UNIVERSIDADE FEDERAL DE SANTA CATARINA
DEPARTAMENTO DE INFORMÁTICA E ESTATÍSTICA

Alec Augusto Gonçalves Ventura

Integração do PCMONS com o OpenStack para Gerência e Monitoramento de Nuvens Privadas

Florianópolis/SC
2012

Alec Augusto Gonçalves Ventura

Instalação e monitoramento de uma Cloud privada usando o OpenStack

Proposta de trabalho de conclusão de curso apresentado como parte dos requisitos para obtenção do grau de bacharel em Ciência da Computação pela Universidade Federal de Santa Catarina.

Orientador
Dr. Carlos Becker Westphall

Florianópolis,
2012

Agradecimentos

Gostaria de agradecer em especial aos meus pais, Luiz Roberto Leal Ventura e Izabel Lemes Gonçalves por me fornecerem todo o apoio e estrutura necessárias que me permitiram concluir o ensino superior.

A minha companheira Jessica Souza Santos pela compreensão e carinho durante a realização deste trabalho.

E a toda a equipe do Laboratório de Rede e Gerência que me garantiu toda infraestrutura e ajuda para a realização deste trabalho com especial ao meu professor e orientador Carlos Becker Westphall pelo encorajamento e orientação sempre estando pronto a me ajudar.

Sumário

Lista de Figuras

Lista de Tabelas

Resumo

Computação em nuvem ou *cloud computing* é um termo que tem interessado muito especialistas, estudantes e profissionais da área de TI. Representa um novo modelo de como utilizamos os recursos dos computadores, mais especificamente dos nossos servidores. Esse modelo é usado cada vez mais e tem ganho interesse e confiança por parte de usuários e empresas, mostrando ter um alto potencial mercadológico. Mas mesmo com esse avanço ainda existe uma falta de informação e ferramentas fáceis, confiáveis e gratuitas o que faz com que muitos usuários deixem de migrar para esse novo modelo.

Esse projeto tem como foco principal a instalação e monitoramento de uma *cloud* privada utilizando o OpenStack e a medição do desempenho dessa nova ferramenta em desenvolvimento utilizando o PCMONS (*Private Cloud Monitoring System*) sendo necessário a integração entre ambos.

Para teste, validação da integração dos componentes e apresentação dos resultados obtidos será implantando no Laboratório de Redes e Gerência (LRG) da Universidade Federal de Santa Catarina um modelo de nuvem privada utilizando o OpenStack monitorando-se as máquinas virtuais utilizando-se o PCMONS.

Palavras-chave

Computação em nuvem, OpenStack, PCMONS.

Abstract

Cloud computing is a term that has been very interested specialists, students and IT professionals, it represents a new model of how to use computer resources, more specifically our servers. This model is increasingly used and has gained interest and trust by consumers and businesses, showing that a high market potential. But even with this progress there is still a lack of information and tools that are easy, reliable and free what makes many users fail to migrate to this new model.

This project focuses primarily on the installation and monitoring of a private cloud using OpenStack and measure the performance of this new tool under development using PCMONS (Private Cloud Monitoring System) is necessary integration between the two.

To test, validate the integration of components and presentation of results, will be deploying at the Network and Management Laboratory (LRG) at Federal University of Santa Catarina a private cloud model using OpenStack by monitoring virtual machines using PCMONS.

Keywords

Cloud computing, OpenStack, PCMONS.

Lista de Siglas:

PCMONS	Private Cloud Monitoring System ou Sistema de Monitoramento para Nuvens Privadas
IaaS	Infrastructure as a Service ou Infraestrutura como Serviço
PaaS	Platform as a Service ou Plataforma como Serviço
SaaS	Software as a Service ou Software como Serviço
EaaS	Everything as a Service ou Tudo como Serviço
HaaS	Hardware as a Service ou Hardware como Serviço
NIST	National Institute Standards and Technology ou Instituto Nacional de Tecnologia e Padrões
IT	Information Technology ou Tecnologia da Informação
API	Application Programming Interface ou Interface de Programação de Aplicativos
KVM	Kernel-based Virtual Machine ou Máquina Virtual baseada em Kernel
GNU	GNU Not Unix
IP	Internet Protocol ou Protocolo de Internet
XML	eXtensive Markup Language
RPC	Remote Procedure Call ou Chamada Remota de Procedimento
S3	Simple Storage Service ou Serviço de Armazenamento Simples
AWS	Amazon Web Services ou Serviços Web Amazon
SLA	Service Level Agreement ou Acordo de Nível de Serviço
LVM	Logical Volume Manager ou Controlador de Volume Lógico

Capítulo 1

Introdução

No dia a dia utilizamos serviços básicos como água, eletricidade e telefone, e pagamos por esses recursos conforme utilizamos, ou seja, pagamos conforme nossas demandas pessoais ou empresariais. A computação em nuvem utiliza o mesmo conceito, fazendo com que o usuário pague pelos serviços de computação que ele utiliza.

A maior vantagem nesse modelo é disponibilidade dos recursos já que em um ambiente onde temos um servidor local, o desempenho dos serviços são limitados pela capacidade do servidor que rapidamente se torna obsoleto, já num ambiente onde se utiliza a computação em nuvem se for necessário maior alocação de recursos esses são disponibilizados dinamicamente, ou seja, sob demanda e são limitados apenas pela capacidade de um grupo de servidores com um potencial muito superior a um servidor local.

Existe ainda uma grande desconfiança por parte de grandes empresas e usuários relacionado com a cobrança correta desses recursos utilizados, e principalmente, sobre a segurança de seus dados, já que eles estariam guardados em servidores remotos, localizados muitas vezes em outros países.

Como todo paradigma novo, a computação em nuvem ainda precisa conquistar a confiança e estabelecer conceitos sólidos sobre seu modelo para a melhoria de suas ferramentas.

Empresas globais como Amazon, Google e Microsoft estão investindo em pesquisas e ferramentas para saírem na frente o que mostra um grande potencial de mercado.

Assim, o presente trabalho tem como objetivo a criação e implementação de uma rede privada utilizando o OpenStack que é uma nova ferramenta open-source, para a análise de seu desempenho.

1.1 Motivação

A computação em nuvem está expandindo rapidamente e conquistando a confiança de pequenos e grandes empresários. Foi desenvolvido pela NASA junto ao Rackspace Hosting uma ferramenta de computação em nuvem que pudesse ser altamente escalável, esta ferramenta recebeu o nome de OpenStack.

Se faz necessário uma ferramenta de monitoramento das novas tecnologias emergentes no mercado para que sejamos capazes de avaliar as melhores ferramentas e seus desempenhos.

A motivação para este trabalho vem da vontade de integrar o PCMONS[CHA 10] ao OpenStack já que esta nova ferramenta de Infraestrutura como Serviço (IaaS) carece deste tipo de funcionalidade. E também de trabalhar em uma ferramenta de monitoramento que aos poucos vem ganhando espaço na comunidade acadêmica com a implementação de novos módulos sendo desenvolvidos por graduandos, mestrandos e doutorandos.

1.2 Objetivo geral

Com o desenvolvimento deste trabalho pretende-se conhecer e aprofundar os conceitos na área de redes de computadores, sistemas distribuídos e computação em nuvem. O objetivo geral é desenvolver a integração do software de monitoramento de cloud, PCMONS[CHA 10], com a ferramenta de Infraestrutura como Serviço (IaaS) OpenStack.

1.3 Objetivos Específicos

Os objetivos específicos deste trabalho são:

- Apresentar os conceitos de tecnologias envolvidas na computação em nuvem;
- Pesquisar ferramentas para implantação de um ambiente de computação em nuvem;
- Pesquisar ferramenta para o monitoramento de nuvens;
- Implantar uma nuvem privada com propósitos acadêmicos no Laboratório de Redes e Gerência (LRG), utilizando um software livre;

- Desenvolver e integrar um módulo para o sistema de monitoramento de nuvens privadas PCMONS; e
- Testar o sistema desenvolvido através de um estudo de caso.

1.4 Método de pesquisa

Será instalado o Openstack em computadores do LRG(Laboratório de Redes e Gerência) localizado no prédio do INE, e será realizado alguns experimentos para se monitorar o desempenho da ferramenta através de monitores fornecidos pela própria ferramenta.

Se o OpenStack não apresentar monitores adequados ou suficientes será adicionado a ele algum dispositivo para se fazer o monitoramento tendo como forte candidato o PCMONS (Private Cloud Monitor System) que é uma ferramenta desenvolvida pelos estudantes Rafael Bruno Uriarte e Shirlei Aparecida de Chaves para efetuar o monitoramento das redes [CHA 10].

1.5 Organização do trabalho

Este trabalho está organizado em seis capítulos, sendo eles:

Capítulo 1 – Introdução – Este capítulo apresenta o trabalho, conceitos sobre o tema, a motivação e o objetivo geral e seus objetivos específicos.

Capítulo 2 - Computação em Nuvem – Apresenta a visão e o conceito da computação em nuvem, mostrando suas classificações em diferentes categorias como em relação ao seu modelo de serviço e modelo de implantação.

Capítulo 3 - Ferramentas de Infraestrutura para Computação em Nuvem – Este capítulo é dedicado a ferramenta de implementação em nuvem na modalidade IaaS que foi utilizada para a implementação deste trabalho, o OpenStack.

Capítulo 4 - Monitoramento em Computação em Nuvem – É apresentado neste capítulo a importância do monitoramento, e duas ferramentas utilizadas nesse trabalho, a ferramenta Nagios que é utilizada como interface para a outra ferramenta de monitoramento que é o PCMONS.

Capítulo 5 – Desenvolvimento – Apresenta os passos utilizados para se implementar o OpenStack como uma nuvem privada no LRG, bem como os passos para a adaptação da ferramenta PCMONS para garantir o suporte ao OpenStack. A implementação é testada e são apresentados os resultados nesse mesmo ambiente em nuvem.

Capítulo 6 – Conclusão – Este capítulo encerra o trabalho com algumas considerações finais.

Capítulo 2

Computação em Nuvem

Segundo a definição do NIST a computação em nuvem permite o acesso à rede sob demanda a um conjunto compartilhado de recursos computacionais configuráveis (redes, servidores, armazenamento, aplicativos e serviços) que podem ser rapidamente provisionados e liberados com o esforço de gerenciamento mínimo ou interação com o provedor de serviços [NIST]. Hoje a computação em nuvem utiliza a Internet como um meio para a distribuição de seus recursos e serviços. Com a evolução de tecnologias como virtualização, computação em grade e computação distribuída a computação em nuvem é capaz de oferecer recursos e serviços com qualidade e segurança rapidamente e com mínimo esforço do provedor de serviço.

Uma definição mais simples seria dizer que a computação em nuvem é o fornecimento de recurso ou serviço sob demanda do cliente. O cliente tem a ilusão de disponibilidade de recursos infinitos diferentemente de se ter um servidor próprio em sua empresa limitado pelo *hardware.* A computação em nuvem permite que empresas usem recursos na quantidade que forem necessários, aumentando ou diminuindo a capacidade computacional dinamicamente.

Bancos de dados, redes, aplicativos, plataformas e até infraestruturas completas são alguns exemplos de recursos computacionais que podem ser fornecidos.

Empresas globais como Microsoft, Amazon, Google e IBM estão investindo milhões em pesquisa e desenvolvimento na área para ficarem a frente do mercado e se tornarem referência em *cloud computing.*

2.1 Classificação quanto ao modelo de serviço

Existem três classificações para os ambientes de computação em nuvem [NIST].

- Software como Serviço (SaaS).
- Plataforma como Serviço (PaaS).
- Infraestrutura como Serviço (IaaS).

2.1.1 Software como Serviço (SaaS)

É um modelo onde softwares de aplicativos são oferecidos por um provedor ou fornecedor de serviço e toda parte de gerenciamento e manutenção é de responsabilidade do provedor, o cliente apenas acessa e utiliza o serviço, sem a necessidade de instalação no cliente.

São características do SaaS:

- Comunicação entre cliente e servidor via rede, normalmente a Internet.
- Atualizações automáticas, sem necessidade de envolvimento do usuário;
- Compatibilidade: todos os clientes utilizarão a mesma versão do aplicativo;
- Acessibilidade global.

O melhor exemplo de SaaS é o Google Apps que engloba um pacote de aplicativos de comunicação e colaboração. No pacote estão incluídos: Gmail[1], Google Calendar[2], Google Talk[3] e Google Docs[4].

