Virtualização Metálica do KVM com MAAS

Jackson L. E. P. Adams[1], **Brivaldo A. S. Junior**[1]

[1]Faculdade de Computação – Universidade Federal de Mato Grosso de Sul (UFMS)
Caixa Postal 549 – 79.070-900 – Campo Grande – MS – Brazil

jackson.adams@aluno.ufms.br, brivaldo@facom.ufms.br

Abstract. *Metal As A Service is a dynamic mode of servers provisioning , make deploying services and set-up hardware more pratical for each service. Offering the user convenience and speed for the creation of services is fundamental to a pleasant human-computer environment. With MAAS and Juju, this theory becomes real and this experience is innovative and surprising. Using the virtual KVM machines which is fully compatible with the MAAS, it is possible to create new services and scale machines.*

Resumo. *Metal As A Service (metal como um serviço) trata-se de um modo dinâmico de aprovisionamento de servidores, tornando prática a instância de serviços e a configuração de hardware para cada serviço. Oferecer ao usuário comodidade e velocidade na criação de serviços é fundamental para um ambiente humano-computador agradável. Com MAAS e Juju, essa teoria se torna real e a experiência homem máquina é inovadora e surpreendente. Por meio de várias máquinas virtuais disponíveis, usufruindo da virtualização KVM que é totalmente compatível com o MAAS, é possível criar serviços e escalar máquinas à vontade, conforme a necessidade.*

1. Introdução

A maneira como os serviços são implementados em tecnologias de nuvem hoje, entregam para o usuário final um modelo complexo de Infraestrutura como Serviço (IaaS). Modelos como Amazon EC2 [Amazon Web Services 2014], OpenStack [Rackspace Cloud Computing 2014] e OpenNebula [OpenNebula Project 2014], virtualizam máquinas e disponibilizam nós virtuais aos usuários. Esse modelo, embora escalável e robusto, é complexo e pouco atrativo para os usuários leigos.

A Canonical (empresa criadora do Ubuntu Linux), criou uma nova abstração para disponibilizar a infraestrutura como serviço. A arquitetura ganhou o nome de MAAS [Ubuntu 2014] (*Metal As A Service*), um modelo que transforma vários servidores nós em uma grande nuvem capaz de executar aplicações. O MAAS é capaz de controlar esses servidores de forma dinâmica para prover um ambiente escalável para aplicações.

A motivação inicial da Canonical ao criar o MAAS era utilizá-lo para simplificar a instanciação do Openstack, uma das ferramentas mais populares para a criação de nuvens públicas e privadas. O Openstack é capaz de gerenciar redes, *clusters*, volumes e VMs (*Virtual Machines*). Contudo, durante esse processo, alguns outros projetos surgiram, como o Juju Charms [Juju 2014a], e tornaram o modelo de uso do MAAS mais prático e dinâmico, mesmo sem o Openstack.

Juju é uma ferramenta criada pela Canonical Ltd. que permite configurar, gerenciar, instanciar e escalar aplicações em um ambiente MAAS, Amazon EC2, OpenStack ou Microsoft Azure. Com o Juju é possível utilizar mais de um ambiente e fazer a migração de aplicações entre os ambientes. A figura 1 ilustra como o Juju aloca servidores disponíveis do MAAS.

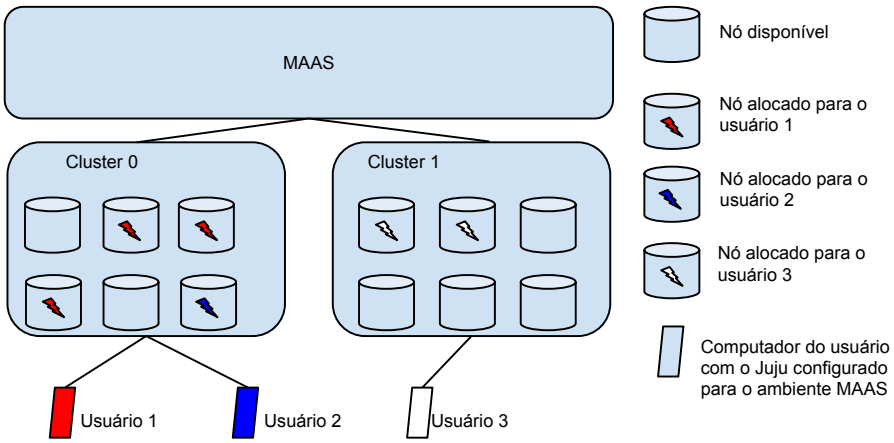

Figura 1. Ilustração do MAAS controlando dois *clusters* com vários nós adicionados e usuários conectados alocando máquinas do MAAS por meio do Juju.

