLayer*. Mirantis. http://www.mirantis.com/blog/benchmarking-openstack-megascale-tested-mirantis-openstack-softlayer/ - Acessado em Dezembro de 2015.

ISO/IEC (2014). *ISO/IEC 19464. Information technology – Advanced Message Queuing Protocol (AMQP) v1.0 specification*. ISO/IEC.

Jennings, B. e Stadler, R. (2015). Resource management in clouds: Survey and research challenges. *Journal of Network and Systems Management*, 23(3):567–619.

Kivity, A., Kamay, Y., Laor, D., Lublin, U. e Liguori, A. (2007). KVM: the linux virtual machine monitor. Em *Linux Symposium*, volume 1, p. 225–230.

OpenStack (2015). *OpenStack - Cloud Software*. OpenStack Foundation. http://www.openstack.org - Acessado em Dezembro de 2015.

RabbitMQ (2015). *RabbitMQ - Messaging that just works*. Pivotal Software. https://www.rabbitmq.com/ - Acessado em Dezembro de 2015.

Rally (2015). *Rally*. Rally Team. http://wiki.openstack.org/wiki/Rally - Acessado em Dezembro de 2015.