[1] http://www.gmail.com

[2] https://www.google.com/calendar

[3] https://www.google.com/talk/

[4] https://docs.google.com/

2.1.2 Plataforma como Serviço (PaaS)

PaaS é um modelo que oferece um conjunto de ferramentas, softwares e middlewares que um desenvolvedor necessita para fazer sua aplicação. O modelo permite que o desenvolvedor acompanhe todas as etapas do desenvolvimento, desde a criação dos casos de uso até a implementação e validação dos testes.

São características do PaaS:

- Cada componente da plataforma é oferecido como serviço;
- Investimento inicial menor e com menos risco, podendo ser expandido sob demanda; e
- Equipes de desenvolvimento geograficamente distribuídas podem trabalhar juntas em projetos de desenvolvimento.

Mesmo com todas as vantagens apresentadas, esse modelo ainda sofre com a desconfiança dos empresários que precisam confiar que a prestadora de serviços garanta a segurança de seus dados pessoais ou corporativos, fato que faz com que muitas empresas ainda prefiram manter seus próprios servidores.

Podemos citar várias empresas globais que oferecem serviços PaaS, como exemplo a Google[5] com o Google App Engine[6], a Microsoft[7] com o Microsoft Azure[8] e a Amazon[9] com o Amazon Elastic Beanstalk[10].

---

[5] http://www.google.com

[6] https://developers.google.com/appengine/

[7] http://www.microsoft.com/pt-br/default.aspx

[8] http://www.windowsazure.com/en-us/

[9] http://www.amazon.com/

[10] http://aws.amazon.com/pt/elasticbeanstalk/

2.1.3 Infraestrutura como Serviço (IaaS)

Nesse modelo é oferecido ao usuário a estrutura completa de hardware, desde processamento, armazenamento e rede. O cliente tem total controle sobre quais sistemas operacionais e tecnologias de virtualização estão instalados sem precisar se preocupar em administrar a infraestrutura.

Esse modelo, referido por alguns autores como Hardware as a Service (HaaS) possui algumas características, entre elas:

- Rápida e fácil mudança da infraestrutura caso seja necessário ampliar a demanda ou diminuí-la; e
- Utilização ótima dos recursos.

A figura 2.1 representa um diagrama de modelos de serviços.

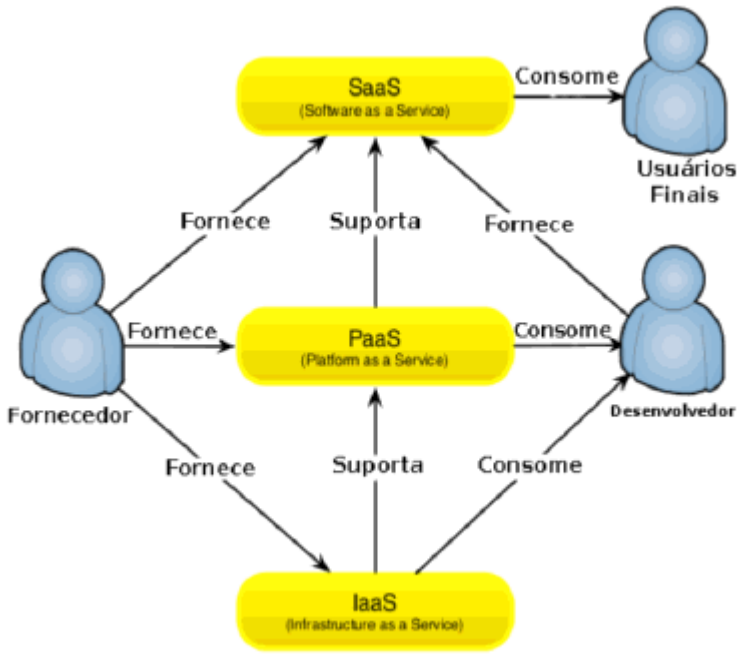


Figura 2.1 - Modelos de serviço em computação em nuvem. Inter-relação entre fornecedores e consumidores. [VIT 12].

2.2 Classificação quanto ao Modelo de Implantação

Além da classificação por modelo de serviço a computação em nuvem pode ser também classificada por modelo de implantação, nuvem pública, nuvem privada, nuvem híbrida e nuvem comunitária.

2.2.1 Nuvem Pública

Neste modelo de nuvem o prestador oferece seu serviço pela Internet, pode ser gratuito ou utilizando um modelo de pagamento baseado no uso (pay-per-use). A infraestrutura da *cloud* é projetada para atender a muitos usuários onde eles tem o mesmo nível de acesso aos recursos.

Um dos benefícios da nuvem pública é que elas tendem a ser muito maiores que as redes privadas e portanto mantém uma maior quantidade de recursos, isso permite uma maior escalabilidade de recursos. Essa característica evita a compra de equipamentos adicionais para resolver alguma necessidade temporária.

Empresas como o Google e Amazon EC2[11] oferecem esse tipo de serviço.

[11] http://aws.amazon.com/ec2/

2.2.2 Nuvem Privada

As nuvens privadas são aquelas que são construídas para uma única empresa, normalmente optam por esse tipo de nuvem as empresas que estão mais preocupadas com a segurança ou a garantia de disponibilidade sem o atraso da Internet. A infraestrutura pode ser administrada pela própria empresa, por terceiros ou por ambos e ela pode existir dentro ou fora das instalações da empresa.

Ferramentas como Eucalyptus, OpenNebula e OpenStack permitem a implantação desse tipo de modelo.

2.2.3 Nuvem Híbrida

Nuvens híbridas combinam os recursos das redes públicas e privadas, ela permite que redes privadas tenham uma reserva de recursos acessíveis via rede pública, isso permite manter o nível dos serviços mesmo que haja uma flutuação rápida da demanda. Uma empresa também pode escolher utilizar uma rede privada apenas para armazenar seus dados críticos, garantindo assim a segurança de seus dados em data centers dentro do domínio da própria empresa.

2.2.4 Nuvem Comunitária

Nuvens comunitárias são *clouds* mantidas por um conjunto de empresas que normalmente tem a mesma área de atuação e preocupações como segurança, política, entre outros. A nuvem pode ser administrada pelas empresas ou por um terceiro e ela pode existir no ambiente da empresa ou fora dele.

2.3 Conclusão

Nesse capítulo foi apresentada uma visão sobre o paradigma de computação em nuvem, suas classificações como modelo de implantação ou de serviços.

Capítulo 3

O OpenStack

Atualmente existem várias ferramentas para implementar um ambiente de computação em nuvem como IaaS, dentre as utilizadas e já pesquisadas pelo LRG estão o Eucalyptus e o OpenNebula [VIT 12]. Neste trabalho pretende-se estudar o OpenStack que é outra ferramenta de código aberto.

OpenStack é uma colaboração global de desenvolvedores e tecnólogos de computação em nuvem que produzem em código aberto implementações para nuvens públicas e privadas. O projeto visa oferecer soluções para todos os tipos de nuvens por ser simples de implementar, altamente escalável, e rico em recursos. A ferramenta consiste em uma série de projetos inter-relacionados que oferecem vários componentes como solução de infraestrutura de nuvem.
Suporta a implementação de nuvens privadas, públicas ou híbridas e oferece uma interface compatível com as interfaces EC2 e S3 da Amazon AWS largamente utilizada e considerada como padrão quando se trata de serviços para a nuvem. Oferece serviços de gerenciamento, segurança, escalabilidade e rede. Oferece aos usuários e administradores a possibilidade de escolha de múltiplos hipervisores: Xen, XenServer/XCP, KVM, UML, VMware. O OpenStack pode ser instalado em qualquer versão GNU Linux a partir de seu código fonte ou do repositório oficial da distribuição Linux. No presente trabalho foi utilizado o projeto com codinome Essex, que no presente momento é o mais atual [OPE 12].

3.1 História

Foi criado pela NASA em conjunto com Rackspace Hosting em julho de 2010, o principal objetivo era criar uma plataforma de *cloud* open source que suportasse uma arquitetura de milhares de servidores. A NASA havia tentado antes utilizar o software Eucalyptus, mas encontrou problemas de limitação.

Depois do lançamento da primeira versão o OpenStack ganhou inúmeros colaboradores e hoje grandes empresas como a Dell oferecem pacotes com seus servidores incluindo o OpenStack. Outras grandes empresas que tem investido em pesquisas e soluções utilizando o OpenStack são a HP, AT&T e até o site de leilões MercadoLibre[12] possui mais de 1000 VMs gerenciadas pelo OpenStack.

A tabela 3.1 apresenta todas as versões do OpenStack desde o seu lançamento:

Nome do release	Data de lançamento
Austin	21 de Outubro de 2010
Bexar	03 de Fevereiro de 2011
Cactus	15 de Abril de 2011
Diablo	22 de Setembro de 2011
Essex	05 de Abril de 2012
Folsom	Estimado para ser lançado no final de 2012

Tabela 3.1 - nome e a data dos releases[13].

[12] http://www.openstack.org/user-stories/mercadolibre-inc/

[13] http://docs.openstack.org/essex/openstack-compute/install/apt/content/version.html

3.2 Arquitetura

O OpenStack foi desenhado de forma modular e é basicamente formado de três grandes projetos de software livre que formam o seu núcleo: Nova, Glance e Swift. Existem ainda projetos adicionais e falaremos adicionalmente de dois, Horizon e Keystone, que foram utilizados no presente trabalho.

A figura 3.1 detalha a arquitetura do OpenStack e a inter-relação entre seus módulos.

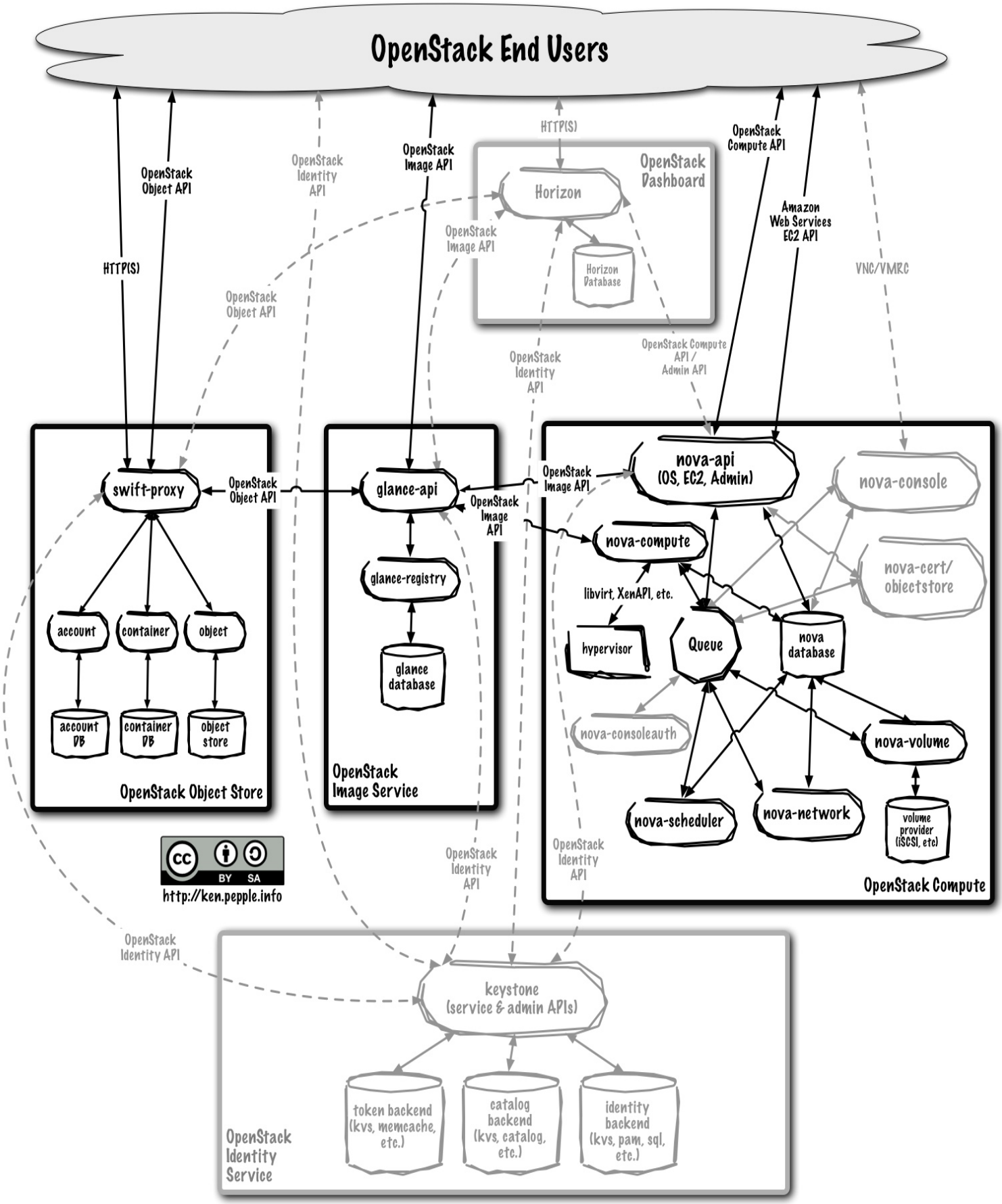


Figura 3.1 – Arquitetura do OpenStack Essex [DOC 12].