O problema do MAAS é que ele transforma cada servidor da nuvem em apenas um único nó. Esse modelo desperdiça o potencial de cada servidor, pois cada nó dentro do MAAS executa um único serviço (por padrão) despachado pelo Juju. Servidores com múltiplos cores, memória e disco podem acabar rodando apenas uma instância simples de uma página com Wordpress, por exemplo.

O objetivo deste trabalho é melhorar o uso de cada servidor de rede no ambiente MAAS e Juju, instanciando múltiplas máquinas virtuais pré-alocadas. Dessa forma, cada serviço pode ser alocado em uma máquina virtual pequena e que atenda ao seus requisitos de uso. Uma das vantagens da implementação realizada é que o Juju é capaz de paralelizar os serviços instanciando novos nós (que serão mais nós virtuais nos servidores reais).

A virtualização KMV [Kernel Based Virtual Machine 2014] foi utilizada para lidar com as VMs. É uma solução completa de virtualização para linux que utiliza um módulo do kernel para virtualizar uma infraestrutura e o hipervisor QEMU (*Quick Emulator*), suportado pelo MAAS. O KVM suporta as tecnologias de virtualização para os procesadores da Intel (Intel-VT) e AMD (AMD-V).

O trabalho está organizado da seguinte maneira. A Seção 2 apresenta trabalhos relacionados e técnicas conceituadas de implementação de nuvens. A Seção 3 detalha a implementação da nuvem e o funcionamento do Juju integrado ao MAAS com a virtualização KVM em cada servidor de rede. A Seção 4 avalia os modelos de uso e interação dos usuários com o ambiente. Por último, a Seção 5 apresenta as conclusões e trabalhos futuros.

2. Trabalhos Relacionados

Nesta Seção, serão apresentados alguns modelos de nuvem muito utilizados atualmente (Amazon EC2 e OpenStack) e que, de certo modo, concorrem com o MAAS e Juju, pois

apesar de não terem exatamente a mesma proposta, todos entregam recursos de servidores físicos ou virtuais para o usuário.

O Amazon Elastic Compute Cloud (Amazon EC2) [Amazon Web Services 2014] é um serviço na Web pago que fornece infraestrutura computacional e permite escalar recursos na nuvem, ou seja, os desenvolvedores podem aumentar ou diminuir a capacidade com facilidade e conforme a necessidade da aplicação.

Neste modelo de nuvem, o cliente executa instâncias do Amazon EC2, que são máquinas virtuais para o sistema da Amazon, também conhecidas por AMI (*Amazon Machine Images*). O sistema da Amazon promete ser rápido e confiável, contudo, sozinho ele não oferece tanta comodidade na implantação de serviços para o cliente uma vez que ele apenas entrega os servidores virtuais.

Por outro lado, uma ferramenta de código aberto que vem se destacando na criação e gerenciamento de nuvens é o OpenStack [Rackspace Cloud Computing 2014], uma plataforma capaz de gerenciar os componentes de múltiplas infraestruturas virtualizadas, assim como um sistema operacional gerencia os componentes de um computador. Essa plataforma fornece APIs, que em conjunto, são capazes de controlar todos os recursos disponíveis, como VMs, rede e armazenamento.

O OpenStack oferece ferramentas necessárias para que o usuário monte uma nuvem confiável e robusta. É possível controlar cada instância de máquina virtual, prover rede como serviço, fornecer imagens de sistemas virtuais, controle de acesso às VMs, controle do armazenamento online de forma anônima aos clientes, manter estatísticas de contas e manter o sistema consistente em caso de falhas. Uma das melhores maneiras de se instalar o OpenStack é usando o MAAS e o Juju, mas o problema é o número de servidores físicos necessários para isso. Seguindo este trabalho, apesar de possível, não é aconselhável implantar o OpenStack porque ele forneceria uma virtualização da virtualização.

Um grupo da Universidade Chinesa de Hong Kong [H. Chan, M. Xu, P.P.C. Lee e T. Wong 2012] criou uma nuvem experimental usando o OpenStack, fornecendo VMs para que os estudantes pudessem desenvolver aplicações em nuvem e aprendessem a gerenciar uma nuvem de teste, desempenhando o papel de um administrador de sistema. Eles utilizaram o MyCloudLab, um sistema de gestão interativo, baseado na Web, que separa os diferentes grupos de VMs e funcionalidades de cada grupo, assim os alunos só tem privilégios para administrar as VMs que lhes são atribuídas. Por meio de seus componentes, o MyCloudLab facilita o uso e criação de clusters e VMS e possui uma interface simplificada. Assim como o MAAS, essa ferramenta permite que vários usuários possam realizar trabalhos no mesmo cluster.

Tabba e Medouri [Y. Tabaa e A. Medouri 2012] propuseram uma SPC (*Scientific Private Cloud*), que permite o uso de frameworks[1], no intuito de realizar cálculos e processamento de dados em grande escala com MapReduce [J. Dean e S. Ghemawat 2008] e ao mesmo tempo evitar problemas com algoritmos iterativos. A SPC foi testada e avaliada em três configurações distintas, sendo duas delas utilizando frameworks. Con-

[1]Framework é uma ferramenta de desenvolvimento composta de códigos que provêm uma funcionalidade genérica

tudo, a instalação dos frameworks testados poderia ser facilitada por meio do Juju, pois o Spark[2] [The Apache Software Foundation 2014], por exemplo, é um dos vários serviços disponíveis para implantação utilizando o Juju.

3. Implementação

Na documentação do MAAS é recomendado que os serviços de DHCP (*Dynamic Host Configuration Protocol*) e de DNS (*Domain Name System*) fiquem concentrados no próprio MAAS. Além disso, para evitar conflitos com outros serviços, o ambiente da nuvem precisa estar em um segmento de rede dedicado. A solução para o problema foi utilizar uma VLAN (*Virtual LAN*) que liga a rede principal com acesso à Internet a uma rede exclusiva para esse ambiente, assim a rede principal se manteve intacta.

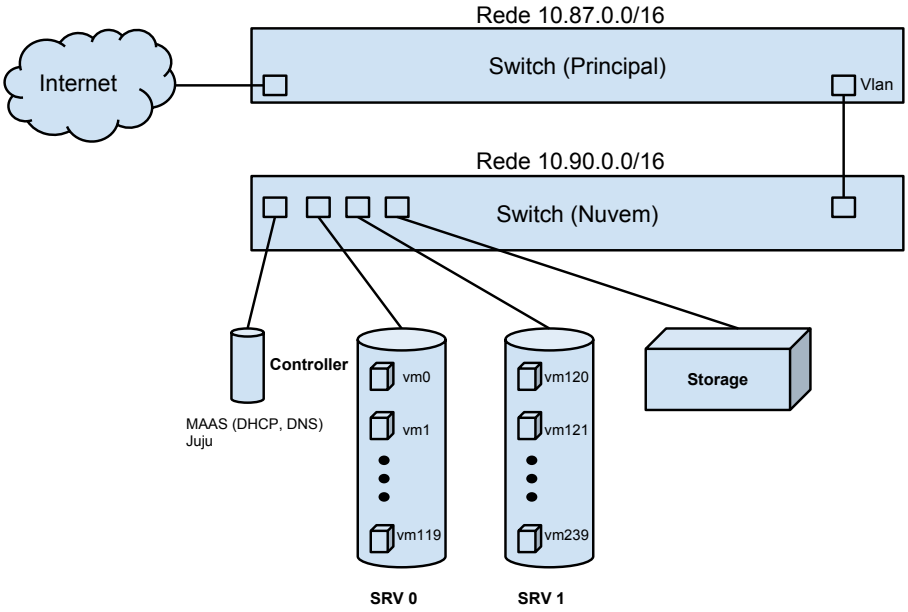

Figura 2. Ilustração da Insfraestrutura da rede com destaque nos componetes da Nuvem criada.

Os experimentos foram realizados usando um servidor de controle com 4 GB de memória e processador Intel Core 2 Duo de 2.66 GHz, dois servidores com 128 GB de memória e dois processadores Intel Xeon E5-2620 de 2 GHz cada e um storage com quatro discos de 2 TB ligados ao *switch* da nuvem. Os três servidores descritos utilizam o Ubuntu Server 14.04 LTS como sistema operacional (preparado para uso integrado com o MAAS) (Figura 2).