3.2.1 Nova

O Nova é a aplicação que gerencia toda a infraestrutura do OpenStack, todas as atividades necessárias para manter o ciclo de vida das instâncias de uma nuvem OpenStack são controlados pelo Nova. Ele gerencia todos os recursos computacionais como a rede, autorização, escalabilidade e recursos computacionais da nuvem. O Nova em si não possui nenhuma capacidade de virtualização, ao invés disso ele faz o uso de APIs para interagir com os *hypervisors* suportados.

No presente trabalho foi utilizado como *hypervisor* o KVM [KVM 12].

O Nova é subdividido em alguns módulos [DOC 12]:

- **Nova-API** – Este módulo é o responsável por fazer a comunicação com o mundo exterior, é através dele que é possível gerenciar a infraestrutura da cloud pelo mundo exterior. Ele garante compatibilidade com a API da Amazon (EC2) e toda troca de mensagens é feita por Web Services. Há também uma API própria do OpenStack que é utilizada para se comunicar com os outros componentes da cloud, e para isso ele utiliza o Message Queue (Rabbit MQ Server);

- **Rabbit MQ Server** – É o Message Queue do OpenStack, é através deste módulo que os componentes do OpenStack se comunicam. O Nova usa chamadas assíncronas para as requisições, com um mecanismo de call-back que é acionado quando a resposta é recebida. Uma vez que a comunicação é assíncrona, nenhum dos componentes ficam bloqueados por muito tempo em um estado de espera. Isto é especialmente importante dado que muitas ações invocadas através das APIs envolvem tempos longos, como criar uma instância ou fazer o upload de uma imagem;

- **Nova-Compute** – É um módulo composto por daemons que lidam com o ciclo de vida das instâncias das máquinas virtuais. Em uma implementação do OpenStack existem vários daemons, uma instância é criada por qualquer um deles, dependendo do tipo de escalonamento utilizado. Eles recebem as requisições do Message Queue;

- **Nova-Scheduler** – Este módulo é o responsável pelo escalonamento da cloud, ele mapeia as chamadas feitas às APIs aos componentes apropriados, disponíveis para ele através de um pool de recursos disponíveis. O escalonamento é feito baseado no tipo de algoritmo de escalonamento configurado;

- **Nova-Network** – É o controlador de rede da cloud, ele é responsável por distribuir os IPs para as VMs, lida com a configuração dos nodos, implementa os grupos de segurança, e faz a comunicação entre o controlador e as VMs;

- **Nova-Volume** - Responsável pelo gerenciamento dos volumes do tipo LVM, que são os volumes de armazenamento das instâncias, todo o processo de se criar, deletar, adicionar ou remover uma LVM a uma instância é realizado por esse módulo.

3.2.2 Glance

O Glance é o serviço de gerenciamento de imagens do OpenStack, é um sistema de busca, controle e armazenamento das imagens de máquinas virtuais da cloud.

Ele pode ser configurado para fazer isso localmente utilizando o Swift, remotamente utilizando o Amazon S3 diretamente, ou utilizando o Amazon S3 com o Swift como intermediário.

Ele suporta uma grande variedade de padrões de imagens, incluindo VDI (VirtualBox), VHD (Microsoft Hyper-V), QCOW2 (QEMU/KVM), VMDK/OVF (VMware) e bruto.

3.2.3 Swift

Equivalente ao serviço S3 da Amazon (Simple Storage Service), o Swift implementa um sistema de armazenamento de objetos provendo redundância e tolerância a falhas, ele é extremamente escalável tanto em termos de tamanho (vários petabytes) como capacidade (bilhões de objetos).

Ele é formado pelos seguintes componentes:

- **Swift Proxy Server** – Os clientes interagem com o Swift através desse componente, o proxy responde as requisições feitas via OpenStack Object API ou HTTP. Ele aceita arquivos para upload, modificações no metadado ou criação de novos containers. O Proxy Server também lida com falhas das entidades.

- **Swift Object Server –** É o servidor de objetos da cloud, sua função é lidar com o armazenamento, recuperação e remoção de objetos. Objetos são tipicamente arquivos binários, armazenados no sistema de arquivos.

- **Swift Container Server –** Contém a lista dos objetos de um Object Server. As listas são mantidas em bancos de dados. O Container Server também mantém estatísticas como o número de objetos armazenados e o tamanho do espaço de armazenamento ocupado por um Object Server;

- **Swift Account Server** - Contém a lista dos Object Servers; e,

- **O Anel –** Contém informações físicas dos objetos armazenados pelo Swift. É um mapeamento dos nomes de suas entidades em relação a sua localização física. Entidades como Accounts, Containers e Objetos possuem seus próprios anéis separados.

3.2.4 Keystone

É um projeto que provê um único ponto de integração para geração e gerenciamento de identidades, tokens, catálogos e políticas para os serviços da família OpenStack. Ele implementa a API de identidade do OpenStack

3.2.5 Horizon

É a interface do OpenStack, apresentando um dashboard de fácil visualização e manuseio, o Horizon fornece algumas opções para que seja possível gerenciar e monitorar a cloud de forma simples. Ela permite visualizar, criar, destruir e gerenciar VMs, instâncias, a rede, usuários e algumas outras funcionalidades básicas.

3.3 Conclusão

No presente capítulo foi apresentado a ferramenta para implementar um ambiente de computação em nuvem como IaaS, sua arquitetura, seus módulos e um pouco de sua história.

Capítulo 4

Monitoramento em Computação em Nuvem

O monitoramento é o processo de colher, analisar e apresentar informações. Para se analisar cada recurso é necessário que se tenha técnicas adequadas para obter informações sobre cada um deles.

Para a computação em nuvem os mais importantes recursos que precisam ser monitorados são os mesmos de uma máquina física, já que em uma máquina virtual é simulado o hardware de forma transparente para o sistema operacional, principalmente porque a ideia da computação em nuvem é de o cliente pagar o uso da máquina sob demanda, ou seja, é necessário saber corretamente quanto de memória física, virtual, processamento, rede, entre outros, um ambiente virtual está consumindo, para que seja feita a cobrança por parte do provedor de forma precisa.

Vários dos sistemas de ambientes virtualizados possuem suas próprias ferramentas de monitoramento de máquinas virtuais, mas esses são dependentes de plataforma e vários destes são de uso proprietário. Já os sistemas genéricos de código livre existentes são inadequados já que apresentam uma interface fraca e incompreensível. Os sistemas de código livre estão carentes de um sistema de monitoramento eficiente [CHA 10].

Em [CHA 10] é proposto um modelo de arquitetura genérica para monitoramento de nuvens privadas dividido em três camadas conforme ilustrado na figura 4.1.

Figura 4.1 – Arquitetura de três camadas para monitoramento de rede privada [CHA 10].

Esta arquitetura de monitoramento em três camadas foi projetada visando principalmente atender as necessidades das nuvens privadas, já que tem fácil integração com os sistemas de monitoramento utilizados nas empresas, como o Nagios, além de ser modular e portanto extensível para atender novos requisitos.

Camada de Visualização

É a camada de maior nível hierárquico, ela serve de interface para o gerenciamento de alto nível através da qual é possível verificar informações como o cumprimento de políticas organizacionais e SLAs. Usuários desta camada podem verificar as imagens das máquinas virtuais disponíveis para o instanciamento, níveis de serviço disponíveis e/ou negociáveis para as VMs instanciadas, e os dados monitorados das VMs em execução.[CHA 10].

Camada de Integração

A camada de integração é responsável por garantir um ambiente transparente em relação a infraestrutura heterogênea que normalmente um ambiente de nuvem possui. É ela que garante a compatibilidade dentre os diferentes tipos de hardware e software, trazendo dados para a camada de visualização. Portanto a camada de integração é responsável por abstrair quaisquer detalhes da infraestrutura [VIT 12].

Camada de Infraestrutura

É a camada base do modelo, consiste de todo hardware e software da infraestrutura, nela podemos monitorar recursos como dispositivos de armazenamento, rede, memória virtual, processamento, entre outros. Nesta camada também podemos especificar a granularidade do monitoramento, por exemplo podemos separar o monitoramento por cluster, nodo e máquina.

No trabalho de [CHA 10], a autora propõe a implementação de uma solução intitulada de PCMONS, que foi utilizado neste trabalho, e vai ser descrita na próxima seção.

4.1 Nagios

O Nagios é um sistema baseado em Unix e uma ferramenta de monitoramento de praticamente qualquer serviço que possa ser conectado a rede [NAG 13]. Ele possui uma licença GPL e vem sendo desenvolvido há mais de 10 anos e portanto possui ampla documentação e suporte da comunidade de código aberto. Ele é composto por três partes: seu núcleo, plugins e sua interface web. Os plugins são responsáveis por realizar as verificações e repassar os estados ao Nagios.

Principais características do Nagios [NAG 13]:

- Monitoração de recursos de computadores ou equipamentos de rede;
- Monitoração remota suportada através de túneis (cifrados SSH ou SSL);
- Monitoração redundante;
- Rotação automática de log;
- Verificar serviços paralelamente;
- Escalável: capaz de monitorar milhares de nodos;
- Suporta multiusuários em sua interface web;
- Monitoramento de serviços de rede (SMTP, POP3, HTTP, NNTP, PING, etc.);

- Sistema simples de plugins que permite aos usuários desenvolver suas próprias verificações de serviços;
- Capacidade de definir tratadores de eventos que executam tarefas em situações pré-determinadas ou para a resolução pró-ativas de problemas;
- Monitoramento de hosts executando diversos sistemas operacionais como Microsoft Windows, Unix/Linux, Novell NetWare, e outros; e,
- Notificações quando problemas de serviços ou de máquinas ocorrerem ou são resolvidas (por e-mail, SMS, pager ou algum método definido pelo usuário).

Foram desenvolvidos para o OpenStack, plugins de comunicação para suportar a compatibilidade com o Nagios, podendo assim monitorar serviços da nuvem, mas no presente trabalho esses plugins não foram utilizados para não comprometer a independência do PCMONS.

4.2 PCMONS

O PCMONS (Private Cloud Monitoring System) ou em português, Sistema de Monitoramento de Nuvens Privadas, é uma ferramenta modular e flexível desenvolvida por [CHA 10] com base na arquitetura genérica de monitoramento de três camadas descrita anteriormente. Após a realização deste trabalho o PCMONS garante compatibilidade com o Nagios em sua camada de visualização, e com o OpenStack, OpenNebula e Eucalyptus em sua camada de infraestrutura. O PCMONS basicamente coleta e prepara os dados para a camada de visualização e ele é dividido em vários módulos para simplificar futuras adaptações para ferramentas específicas e para facilitar seu estudo e aplicação, de acordo com [DC 11].

Em [VIT 12] ele específica cada um desses módulos:

Coletor de Informações do Nó (Node Information Gatherer) - Este módulo é responsável por coletar as informações locais em cada um dos nós da nuvem. Reúne informações sobre as máquinas virtuais locais e envia para o Data Integrator Cluster.

Cluster Integrador de Dados (Data Integrator Cluster) - Como a maioria das ferramentas organiza seus nós em clusters, há um agente específico que reúne e prepara os dados para a próxima camada. Este agente evita a transferência de dados desnecessários de cada nó para o Monitoring Data Integrator.

Integrador de Dados do Monitoramento (Monitoring Data Integrator) - Coleta e armazena dados da nuvem no banco de dados para fins históricos, e fornece esses dados para o Configuration Generator.

Monitor de Máquina Virtual (Virtual Machine Monitor) - Este módulo injeta scripts nas máquinas virtuais que enviam dados úteis a partir dela para o sistema de monitoramento. Exemplos desses dados são carga do processador e da memória.