No servidor de controle foram instalados o MAAS e seus serviços de DHCP e DNS[3]. Ao término da instalação dos pacotes é necessário criar uma senha de administrador para o acesso à ferramenta[4]. Além disso, para o que os nós do MAAS funcionem, é fundamental que ao menos uma imagem do Ubuntu esteja disponível para boot

[2]O Apache Spark foi um dos frameworks utilizados por Tabba e Medouri em seus testes

[3]Comando para instalação do MAAS e seus serviços: `sudo apt-get install maas maas-dhcp maas-dns`

[4]Comando necessário para criar a senha de administrador: `sudo maas-region-admin createadmin --username=root --email=MYEMAIL@EXAMPLE.COM`

nas máquinas nós. Portanto, para que exista essa imagem inicial é preciso realizar a importação das imagens de boot. Essa fase da instalação exige conexão com a Internet para executar o *download* das imagens e pode ser executada pelo editor de *clusters* do MAAS.

Em seguida é necessário configurar o controlador de *cluster* para gerenciar DHCP e DNS. Dados como o endereço IP do MAAS, a máscara de subrede, endereço de *broadcast*, *gateway* e a faixa de endereços IP da subrede sob controle do MAAS são fundamentais neste processo. Essa etapa foi realizada usando o editor de *clusters* do MAAS pela interface Web.

Uma vez que a configuração do controlador do MAAS foi realizada, os nós podem ser adicionados e inicializados via PXE (*Preboot Execution Environment*). O processo pode ser executado de duas maneiras. A primeira é a descoberta automática, em que ao ligar a máquina configurada para iniciar (*boot*) via rede, ela vai procurar um servidor DHCP, receber as informações padrões do boot, iniciar e contactar o servidor MAAS e desligar. A outra maneira é a adição manual dos nós, em que devem ser passadas informações da máquina, como *Wake-on-LAN* (tipo de fonte de energia para boot na rede) e endereço MAC e declarar qual imagem de sistema, cluster e zona do MAAS será utilizada para o nó. Esse processo é conhecido como "declaração" (*Declaring*) dos nós no MAAS.

Para o melhor aproveitamento dos recursos dos dois servidores disponíveis, optou-se por utilizar máquinas virtuais, assim é possível disponibilizar muito mais que apenas dois nós a serem alocados pelo MAAS. Dessa forma, cada serviço pode ser alocado em uma VM pequena e ao mesmo tempo suficiente para seu uso. Nesses servidores, foram instalados o Virtinst 0.6 junto ao Libvirt 1.2 para criar as máquinas virtuais KVM. Além disso, foi necessário a configuração de uma *bridge* na interface de rede dos servidores para que as VMs fiquem na mesma subrede e recebam IP e DNS do controlador [Server World 2014].

Durante o planejamento da criação das inúmeras VMs por servidor, para que os recursos fossem utilizados ao máximo, o *storage* de dados teve que ser anexado à rede, atendendo às necessidades de armazenamento das VMs. Para que a comunicação entre o *storage* e os servidores não fosse comprometida, foi preciso utilizar o OCFS2 (*Oracle Cluster File System*) [Oracle 2014] no armazenamento do *storage*, um sistema de arquivos clusterizado que permite que vários computadores possam ler e escrever no disco concorrentemente, o que não é possível com sistemas de arquivos nativos do Linux como o Ext4. Os servidores foram conectados ao *storage* por intermédio da tecnologia iSCSI (*Internet Small Computer System Interface*) via conexão TCP/IP.

A declaração das VMs como nós é feita manualmente e para que uma VM possa ser controlada pelo MAAS, é fundamental criar e exportar uma chave SSH pública do usuário do sistema chamado **maas** para os servidores hospedeiros das VMs. Para isso, deve-se habilitar o *login*[5], se conectar como **maas**[6] e executar os comandos para a cópia

[5]Comando para adicionar o login via shell do usuário maas: `sudo chsh -s /bin/bash maas`
[6]Realizar login como usuário **maas**: `sudo su - maas`

das chaves[7].

Durante a adição de um nó, deve-se setar a opção "virsh (virtual systems)" como tipo de fonte de energia (*power type*), adicionar os parâmetros da conexão *libvirt* (contendo o tipo de conexão, usuário e *IP* do servidor físico conforme a chave SSH adicionada e um ID para o nó virtual) e confirmar a adição. Ao criar e iniciar uma VM pela rede, ela se registra como um nó com um *hostname* aleatório, mas sua declaração não fica completa como no modo automático. Os dados citados devem ser completados por meio do gerenciador de nós do MAAS para finalizar o processo.

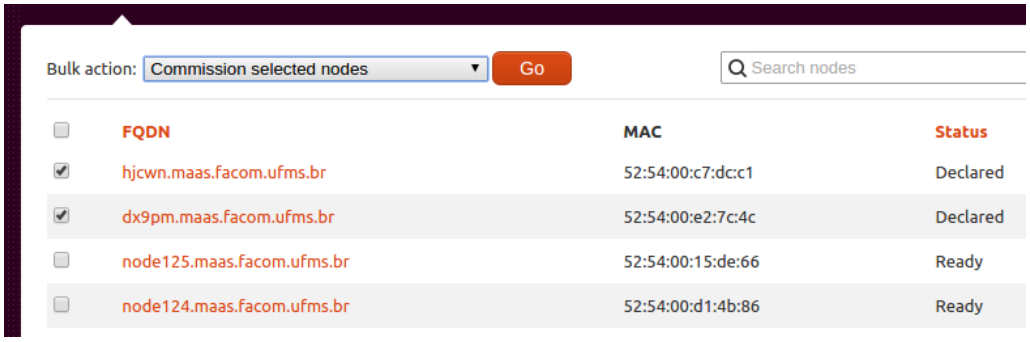

Figura 3. Tela do gerenciador de nós do MAAS. Os nós selecionados são exemplos de nós adicionados com nomes aleatórios.

Após declarar um nó virtual por completo, o MAAS deve ser capaz de controlar o estado da VM sozinho. Todo nó, depois de declarado, deve ser comissionado (*Commissioning*), processo em que a imagem do Ubuntu configurada é instalada no nó correspondente e então a máquina nó é desligada. Para iniciar o comissionamento pela interface Web, basta selecionar um ou mais nós declarados, marcar *"Commission selected nodes"* e clicar em *"Go"* no gerenciador de nós (Figura 3). Depois de comissionado, o nó é considerado pronto para ser alocado (*Ready*).

Afim de facilitar a adição dos nós, foi criado um *script* em bash para executar no servidor do MAAS, esse *script* cria uma VM em um dos servidores (SRV 0 ou SRV 1), adiciona, edita e comissiona o nó referente a essa VM. Na criação das VMs, foram utilizados parâmetros como tipo de virtualização, quantidade de memória RAM, nome da VM, local de armazenamento, tipo de inicialização e número de *CPUs*. Ao criar uma VM pelo pelo comando *virt-install*, ela é inicializada imediatamente pela rede, iniciando o processo de declaração. Em seguida, o script atualiza o nó, completando os dados necessários, alterando o *hostname* para o formato "nodeXXX" (XXX representa o número da VM criada) e então inicia o processo de comissionamento. Para que o *script* funcione, a API do MAAS deve ser ativada para o MAAS poder aceitar os comandos do usuário. Realizar o primeiro login na API é o suficiente para ativá-la, basta obter uma chave do MAAS (*MAAS Key*) no editor de preferências via Web (Figura 4) e executar um comando pelo terminal[8].

[7]Criar chave SSH pública : `ssh-keygen`. Exportar a chave SSH pública : `ssh-copy-id root@<ip-host-vms>`

[8]Comando para realizar primeiro Login na API do MAAS: `maas login root http://<ip-maas>/MAAS/api/1.0 <chave do MAAS>`

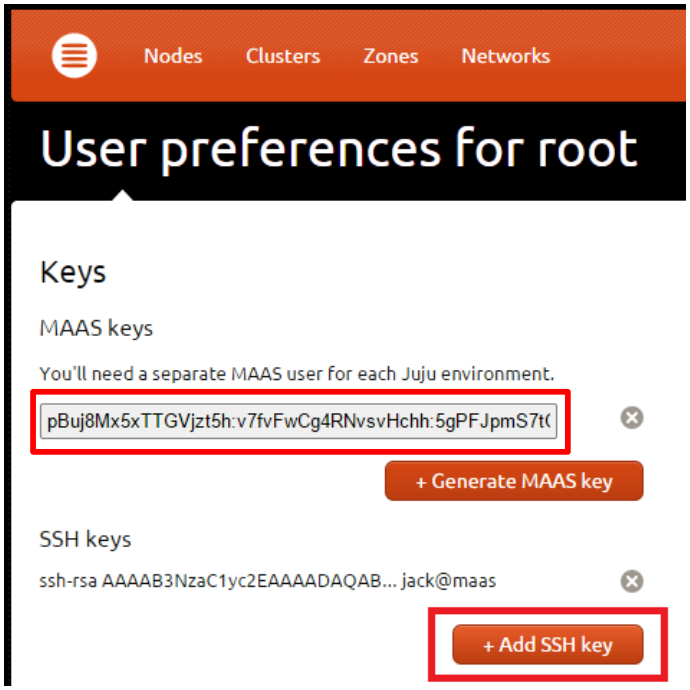

Figura 4. Tela do editor de preferências do MAAS. Em destaque, a chave do MAAS para realizar o login na API e o botão para adicionar a chave SSH pública do usuário do Juju.