Gerador de Configuração (Configuration Generator) - Recupera informações do banco de dados, por exemplo, para gerar os arquivos de configurações necessários para ferramentas de visualização a serem utilizadas na camada de visualização.

Servidor de Monitoramento (Monitoring Tool Server) - Este módulo é responsável por receber dados de monitoramento de recursos diferentes (por exemplo, do Monitor de Máquinas Virtuais). Seu objetivo é receber informações de monitoramento e tomar ações como armazená-las no módulo de banco de dados para fins históricos.

Interface de Usuário (User Interface) - O PCMONS utiliza a própria interface da ferramenta Nagios como interface do usuário.

Banco de Dados (Database) - Armazena dados necessários para o Gerador de Configuração e do Integrador de Dados do Monitoramento.

A figura 4.2 mostra os módulos descritos e a interação entre eles.

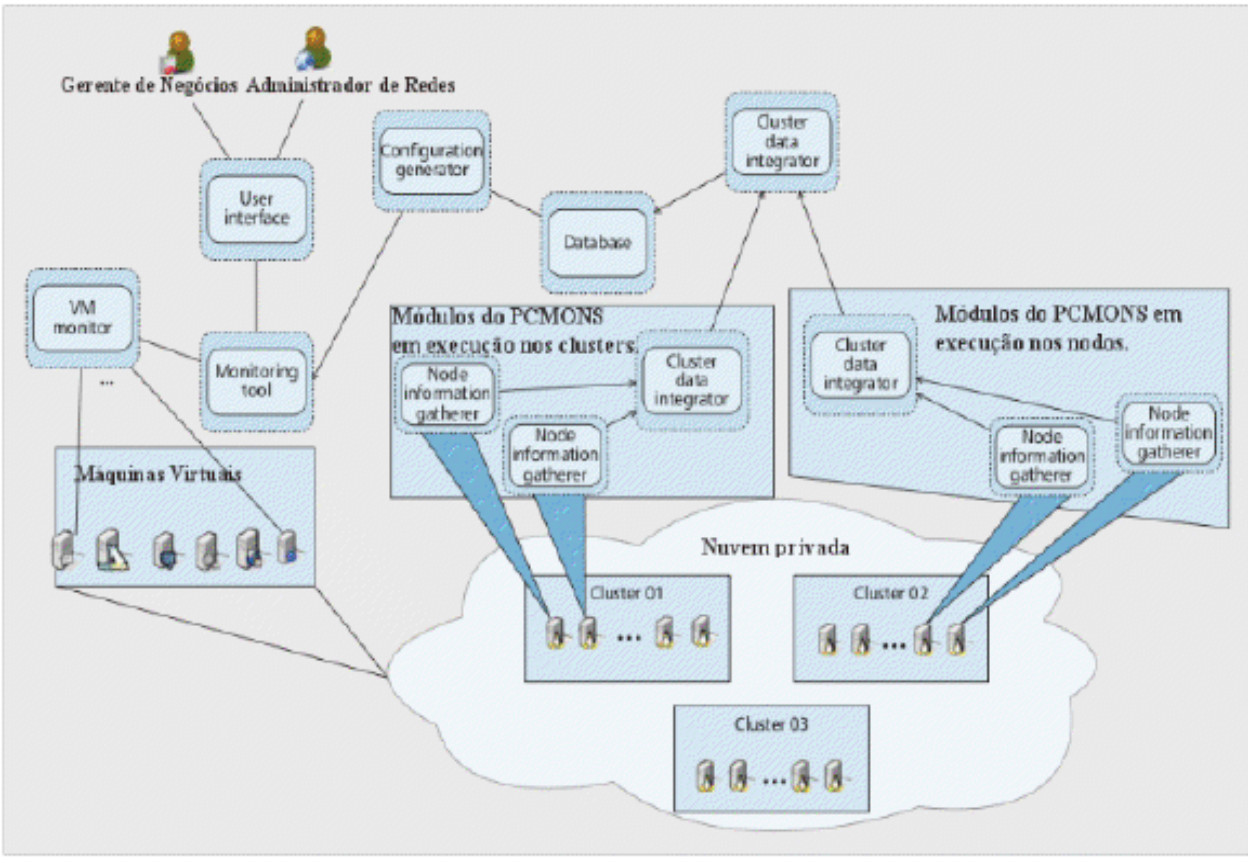


Figura 4.2 – Arquitetura da ferramenta PCMONS e interação entre seus componentes. [VIT 12].

O PCMONS funciona da seguinte maneira: a ferramenta colhe informações básicas sobre as máquinas virtuais como seu endereço IP, o usuário que a instanciou, e seu número de identificação e envia para a camada de visualização que em seguida cria os arquivos necessários para a monitoração destas máquinas virtuais. Neste processo é feito também o mapeamento entre a máquina virtual e seu host, característica importante do PCMONS já que para as ferramentas disponíveis para o gerenciamento de sistemas do tipo IaaS, esse mapeamento é abstraído, dificultando assim a resolução de problemas [URI 10].

4.3 Conclusão

Neste capítulo foi apresentado o conceito de monitoramento, principalmente de ambientes em nuvem, foram apresentadas também as duas principais ferramentas utilizadas para efetuar o monitoramento do ambiente implementado neste trabalho que foram o PCMONS, desenvolvido por [CHA 10] e o Nagios que é uma poderosa ferramenta flexível e modular, e de código aberto e livre.

Capítulo 5

Estudo de caso

5.1 Introdução

Foi implementado no LRG (Laboratório de Rede e Gerencia) da Universidade Federal de Santa Catarina um ambiente de computação em nuvem utilizando a ferramenta de IaaS chamada OpenStack. O OpenStack foi escolhido como ferramenta de implementação por ser de código aberto e por estar conquistando o topo entre as melhores ferramentas para implementação em nuvem no ramo de IaaS. Dentre suas principais vantagens, podemos citar:

- Controle e flexibilidade – Com uma tecnologia de código livre o usuário nunca está preso a um desenvolvedor proprietário, e o design modular do OpenStack permite que o usuário integre softwares de terceiros para atender suas necessidades.

- Escalabilidade – O OpenStack já é utilizado por empresas mundiais que oferecem grandes nuvens públicas que operam com volume de dados da ordem de petabytes.

- Padrão aberto escolhido pela indústria – Mais de 75 (setenta e cinco) empresas líderes de mais de uma dúzia de países estão participando do projeto do OpenStack, incluindo Cisco, Citrix, Dell, Intel e Microsoft.

- Compatibilidade – A ferramenta garante compatibilidade com outros ambientes, possuindo APIs para se comunicar com a AWS e S3 da Amazon e com o pacote euca-tools do Eucalyptus, além de conseguir operar utilizando vários hypervisores diferentes como por exemplo o ESX, Hyper-V, KVM, LXC, QEMU, UML, Xen, e o XenServer.

O PCMONS foi projetado para coletar informações na camada de integração independente da ferramenta IaaS utilizada, mas para isso é necessário estender as compatibilidade do módulo, e esse foi um dos principais focos deste trabalho, permitir a utilização do PCMONS com o OpenStack.

5.2 Implementação

Foi necessário estender alguns módulos do PCMONS para que este suportasse o monitoramento do OpenStack. As alterações de código feitas em cada um dos módulos foram:

Cluster Integrador de Dados

- Arquivo de configuração do agente;
- Mapeamento dos nós de acordo com a ferramenta de IaaS utilizada;
- Atualização das informações das máquinas virtuais que estão sendo gerenciadas.

Coletor de informações do Nó

- Obtenção de informações das máquinas virtuais locais e envio para o Cluster Integrador de Dados.

Adaptação para nova versão da biblioteca BOTO (python - AWS)

- Mudanças no código para adequar o PCMONS ao BOTO.

E foram detalhadas na sequência do texto.

Running_vms: Neste módulo se localiza o arquivo de configuração que indica qual o tipo de infraestrutura foi utilizado, que após o presente trabalho suporta o Eucalyptus, o OpenNebula e agora o OpenStack. Dentro deste arquivo foram adicionadas variáveis conforme mostrado na figura 5.1.

```
#Configurations file for openStack server, must be accessible by this server

PATH_TO_OPENSTACK_CONFIG_FILE = "/etc/nova/nova.conf"
```

Figura 5.1 – Atributo criado no arquivo de configuração do PCMONS.

Neste mesmo módulo foi incrementado o método que faz a busca das máquinas virtuais presentes no servidor. A figura 5.2 mostra o código modificado sem que o mesmo perdesse a compatibilidade com as outras ferramentas suportadas anteriormente.

```
def grepNodes(self, infra=cluster_config.INFRA):
        '''
        provides the list eucalyptus nodes
        '''
        if infra == 'eucalyptus':
                file=cluster_config.PATH_TO_EUCALYPTUS_CONFIG_FILE
                pattern="NODES"
            fileConf = open(file,'r')
            Nodes = ''
            for line in fileConf:
                if pattern in line:
                    Nodes = Nodes + line
                    Nodes = Nodes.split('\"')
                    NC = Nodes[1].split()
                    return NC

        elif infra == 'opennebula':
                command = cluster_config.PATH_TO_OPENNEBULA + "/bin/onehost list | grep '
on' | awk {' print $2 '}"
            get_names = commands.getoutput(command).split("\n")
            Nodes = []
            for i in range(len(get_names)):
                addr = socket.gethostbyname(get_names[i])
                Nodes.append(addr)
            return Nodes

            elif infra == 'openStack':
                file=cluster_config.PATH_TO_OPENSTACK_CONFIG_FILE
```

```
            pattern="--cc_host"
            fileConf = open(file,'r')
        Nodes = ''
        for line in fileConf:
            if pattern in line:
                Nodes = Nodes + line
                Nodes = Nodes.split('=')
                NC = Nodes[1].split()
                return NC
    else:
        return 'not implemented yet'
```

Figura 5.2 – Método que recupera os Nodos de cada infraestrutura.

Nesta mesma classe foi preciso também fazer uma outra alteração devido a diferença de versão da biblioteca BOTO que é usada pelo python para se comunicar com a API EC2 da ferramenta.

Ainda dentro deste módulo no arquivo responsável por captar informações de cada um dos nós precisou ser adicionado uma função que suportasse o OpenStack como infraestrutura. Na figura 5.3 está o código da função adicionada.

```
def openStack(self):
        '''
        This function get information about vms managed by openstack
        '''
        hostname = commands.getoutput('hostname')
    vms = [ ]
    vm = { }
        vms_running_host = commands.getoutput("virsh list | cut -c5-21")
    vms_running_host = vms_running_host.split()[3:]
    for vm in vms_running_host:
        vm = {'node_hostname':hostname,'instance_id':vm,}
        vms.append(vm)
    if len(vms) > 0:
        logging.debug("Vms returned :%s"%vms)
    else:
        logging.debug("No vm was returned")
        print vms
    return vms
```

Figura 5.3 – Método que recupera informações das VMs de cada nó.

Já os outros arquivos deste módulo como o de acesso ao banco não sofreram modificações tendo em vista que os processos e recursos monitorados e seus dados armazenados continuaram os mesmos.

Interface: Nenhum arquivo foi modificado neste módulo, já que a versão do Nagios utilizada permaneceu a 3.2.0 da qual já garantia compatibilidade. Novas versões do Nagios já foram lançadas, mas esta versão é dita a mais estável e atualmente é a mais utilizada pela comunidade internacional.

Instalação: Os arquivos para a criação do banco também não foram modificados, já que não foram adicionados novos parâmetros de monitoramento.

5.3 Ambiente

O OpenStack permite que em uma mesma máquina sejam instanciados todos os serviços necessários para a infraestrutura da nuvem, tais como: controlador da nuvem, controlador de imagem, armazenamento de imagem e serviço de autenticação. A figura 5.4 descreve os principais hardwares e softwares utilizados para a implementação deste trabalho.

Hardware	Software
AMD Phenom(tm) II X4 965 Processor 4GB DDR2 667MHz HD Samsung HD502HJ 500GB ATA	Ubuntu 12.04 OpenStack Essex Nagios Core 3.2.0

Figura 5.4 – Hardwares e Softwares utilizados no estudo de caso.

O Ubuntu foi escolhido como sistema operacional base pelo fato de ser muito utilizado e ter uma documentação sólida e atualizações constantes, além da facilidade de uso e ser altamente recomendado pela documentação do OpenStack como host de seu sistema.