Posto que os nós foram devidamente comissionados, inicia-se a instalação e configuração do Juju [Juju 2014b]. No ambiente de testes o Juju foi instalado no controlador do MAAS[9]. Contudo, na prática, cada usuário da nuvem deve instalá-lo e configurar as chaves SSH da API do MAAS localmente. Cada usuário deve ter sua própria chave na API e isso é feito no MAAS. Com finalidade de utilizar o Juju para executar serviços nos nós, foi imprescindível criar e copiar uma chave SSH pública do usuário do Juju para o MAAS utilizando o editor de preferências via Web (Figura 4).

Com o propósito de iniciar o ambiente do Juju foi essencial realizar o processo de *bootstrap*[10]. Esse processo é uma instância da nuvem que o Juju usa para implantar e controlar os serviços. Com os nós prontos e o ambiente do Juju configurado, os serviços estão prontos para serem lançados. Por meio de um terminal conectado ao servidor do Juju, um serviço é implantado com apenas uma linha de comando[11].

Ao iniciar um serviço com o Juju, automaticamente o MAAS reconhece o nó que está sendo utilizado e o marca como uma máquina para o Juju, assim um novo serviço utilizará outro nó disponível. Caso haja a necessidade de adicionar um serviço a um nó em uso, é fundamental que o novo serviço não utilize portas ocupadas e basta especificar qual máquina que o novo serviço ocupará. Por exemplo, é possível adicionar um servidor MySQL a um nó que já esteja executando o Wordpress[12].

[9]Instalação do Juju: `apt-get install juju-core`
[10]Bootstrap do Juju: `juju bootstrap`
[11]Exemplo: implantar o MySQL: `juju deploy mysql`
[12]Comando para implantar o MySQL em uma máquina específica: `juju deploy --to 0 mysql`. Observe que o Juju organiza as máquinas em uma numeração crescente a partir do 0

Após instanciar os serviços, é necessário marcar quais serviços serão acessíveis na rede e isso também é feito pelo Juju. Assim o serviço é configurado para responder as requisições de visitantes. Por exemplo, o serviço Wordpress, só torna-se acessível pela Web após ser liberado pelo Juju[13].

4. Modelos de Uso

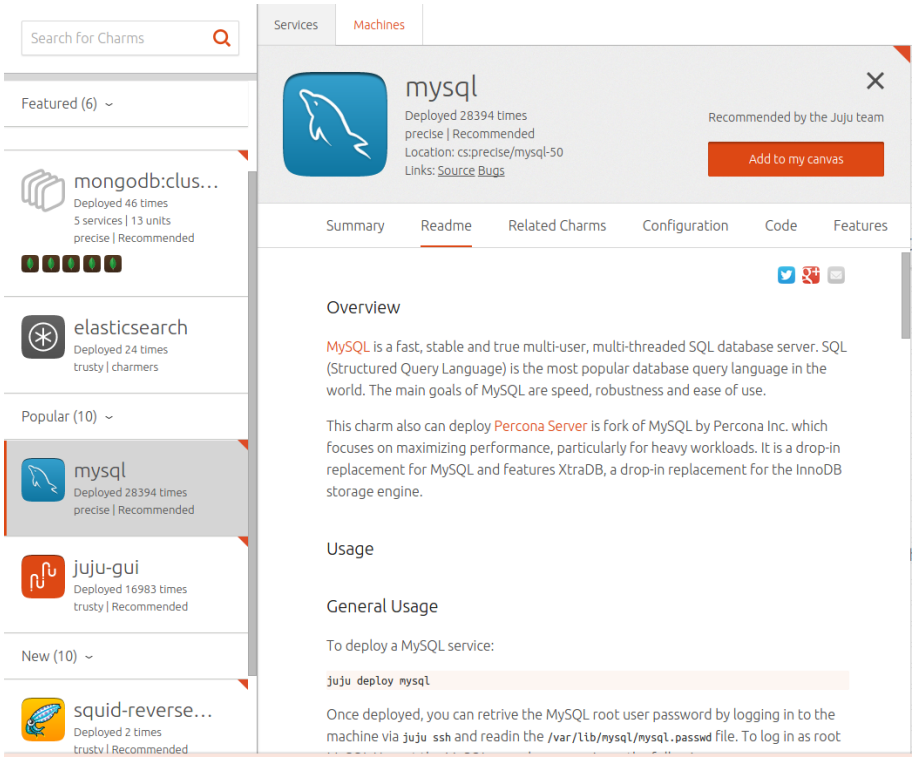

Figura 5. Exemplo: Visualizando *charm* do MySQL Server 5.0 para Ubuntu precise (12.04)

Cada serviço pronto para ser implantado no Juju é chamado de *charm*. Os usuários do Juju podem tanto utilizar os *charms* já disponíveis quanto *charms* personalizados que podem ser criados por meio de ferramentas específicas (*charm tools*) [Juju 2014c]. Felizmente há uma grande variedade de *charms* acessíveis para as duas ultimas versões do Ubuntu LTS.