O hipervisor escolhido foi o KVM, pois como o hardware utilizado apresenta suporte a virtualização, foi possível efetuar a virtualização completa, não sendo necessário assim efetuar nenhuma alteração no kernel do sistema operacional. Mas o OpenStack suporta também outros tipos de hipervisores caso se deseje adicionar mais nodos, tendo eles suporte nativo a virtualização ou não, e neste último caso seria utilizado a paravirtualização, sendo necessárias alterações no kernel para simular em uma camada de software, o hardware de virtualização.

O Nagios foi escolhido por ser de código aberto, amplamente utilizado e com arquitetura modular, além de já ser suportado pelo PCMONS.

5.4 OpenStack Essex

Logo após a instalação do sistema operacional é preciso instalar algumas dependências antes da instalação do OpenStack propriamente dito. O OpenStack é compatível com dois bancos de dados amplamente utilizados e de código aberto, o MySQL e o SQLAlchemy que é um pacote de ferramentas SQL da linguagem python. Neste trabalho foi escolhido o MySQL como banco de dados pela familiaridade do autor com a ferramenta.

O OpenStack tem duas maneiras de implementação para se comunicar com as máquinas virtuais através da rede, utilizando-se de Vlans ou através de uma Bridge, neste trabalho foi escolhido a implementação por Bridge por não requerer a mudança na configuração dos Switches do laboratório.

5.4.1 Pré-instalação

Os requisitos de software para a instalação do OpenStack são [DOC 12] [LOS 12]:

- **bridge-utils;**
- **ntp;**
- **tgt;**
- **open-iscsi e open-iscsi-utils;**
- **python-mysqldb e mysql-server**

Após a instalação desses pacotes que podem ser feitas utilizado-se a ferramenta apt-get do Ubuntu, é necessário criar os databases no banco de dados que vão ser utilizados pelos diferentes serviços do OpenStack (Nova, Glance e Keystone). Os comandos para a criação dos databases está descrito na figura 5.5 e podem ser executados no terminal:

```
mysql -u root -p
CREATE DATABASE keystone_db;
GRANT ALL ON keystone_db.* TO 'keystone'@'%' IDENTIFIED BY 'keystone';
GRANT ALL ON keystone_db.* TO 'keystone'@'localhost' IDENTIFIED BY 'keystone';
CREATE DATABASE glance_db;
GRANT ALL ON glance_db.* TO 'glance'@'%' IDENTIFIED BY 'glance';
GRANT ALL ON glance_db.* TO 'glance'@'localhost' IDENTIFIED BY 'glance';
CREATE DATABASE nova_db;
GRANT ALL ON nova_db.* TO 'nova'@'%' IDENTIFIED BY 'nova';
GRANT ALL ON nova_db.* TO 'nova'@'localhost' IDENTIFIED BY 'nova';
```

Figura 5.5 – Comandos para criação dos bancos de dados do OpenStack.

5.4.2 Instalação

Logo após a pré-instalação o sistema está pronto para a instalação dos serviços do OpenStack.

5.4.2.1 Keystone

Começamos por instalar o serviço Keystone:

```
sudo apt-get install keystone python-keystone python-mysqldb python-keystoneclient
```

Figura 5.6 – Comandos para instalação do Keystone.

E logo em seguida fazemos as configurações necessárias para que ele funcione com o MySQL modificando seu arquivo de configuração presente em:

`/etc/keystone/keystone.conf`

E modificando as linhas mostradas na figura 5.7:

```
connection = mysql://keystone:keystone@150.162.63.26/keystone_db

admin_token = testelab
```

Figura 5.7 – Configuração do Keystone para funcionar com o MySQL.

Após a reinicialização dos serviços é possível executar o comando que sincroniza a ferramenta com o banco de dados, criando as tabelas necessárias pela utilização do mesmo:

```
sudo keystone-manage db_sync
```

Figura 5.8 – Comando para sincronizar a base de dados do KeyStone.

Foram utilizados dois scripts: keystone_data.sh e endpoints.sh criados por [LOS 12] que devem ser executados de forma sequencial e na ordem que foram apresentados para a criação de variáveis de sistema e a inserção de dados de autenticação ao Keystone e dos demais serviços que vão ser instalados posteriormente. Estes dois scripts estão descritos no APÊNDICE A e APÊNDICE B.

A utilização dos scripts é mostrada na figura 5.9:

```
sudo ./keystone_data.sh

sudo ./endpoints.sh -m 150.162.63.26 -u keystone -D keystone_db -p
keystone -K 150.162.63.26 -R RegionOne
-E "http://localhost:35357/v2.0" -S 150.162.63.26 -T testelab
```

Figura 5.9 – Comandos para executar scripts que criam varáveis de ambiente necessárias.

5.4.2.2 Glance

Com o Keystone instalado chegou a vez de instalar o Glance, que é o serviço de armazenamento de imagens do OpenStack. Para baixar a última versão basta utilizar o apt-get como mostrado na figura 5.10:

```
sudo apt-get install glance glance-api glance-client glance-common
glance-registry python-glance
```

Figura 5.10 – Comando para instalação do Glance.

Alterar os arquivos de configuração presentes em: /etc/glance/glance-api-paste.ini e /etc/glance/glance-registry-paste.ini, para utilizar os parâmetros setados na figura 5.9, modificando as variáveis conforme a figura 5.11:

```
admin_tenant_name = service
admin_user = glance
admin_password = testelab
```

Figura 5.11 – Valores das variáveis necessárias para se acessar o banco de dados do Glance.

E adicionando as linhas da figura 5.12 ao final de cada um dos dois arquivos e também nos arquivos presentes em:
"/etc/glance/glance-registry.conf" e "/etc/glance/glance-api.conf"

```
[paste_deploy]
flavor = keystone
```

Figura 5.12 – Linha que setam o Keystone como programa de autenticação do Glance.

Ainda dentro de /etc/glance/glance-registry.conf é necessário mudar a linha que especifica o tipo de banco de dados utilizado, configurado-a para o MySQL conforme é mostrado na figura 5.13 :

```
sql_connection = mysql://glance:glance@150.162.63.26/glance_db
```

Figura 5.13 – Linha que especifica o MySQL como banco de dados do Glance.

Após reiniciar o serviço Glance para que as alterações entrem em vigor, o sistema está pronto para efetuar a sincronização com o banco de dados para criar as tabelas necessárias para seu funcionamento:

```
sudo glance-manage version_control 0
sudo glance-manage db_syn
```

Figura 5.14 – Comandos para sincronizar o Glance com o banco de dados.

5.4.2.3 Nova

Depois da instalação do Keystone e do Glance é preciso instalar o Nova, que é o gerenciador da infraestrutura computacional de uma nuvem OpenStack.

Primeiro fazemos utilizamos o apt-get para baixar e instalar todos os arquivos do Nova e suas dependências:

```
sudo apt-get install nova-api nova-compute nova-compute-kvm nova-doc
nova-network nova-objectstore nova-scheduler nova-volume  rabbitmq-
server novnc nova-consoleauth
```

Figura 5.15 – Comando para baixar e instalar o Nova e suas dependências.

Depois da instalação é preciso mudar as permissões da pasta permitir o acesso ao usuário Nova:

```
sudo chown -R nova:nova /etc/nova
sudo chmod 644 /etc/nova/nova.conf
```

Figura 5.16 – Comandos para criar o usuário Nova e dar-lhe acesso à pasta.

E no arquivo localizado em "/etc/nova/api-paste.ini" altera-se as linhas da figura 5.17 para utilizar os parâmetros setados na figura 5.9:

```
admin_tenant_name = service
admin_user = nova
admin_password = testelab
```

Figura 5.17 – Valores necessários para acessar o banco de dados do Nova.

Para permitir o acesso ao Keystone, utilizando as mesmas credenciais configuradas previamente.

No Nova, a maioria da configuração é feita em apenas um arquivo, o nova.conf, presente em "/etc/nova/". É neste arquivo que é determinado o hipervisor do nodo, o endereço do servidor onde está hospedado o banco de dados, o tipo de infraestrutura de rede utilizada para fazer a comunicação com as máquinas virtuais, as APIs que vão ser utilizadas, o endereço do servidor de imagens (Glance), os arquivos de logs e muitas outras configurações, algumas das quais não foram especificadas pois foi utilizado seu valor padrão como porta e endereço de acesso de alguns serviços.

Na figura 5.18 está especificado o conteúdo do arquivo "nova.conf" utilizado neste trabalho:

```
##### RabbitMQ #####
--rabbit_host=150.162.63.26

##### MySQL #####
--sql_connection=mysql://nova:nova@150.162.63.26/nova_db

##### nova-api #####
--auth_strategy=keystone
--cc_host=150.162.63.26

##### nova-network #####
--network_host=150.162.63.26
--fixed_range=10.0.0.1/26
--network_size=256
--force_dhcp_release=True
--fixed_ip_disassociate_timeout=30
--my_ip=150.162.63.26
--routing_source_ip=150.162.63.26
--auto_assign_floating_ip=true
```

```
--dhcpbridge_flagfile=/etc/nova/nova.conf
--dhcpbridge=/usr/bin/nova-dhcpbridge
--network_manager=nova.network.manager.FlatDHCPManager
--flat_network_dhcp_start=10.0.0.2
--flat_network_bridge=br100
--flat_interface=eth0
--flat_injected=False
--public_interface=br100
--floating_range=10.0.1.1/26
--vlan_interface=eth0


##### nova-compute settings #####
--connection_type=libvirt
--libvirt_type=kvm # if you demo on VirtualBox, use 'qemu'
--libvirt_use_virtio_for_bridges=True
--use_cow_images=True
--snapshot_image_format=qcow2


##### nova-volume #####
--iscsi_ip_prefix=150.162.63.26
--num_targets=100
--iscsi_helper=tgtadm

##### glance #####
--image_service=nova.image.glance.GlanceImageService
--glance_api_servers=150.162.63.26:9292

##### VNC #####
--novnc_enabled=true
--novncproxy_base_url=http://150.162.63.26:6080/vnc_auto.html
--vncserver_proxyclient_address=150.162.63.26
--vncserver_listen=150.162.63.26

##### Misc #####
--logdir=/var/log/nova
--state_path=/var/lib/nova
--lock_path=/var/lock/nova
--root_helper=sudo nova-rootwrap
--verbose
```

Figura 5.18 – Arquivo de configuração do Nova.

O nova-volume, que é o módulo do nova onde é armazenado as informações das instâncias das máquinas virtuais e precisa ser configurado. Neste trabalho foi criada uma partição específica para este módulo, usando os seguintes comandos:

```
sudo pvcreate /dev/sda6
sudo vgcreate nova-volumes /dev/sda6
```

Figura 5.19 – Comando para especificar o sda6 como volume onde vão ser armazenadas as VMs

Após a configuração do nova-volume e a reinicialização de todos os serviços do Nova é preciso fazer a sincronização com o banco de dados:

```
sudo nova-manage db sync
```

Figura 5.20 – Comando que sincroniza o Nova com o banco de dados.

Como a infraestrutura de rede escolhida para se fazer a comunicação do Nova com as instâncias das máquinas virtuais foi o bridge, é necessário executar o comando mostrado na figura 5.21 que cria uma rede para que o servidor e as VMs se comuniquem:

```
sudo nova-manage network create –label=nova-network --fixed_range_v4=10.0.0.1/28
--num_networks=1 --bridge=br100 --bridge_interface=eth0 --network_size=8
```

Figura 5.21 – Comando para criar a rede entre o servidor e as VMs.

E criar um pool de endereços IPs que vão ser distribuídos automaticamente para as instâncias em tempo de criação:

```
sudo nova-manage floating create --pool pool1 --ip_range 10.0.0.1/28
```

Figura 5.22 – Comando para criar o pool de endereços Ips que vão ser utilizados pelas VMs.

Para poder acessarmos as máquinas virtuais através do ssh, é necessário incluirmos regras ao grupo de rede utilizado pelas instâncias:

```
nova secgroup-add-rule default icmp -1 -1 0.0.0.0/0
nova secgroup-add-rule default tcp 22 22 0.0.0.0/0
nova secgroup-add-rule default tcp 80 80 0.0.0.0/0
```

Figura 5.23 – Comandos que liberam o acesso a algumas portas das VMs.

Neste ponto o sistema já está pronto para adicionar imagens e instanciá-las. Para este trabalho foram baixadas imagens prontas da Internet, mas é possível achar vários tutorias que descrevem como criá-las.

5.4.2.4 Horizon

O Horizon é o projeto de dashboard do OpenStack, ele tem a função de oferecer uma interface simples onde se é possível criar, remover, reiniciar ou pausar instâncias, além de permitir alterações em algumas configurações de rede ou de acesso, tudo isso apenas sendo necessário um navegador.

Para instalar o Horizon também se utiliza o apt-get como é mostrado na figura 5.24:

```
sudo apt-get install apache2 libapache2-mod-wsgi openstack-dashboard
```

Figura 5.24 – Comando para instalar o Horizon.

Na figura 5.25 temos a página inicial do Horizon onde ele apresenta um relatório mensal do uso de processamento e memória, separados por projeto:

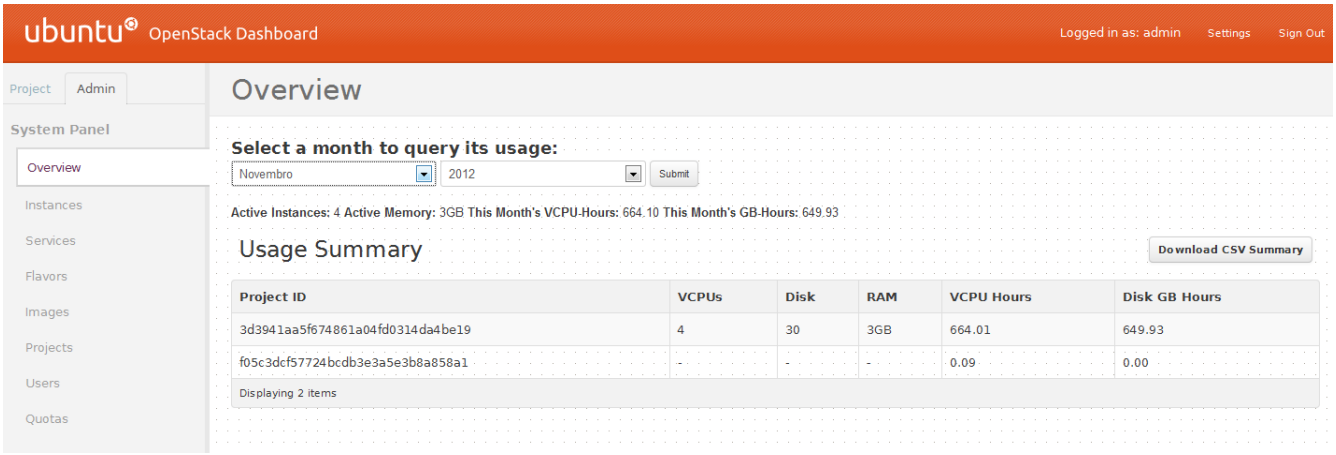


Figura 5.25 – Tela do relatório do consumo de memória e processamento.

E na figura 5.26 apresenta a tela que mostra as imagens previamente adicionadas ao Glance, nota-se que é possível apenas remover as imagens, para se adicionar uma outra é necessário fazê-lo via linha de comando no terminal.

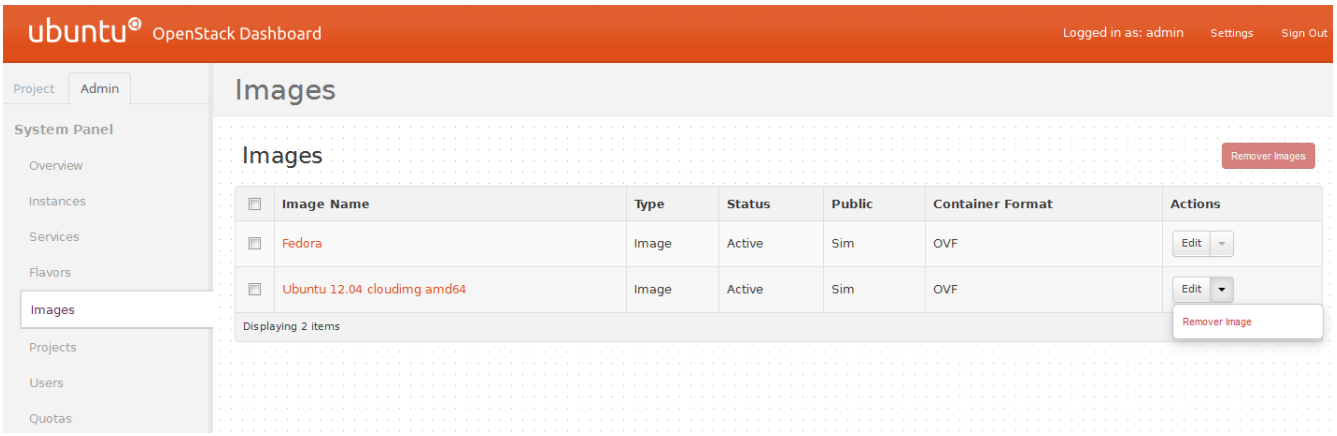


Figura 5.26 – Tela da lista de imagens disponíveis no servidor.

O Horizon também apresenta uma tela que permite o gerenciamento das instâncias das máquinas virtuais, permitindo funções como: acesso ao console da instância, pausa, suspensão, reinicialização e exclusão da mesma. Como mostra a figura 5.27.

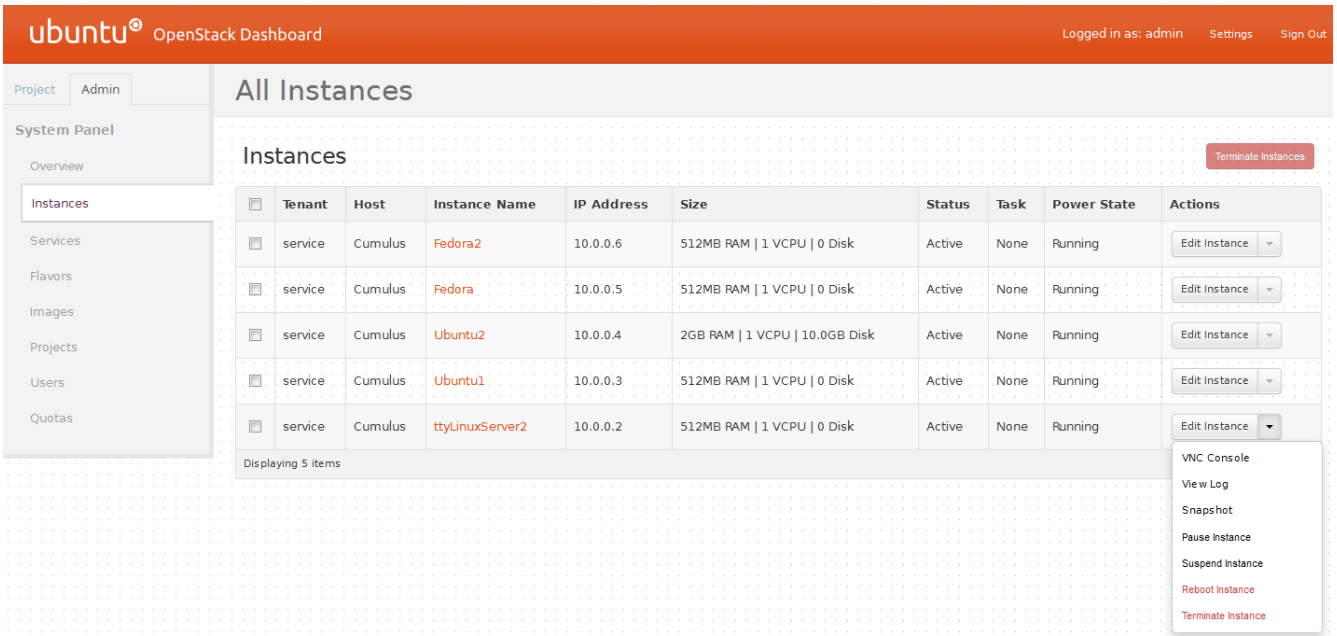


Figura 5.27 – Tela de gerenciamento das instâncias das VMs.

É possível verificar na figura 5.27 que foram criadas cinco instâncias de máquinas virtuais a partir de duas imagens. Três delas utilizando o Ubuntu e duas delas utilizando a distribuição Fedora. E para uma das instâncias foi disponibilizado um pouco mais de memória física e virtual mostrando que é possível ter vários tipos de modelos, chamados de *flavors*, para se instanciar uma VM.

5.5 Nagios e PCMONS

O Nagios 3.2.0 também foi instalado através do apt-get utilizando o comando:

```
sudo apt-get install nagios3
```

Figura 5.28 – Comando para instalar o Nagios 3.2.0.

Após a instalação do Nagios é necessário modificar o arquivo de configuração "nagios.cfg" presente em "/etc/nagios3/" para que o Nagios passe a verificar se existe alguma informação no arquivo "/var/lib/nagios3/rw/nagios.cmd" que é onde ficam armazenadas informações monitoradas das instâncias enviadas pelo PCMONS. Para isso é preciso mudar o valor da variável "check_external_commands" de 0 para 1.

Instalar o PCMONS é uma tarefa simples, basta copiá-lo para dentro da pasta "/opt/" e iniciar os dois serviços que estão em "pcmons/init.d". Para facilitar a inicialização do PCMONS, foi adicionado à "/etc/rc.local" o comando para iniciar os serviços, e com isso eles são iniciados em tempo de boot.

Já para as máquinas virtuais, também é necessário copiar o PCMONS para a pasta "/opt/" e executar o script dentro de "booting_vms" chamado "Monitor.py".

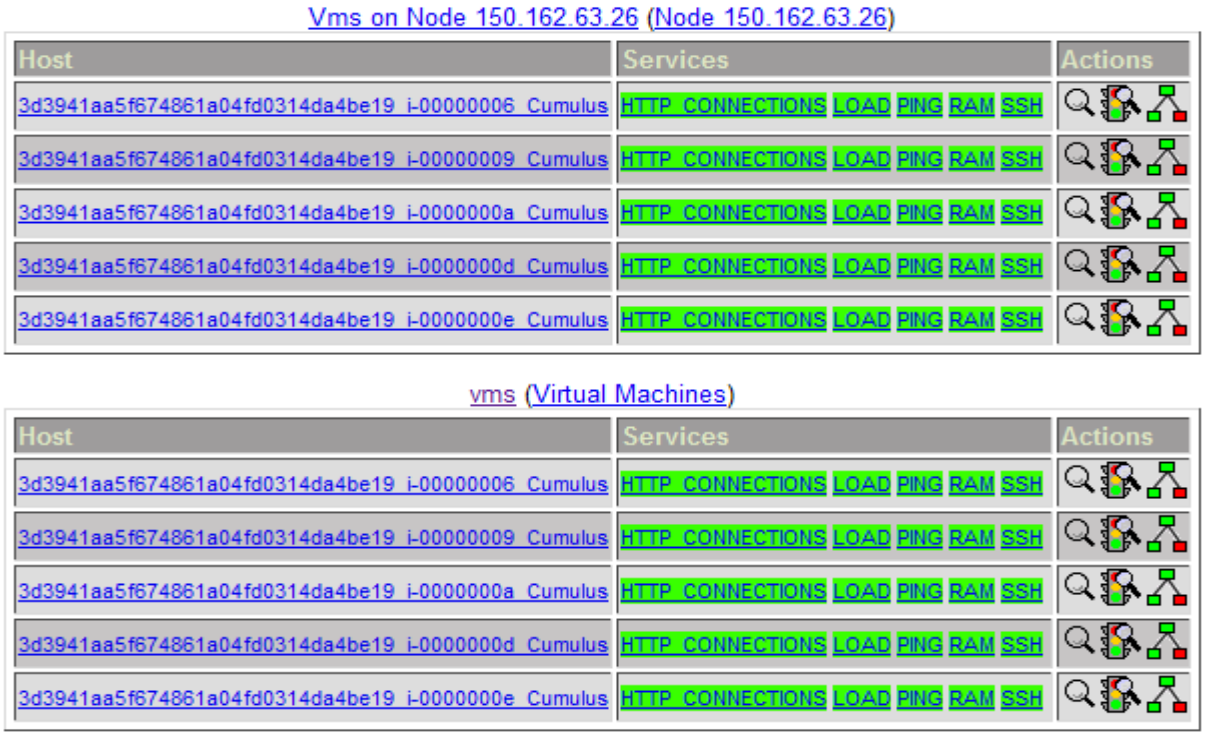
Com o PCMONS e o Nagios instalados e operacionais, deve-se rodar o arquivo "Generate_VMs_Nagios_Conf-v0.2.py" que está dentro de "/opt/pcmons/interface/nagios/cron" para que seja atualizado a lista de instâncias no Nagios. O ideal é que esse script seja adicionado ao cron do Linux para rodar a cada cinco minutos, automatizando assim o processo.