Para utilizar a interface gráfica do Juju via Web, é preciso instalar o *charm* `juju-gui`, expô-lo e acessá-lo pela URL do nó alocado. *Charms* podem ser procurados utilizando a barra de busca na interface, que os organiza de acordo com o tipo do serviço.

No sistema Web, ao selecionar um *charm* para uso, o usuário tem acesso às informaçoes de versão, resumo, modo de uso, *charms* relacionados e configurações do serviço. Para implantá-lo basta clicar em *Add to my canvas* (Figura 5). Para um serviço funcionar corretamente, é necessário que exista pelo menos um nó disponível com a mesma versão do Ubuntu para qual o *charm* foi criado. Por exemplo, se foram adici-

[13]Exemplo: expor o serviço wordpress: `juju expose wordpress`

onadas somente máquinas com a versão *trusty* (14.04) do Ubuntu, *charms* para a versão *precise* (12.04) não devem operar regularmente.

O *canvas* é a área onde são mostrados os serviços implantados no Juju e suas respectivas máquinas. Nessa área é possível mover os serviços na tela para organizá-los. Além disso, alterações na configuração, exposição, criação de relacionamentos e adição de máquinas para um serviço também podem ser realizadas no *canvas* e no painel lateral (Figura 6).

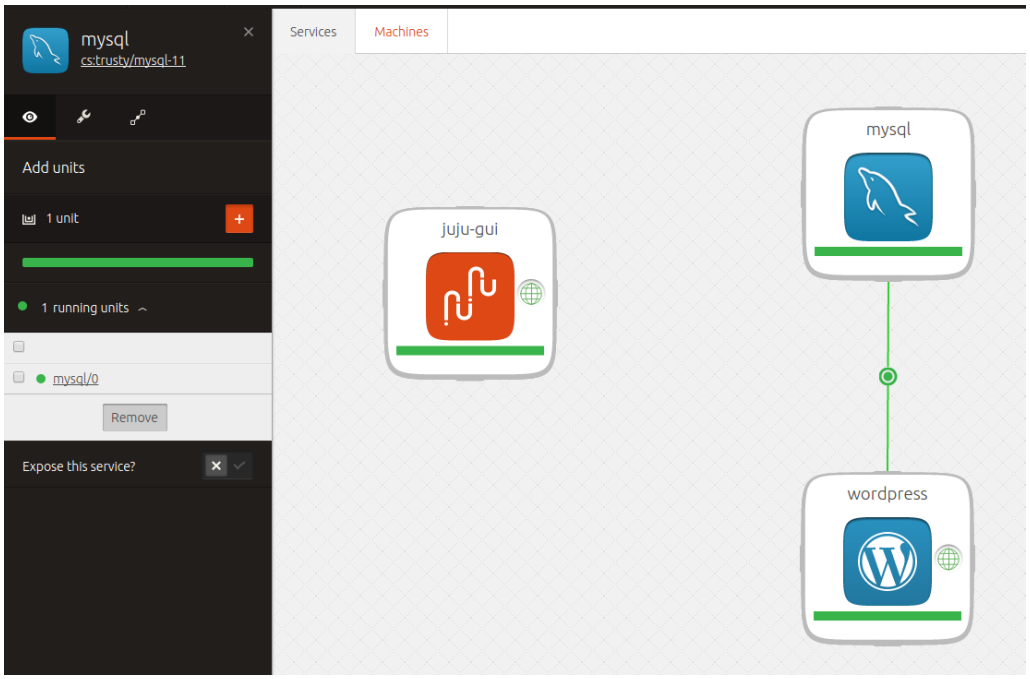

Figura 6. Painel lateral e *canvas* no Juju-Gui: Permitem a manipulação e configuração dos serviços do Juju via Web

Com um comando no terminal[14] pode-se obter várias informações das máquinas em uso pelo Juju, como o nome (*dns-name*), versão do sistema (*series*) e estado (*agent-state*). Além disso, são mostrados detalhes dos serviços presentes: versão do *charm*, em qual máquina está executando e se está exposto. Essas informações também estão acessíveis na interface gráfica do Juju (`juju-gui`).