A figura 5.29 mostra a interface da ferramenta Nagios sendo utilizada pelo PCMONS, que mostra a lista de todas as instâncias de máquinas virtuais sendo utilizadas no momento. A nomenclatura utilizada para representar cada uma delas é composta do identificador do usuário que iniciou a VM, do identificador da VM, e o nome da máquina física onde está a máquina virtual.

Host	Status	Last Check	Duration	Status Information
3d3941aa5f674861a04fd0314da4be19_i-00000006_Cumulus	UP	2012-11-23 20:04:53	0d 19h 18m 21s	PING OK - Packet loss = 0%, RTA = 0.56 ms
3d3941aa5f674861a04fd0314da4be19_i-00000009_Cumulus	UP	2012-11-23 20:05:53	0d 19h 18m 0s	PING OK - Packet loss = 0%, RTA = 0.58 ms
3d3941aa5f674861a04fd0314da4be19_i-0000000a_Cumulus	UP	2012-11-23 20:06:53	0d 19h 16m 48s	PING OK - Packet loss = 0%, RTA = 0.42 ms
3d3941aa5f674861a04fd0314da4be19_i-0000000d_Cumulus	UP	2012-11-23 20:02:43	0d 19h 17m 38s	PING OK - Packet loss = 0%, RTA = 0.48 ms
3d3941aa5f674861a04fd0314da4be19_i-0000000e_Cumulus	UP	2012-11-23 20:06:53	0d 17h 48m 28s	PING OK - Packet loss = 0%, RTA = 0.48 ms
localhost	UP	2012-11-23 20:03:13	17d 6h 9m 35s	PING OK - Packet loss = 0%, RTA = 0.11 ms

Figura 5.29 – Nagios: Lista das máquinas virtuais monitoradas pelo sistema.

Neste estudo de caso as informações monitoradas pelo PCMONS englobam dois conjuntos de monitoramento, um em relação ao serviço que está sendo executado (número de conexões HTTP), e outro em relação ao desempenho e disponibilidade da máquina virtual (memória, ssh, ping e carga). A figura 5.30 mostra todos esses serviços passivos (HTTP CONNECTIONS, LOAD, RAM) e ativos (PING, SSH) que estão sendo monitorados pelo PCMONS.



Vms on Node 150.162.63.26 (Node 150.162.63.26)

Host	Services	Actions
3d3941aa5f674861a04fd0314da4be19_i-00000006_Cumulus	HTTP_CONNECTIONS LOAD PING RAM SSH	
3d3941aa5f674861a04fd0314da4be19_i-00000009_Cumulus	HTTP_CONNECTIONS LOAD PING RAM SSH	
3d3941aa5f674861a04fd0314da4be19_i-0000000a_Cumulus	HTTP_CONNECTIONS LOAD PING RAM SSH	
3d3941aa5f674861a04fd0314da4be19_i-0000000d_Cumulus	HTTP_CONNECTIONS LOAD PING RAM SSH	
3d3941aa5f674861a04fd0314da4be19_i-0000000e_Cumulus	HTTP_CONNECTIONS LOAD PING RAM SSH	

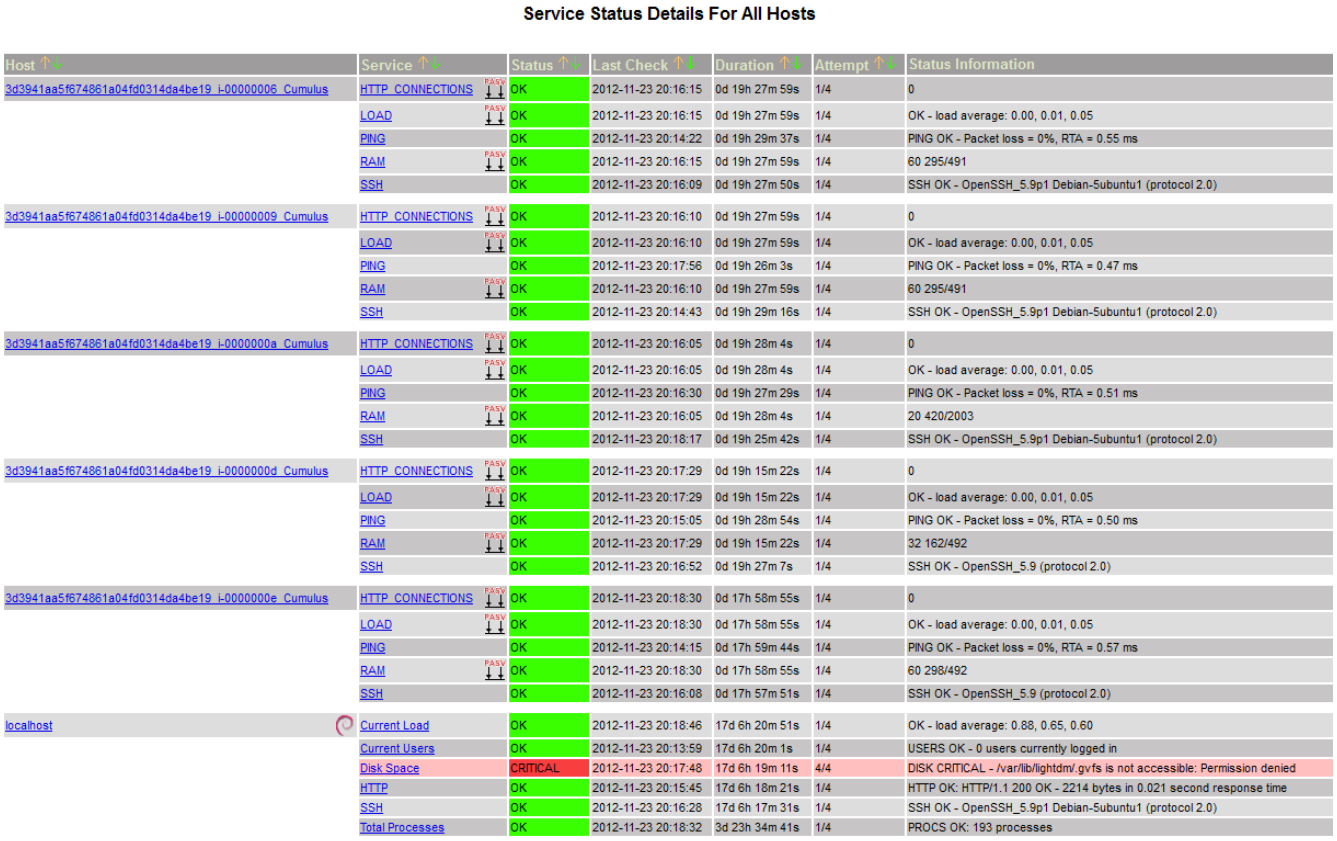
vms (Virtual Machines)

Host	Services	Actions
3d3941aa5f674861a04fd0314da4be19_i-00000006_Cumulus	HTTP_CONNECTIONS LOAD PING RAM SSH	
3d3941aa5f674861a04fd0314da4be19_i-00000009_Cumulus	HTTP_CONNECTIONS LOAD PING RAM SSH	
3d3941aa5f674861a04fd0314da4be19_i-0000000a_Cumulus	HTTP_CONNECTIONS LOAD PING RAM SSH	
3d3941aa5f674861a04fd0314da4be19_i-0000000d_Cumulus	HTTP_CONNECTIONS LOAD PING RAM SSH	
3d3941aa5f674861a04fd0314da4be19_i-0000000e_Cumulus	HTTP_CONNECTIONS LOAD PING RAM SSH	

Figura 5.30 – Nagios: Serviços sendo monitorados pelo sistema.

Na figura 5.30 também mostra que o Nagios faz o agrupamento de máquinas virtuais por nodo, facilitando assim a identificação de falhas no caso de um nodo com problemas.

O Nagios ainda apresenta uma tela com as informações detalhadas de cada serviço de todas as máquinas virtuais, como mostra a figura 5.31:



Service Status Details For All Hosts

Host	Service	Status	Last Check	Duration	Attempt	Status Information
3d3941aa5f674861a04fd0314da4be19_i-00000006_Cumulus	HTTP_CONNECTIONS	OK	2012-11-23 20:16:15	0d 19h 27m 59s	1/4	0
	LOAD	OK	2012-11-23 20:16:15	0d 19h 27m 59s	1/4	OK - load average: 0.00, 0.01, 0.05
	PING	OK	2012-11-23 20:14:22	0d 19h 29m 37s	1/4	PING OK - Packet loss = 0%, RTA = 0.55 ms
	RAM	OK	2012-11-23 20:16:15	0d 19h 27m 59s	1/4	60 295/491
	SSH	OK	2012-11-23 20:16:09	0d 19h 27m 50s	1/4	SSH OK - OpenSSH_5.9p1 Debian-5ubuntu1 (protocol 2.0)
3d3941aa5f674861a04fd0314da4be19_i-00000009_Cumulus	HTTP_CONNECTIONS	OK	2012-11-23 20:16:10	0d 19h 27m 59s	1/4	0
	LOAD	OK	2012-11-23 20:16:10	0d 19h 27m 59s	1/4	OK - load average: 0.00, 0.01, 0.05
	PING	OK	2012-11-23 20:17:56	0d 19h 26m 3s	1/4	PING OK - Packet loss = 0%, RTA = 0.47 ms
	RAM	OK	2012-11-23 20:16:10	0d 19h 27m 59s	1/4	60 295/491
	SSH	OK	2012-11-23 20:14:43	0d 19h 29m 16s	1/4	SSH OK - OpenSSH_5.9p1 Debian-5ubuntu1 (protocol 2.0)
3d3941aa5f674861a04fd0314da4be19_i-0000000a_Cumulus	HTTP_CONNECTIONS	OK	2012-11-23 20:16:05	0d 19h 28m 4s	1/4	0
	LOAD	OK	2012-11-23 20:16:05	0d 19h 28m 4s	1/4	OK - load average: 0.00, 0.01, 0.05
	PING	OK	2012-11-23 20:16:30	0d 19h 27m 29s	1/4	PING OK - Packet loss = 0%, RTA = 0.51 ms
	RAM	OK	2012-11-23 20:16:05	0d 19h 28m 4s	1/4	20 420/2003
	SSH	OK	2012-11-23 20:18:17	0d 19h 25m 42s	1/4	SSH OK - OpenSSH_5.9p1 Debian-5ubuntu1 (protocol 2.0)
3d3941aa5f674861a04fd0314da4be19_i-0000000d_Cumulus	HTTP_CONNECTIONS	OK	2012-11-23 20:17:29	0d 19h 15m 22s	1/4	0
	LOAD	OK	2012-11-23 20:17:29	0d 19h 15m 22s	1/4	OK - load average: 0.00, 0.01, 0.05
	PING	OK	2012-11-23 20:15:05	0d 19h 28m 54s	1/4	PING OK - Packet loss = 0%, RTA = 0.50 ms
	RAM	OK	2012-11-23 20:17:29	0d 19h 15m 22s	1/4	32 162/492
	SSH	OK	2012-11-23 20:16:52	0d 19h 27m 7s	1/4	SSH OK - OpenSSH_5.9 (protocol 2.0)
3d3941aa5f674861a04fd0314da4be19_i-0000000e_Cumulus	HTTP_CONNECTIONS	OK	2012-11-23 20:18:30	0d 17h 58m 55s	1/4	0
	LOAD	OK	2012-11-23 20:18:30	0d 17h 58m 55s	1/4	OK - load average: 0.00, 0.01, 0.05
	PING	OK	2012-11-23 20:14:15	0d 17h 59m 44s	1/4	PING OK - Packet loss = 0%, RTA = 0.57 ms
	RAM	OK	2012-11-23 20:18:30	0d 17h 58m 55s	1/4	60 298/492
	SSH	OK	2012-11-23 20:16:08	0d 17h 57m 51s	1/4	SSH OK - OpenSSH_5.9 (protocol 2.0)
localhost	Current Load	OK	2012-11-23 20:18:46	17d 6h 20m 51s	1/4	OK - load average: 0.88, 0.65, 0.60
	Current Users	OK	2012-11-23 20:13:59	17d 6h 20m 1s	1/4	USERS OK - 0 users currently logged in
	Disk Space	CRITICAL	2012-11-23 20:17:48	17d 6h 19m 11s	4/4	DISK CRITICAL - /var/lib/lightdm/.gvfs is not accessible: Permission denied
	HTTP	OK	2012-11-23 20:15:45	17d 6h 18m 21s	1/4	HTTP OK: HTTP/1.1 200 OK - 2214 bytes in 0.021 second response time
	SSH	OK	2012-11-23 20:16:28	17d 6h 17m 31s	1/4	SSH OK - OpenSSH_5.9p1 Debian-5ubuntu1 (protocol 2.0)
	Total Processes	OK	2012-11-23 20:18:32	3d 23h 34m 41s	1/4	PROCS OK: 193 processes

Figura 5.31 – Nagios: Status dos serviços monitorados em cada uma das VMs.