5. Conclusão

Neste trabalho, foi criada uma nuvem de testes utilizando o MAAS, Juju e virtualização com KVM, mostrando os detalhes da implementação e os modelos de uso. Assim como as núvens já existentes, foram utilizadas máquinas virtuais. Entretanto, a nuvem criada mostrou-se muito diferente das demais, uma vez que ocorre a entrega direta de serviços ao invés da entrega de apenas um servidor virtual.

Ao invés de fazer uso de somente servidores físicos, como a Canonical sugere, ao lidar com máquinas virtuais que aproveitam ao máximo dos servidores reais para adicionar ao MAAS, foi obtido um excelente custo-benefício e alta disponibilidade de nós,

[14]Listar máquinas e serviços do Juju pelo terminal: `juju status`

melhorando o uso de cada servidor de rede. Embora haja uma certa complexidade para realizar todos os passos da implementação, a simplicidade no uso da nuvem compensa o trabalho.

5.1. Trabalhos Futuros

Planeja-se montar um ambiente MAAS e Juju na Faculdade de Computação da UFMS para que os alunos e professores possam usufruir da nuvem, testando e executando os serviços que precisarem. Para isso, devem ser estudadas algumas funcionalidades do MAAS que não foram exploradas neste trabalho, como criação de grupos, permissões, lidar com múltiplos clusters e múltiplas redes.

A criação de novos *charms* do Juju para a comunidade também deve ser feita. Deverão ser testados serviços que ainda não compõem a loja de *charms* (*Charm store*) e futuramente submetê-los para aprovação.

Referências

[Amazon Web Services 2014] Amazon Web Services (2014). Amazon EC2. In *aws.amazon.com/pt/ec2/*. Amazon. [Online; acessado 22-Set-2014].

[H. Chan, M. Xu, P.P.C. Lee e T. Wong 2012] H. Chan, M. Xu, P.P.C. Lee e T. Wong (2012). MyCloudLab: An Interactive Web-based Management System for Cloud Computing Administration. In *Teaching and Learning Innovation Expo*. TLIE'12.

[J. Dean e S. Ghemawat 2008] J. Dean e S. Ghemawat (2008). MapReduce: simplified data processing on large clusters. In *Magazine Communications of the ACM*. CACM'08.

[Juju 2014a] Juju (2014a). Juju Charms. In *jujucharms.com*. Canonical. [Online; acessado 20-Out-2014].

[Juju 2014b] Juju (2014b). Juju Documentation. In *https://juju.ubuntu.com/docs/*. Canonical. [Online; acessado 09-Out-2014].

[Juju 2014c] Juju (2014c). Writing a Charm. In *juju.ubuntu.com/docs/authors-charm-writing.html*. Canonical. [Online; acessado 20-Out-2014].

[Kernel Based Virtual Machine 2014] Kernel Based Virtual Machine (2014). Kernel Based Virtual Machine. In *www.linux-kvm.org*. MediaWiki. [Online; acessado 22-Set-2014].

[OpenNebula Project 2014] OpenNebula Project (2014). OpenNebula. In *opennebula.org*. OpenNebula. [Online; acessado 22-Set-2014].

[Oracle 2014] Oracle (2014). OCFS2 - Oracle Cluster File System for Linux. In *www.oracle.com/us/technologies/linux/025995.htm*. Oracle. [Online; acessado 13-Out-2014].

[Rackspace Cloud Computing 2014] Rackspace Cloud Computing (2014). OpenStack. In *openstack.org*. Rackspace. [Online; acessado 12-Ago-2014].

[Server World 2014] Server World (2014). Install KVM. In *www.server-world.info/en/note?os=Ubuntu_14.04&p=kvm*. Server World. [Online; 21-Abr-2014].

[The Apache Software Foundation 2014] The Apache Software Foundation (2014). Apache Spark. In *spark.apache.org*. Apache. [Online; acessado 22-Mai-2014].

[Ubuntu 2014] Ubuntu (2014). MAAS - Metal As A Service. In *maas.ubuntu.com*. Canonical. [Online; acessado 12-Ago-2014].

[Y. Tabaa e A. Medouri 2012] Y. Tabaa e A. Medouri (2012). Towards a Next Generation of Scientific Computing in the Cloud. In *International Journal Computer Science Issue 6, No 03*. IJCSI'12.