5.6 Conclusão

Neste capítulo foram apresentados os passos efetuados para realizar a implementação do PCMONS com o OpenStack, bem como o motivo da seleção das ferramentas e dos sistemas operacionais. Foi apresentado também a implementação realizada no Laboratório de Rede e Gerência que foi feita para simular um caso de uso.

Capítulo 6

Conclusão

A computação em nuvem vem ganhando força no mercado, cada vez mais empresas tem migrado de forma parcial ou integral para um ambiente de computação em nuvem, utilizando infraestruturas públicas e/ou privadas para atender a necessidade de seus negócios. Com isso administradores e gerentes de tecnologia da informação necessitam de ferramentas para monitorar e manter esses ambientes.

Na área de monitoramento de nuvens privadas pouco se tem feito, é possível encontrar algumas ferramentas que fazem o monitoramento de ferramentas de IaaS especificas, mas quase nenhuma com o propósito genérico do PCMONS, já que a maioria dos esforços estão voltados para implementação de tecnologias de fornecimento e padronização.

O OpenStack se apresenta como uma das principais ferramentas para implementação em ambiente de nuvem no modelo IaaS, ganhando força, desenvolvedores e empresas parceiras. E por ser uma ferramenta de código aberto e com o grande suporte da comunidade internacional ela se mostrou uma ferramenta de fácil utilização e baixo custo de implementação no estudo de caso utilizado neste trabalho.

Este trabalho comprovou mais uma vez a facilidade de extensão do PCMONS, realizando sua integração com o OpenStack. Essa facilidade é decorrente de sua estrutura modular proposta e desenvolvida por [CHA 10]. O código fonte do PCMONS está disponível sob a licença GPL (General Public License ou Licença Pública Geral) e pode ser encontrado em http://code.google.com/p/pcmons/.

Referências

[CHA 10] CHAVES, S. Arquitetura e Sistema de Monitoramento para Computação em Nuvem Privada. Florianópolis: Universidade Federal De Santa Catarina, Instituto de Informática e Estatística, Programa de Pós-Graduação em Ciências da Computação, 2010. Dissertação de Mestrado

[DC 11] DE CHAVES, S.; URIARTE, R.; WESTPHALL, C. Toward an architecture for monitoring private clouds. Communications Magazine, IEEE, [S.l.], v.49, n.12, p.130 –137, december, 2011.

[DOC 12] Documentação do OpenStack. Disponível em <http://docs.openstack.org/> Acesso em: 23 de Novembro de 2012.

[HUR 10] HURWITZ, Judith. **Cloud Computing for DUMMIES.** Wiley, Indianapolis, Indiana, USA, 2010.

[KVM 13] Site do KVM. Disponível em <http://www.linux-kvm.org/page/Main_Page> Acesso em: 13/02/2013

[LOS 12] Instalação passo a passo do OpenStack Essex. Disponível em <http://www.hastexo.com/resources/docs/installing-openstack-essex-20121-ubuntu-1204-precise-pangolin> Acesso em: 23 de Novembro de 2012.

[NAG 13] Site do Nagios. Disponível em <http://http://www.nagios.org/> Acesso em: 14/02/2013

[TAU 03] TAURION, Cezar. **Cloud Computing - Computação em Nuvem.** Rio de Janeiro: Brasport, 2009.

[VIT 12] VITTI, Pedro A. F. **Integração do PCMONS com o OpenNebula para Gerência e Monitoramento de Nuvens Privadas** 2012. 69p. Trabalho de conclusão de curso do curso de Ciência da Computação. UFSC. Florianópolis - SC.

Apêndice A

keystone_data.sh

```
#!/bin/bash
#
# Initial data for Keystone using python-keystoneclient
#
# Tenant               User      Roles
# -------------------------------------------------------------------
# admin                admin     admin
# service              glance    admin
# service              nova      admin, [ResellerAdmin (swift only)]
# service              quantum   admin        # if enabled
# service              swift     admin        # if enabled
# demo                 admin     admin
# demo                 demo      Member, anotherrole
# invisible_to_admin   demo      Member
#
# Variables set before calling this script:
# SERVICE_TOKEN - aka admin_token in keystone.conf
# SERVICE_ENDPOINT - local Keystone admin endpoint
# SERVICE_TENANT_NAME - name of tenant containing service accounts
# ENABLED_SERVICES - stack.sh's list of services to start
# DEVSTACK_DIR - Top-level DevStack directory

ADMIN_PASSWORD=${ADMIN_PASSWORD:-testelab}
SERVICE_PASSWORD=${SERVICE_PASSWORD:-$ADMIN_PASSWORD}
export SERVICE_TOKEN="testelab"
export SERVICE_ENDPOINT="http://localhost:35357/v2.0"
SERVICE_TENANT_NAME=${SERVICE_TENANT_NAME:-service}
ENABLED_SERVICES="swift"

function get_id () {
    echo `$@ | awk '/ id / { print $4 }'`
}

# Tenants
ADMIN_TENANT=$(get_id keystone tenant-create --name=admin)
```

```
SERVICE_TENANT=$(get_id keystone tenant-create --name=$SERVICE_TENANT_NAME)
DEMO_TENANT=$(get_id keystone tenant-create --name=demo)
INVIS_TENANT=$(get_id keystone tenant-create --name=invisible_to_admin)


# Users
ADMIN_USER=$(get_id keystone user-create --name=admin \
                                          --pass="$ADMIN_PASSWORD" \
                                          --email=admin@hastexo.com)
DEMO_USER=$(get_id keystone user-create --name=demo \
                                         --pass="$ADMIN_PASSWORD" \
                                         --email=demo@hastexo.com)


# Roles
ADMIN_ROLE=$(get_id keystone role-create --name=admin)
KEYSTONEADMIN_ROLE=$(get_id keystone role-create --name=KeystoneAdmin)
KEYSTONESERVICE_ROLE=$(get_id keystone role-create
--name=KeystoneServiceAdmin)
# ANOTHER_ROLE demonstrates that an arbitrary role may be created and used
# TODO(sleepsonthefloor): show how this can be used for rbac in the future!
ANOTHER_ROLE=$(get_id keystone role-create --name=anotherrole)


# Add Roles to Users in Tenants
keystone user-role-add --user $ADMIN_USER --role $ADMIN_ROLE --tenant_id
$ADMIN_TENANT
keystone user-role-add --user $ADMIN_USER --role $ADMIN_ROLE --tenant_id
$DEMO_TENANT
keystone user-role-add --user $DEMO_USER --role $ANOTHER_ROLE --tenant_id
$DEMO_TENANT

# TODO(termie): these two might be dubious
keystone user-role-add --user $ADMIN_USER --role $KEYSTONEADMIN_ROLE
--tenant_id $ADMIN_TENANT
keystone user-role-add --user $ADMIN_USER --role $KEYSTONESERVICE_ROLE
--tenant_id $ADMIN_TENANT


# The Member role is used by Horizon and Swift so we need to keep it:
MEMBER_ROLE=$(get_id keystone role-create --name=Member)
keystone user-role-add --user $DEMO_USER --role $MEMBER_ROLE --tenant_id
$DEMO_TENANT
keystone user-role-add --user $DEMO_USER --role $MEMBER_ROLE --tenant_id
$INVIS_TENANT


# Configure service users/roles
NOVA_USER=$(get_id keystone user-create --name=nova \
                                         --pass="$SERVICE_PASSWORD" \
                                         --tenant_id $SERVICE_TENANT \
                                         --email=nova@hastexo.com)
```

```
keystone user-role-add --tenant_id $SERVICE_TENANT \
                       --user $NOVA_USER \
                       --role $ADMIN_ROLE

GLANCE_USER=$(get_id keystone user-create --name=glance \
                                          --pass="$SERVICE_PASSWORD" \
                                          --tenant_id $SERVICE_TENANT \
                                          --email=glance@hastexo.com)
keystone user-role-add --tenant_id $SERVICE_TENANT \
                       --user $GLANCE_USER \
                       --role $ADMIN_ROLE

if [[ "$ENABLED_SERVICES" =~ "swift" ]]; then
    SWIFT_USER=$(get_id keystone user-create --name=swift \
                                             --pass="$SERVICE_PASSWORD" \
                                             --tenant_id $SERVICE_TENANT \
                                             --email=swift@hastexo.com)
    keystone user-role-add --tenant_id $SERVICE_TENANT \
                           --user $SWIFT_USER \
                           --role $ADMIN_ROLE
    # Nova needs ResellerAdmin role to download images when accessing
    # swift through the s3 api. The admin role in swift allows a user
    # to act as an admin for their tenant, but ResellerAdmin is needed
    # for a user to act as any tenant. The name of this role is also
    # configurable in swift-proxy.conf
    RESELLER_ROLE=$(get_id keystone role-create --name=ResellerAdmin)
    keystone user-role-add --tenant_id $SERVICE_TENANT \
                           --user $NOVA_USER \
                           --role $RESELLER_ROLE
fi

if [[ "$ENABLED_SERVICES" =~ "quantum" ]]; then
    QUANTUM_USER=$(get_id keystone user-create --name=quantum \
                                               --pass="$SERVICE_PASSWORD" \
                                               --tenant_id
$SERVICE_TENANT \
                                               --email=quantum@hastexo.com)
    keystone user-role-add --tenant_id $SERVICE_TENANT \
                           --user $QUANTUM_USER \
                           --role $ADMIN_ROLE
fi
```

Apêndice B

endpoints.sh

```
#!/bin/sh

# Author:       Martin Gerhard Loschwitz
# (c) 2012      hastexo Professional Services GmbH

# Licensed under the Apache License, Version 2.0 (the "License");
# you may not use this file except in compliance with the License.
# You may obtain a copy of the License at
#
#    http://www.apache.org/licenses/LICENSE-2.0
#
# Unless required by applicable law or agreed to in writing, software
# distributed under the License is distributed on an "AS IS" BASIS,
# WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
# See the License for the specific language governing permissions and
# limitations under the License.
#
# On Debian-based systems the full text of the Apache version 2.0
# license can be found in `/usr/share/common-licenses/Apache-2.0'.

# MySQL definitions
MYSQL_USER=keystone
MYSQL_DATABASE=keystone
MYSQL_HOST=localhost

# other definitions
MASTER=localhost

while getopts "u:D:p:m:K:R:E:S:T:vh" opt; do
  case $opt in
    u)
```

```
      MYSQL_USER=$OPTARG
      ;;
    D)
      MYSQL_DATABASE=$OPTARG
      ;;
    p)
      MYSQL_PASSWORD=$OPTARG
      ;;
    m)
      MYSQL_HOST=$OPTARG
      ;;
    K)
      MASTER=$OPTARG
      ;;
    R)
      KEYSTONE_REGION=$OPTARG
      ;;
    E)
      export SERVICE_ENDPOINT=$OPTARG
      ;;
    S)
      SWIFT_MASTER=$OPTARG
      ;;
    T)
      export SERVICE_TOKEN=$OPTARG
      ;;
    v)
      set -x
      ;;
    h)
      cat <<EOF
Usage: $0 [-m mysql_hostname] [-u mysql_username] [-D mysql_database] [-p
mysql_password]
       [-K keystone_master ] [ -R keystone_region ] [ -E
keystone_endpoint_url ]
       [ -S swift_master ] [ -T keystone_token ]

Add -v for verbose mode, -h to display this message.
EOF
      exit 0
      ;;
    \?)
      echo "Unknown option -$OPTARG" >&2
      exit 1
      ;;
    :)
      echo "Option -$OPTARG requires an argument" >&2
      exit 1
      ;;
  esac
done
```

```
if [ -z "$KEYSTONE_REGION" ]; then
  echo "Keystone region not set. Please set with -R option or set
KEYSTONE_REGION variable." >&2
  missing_args="true"
fi

if [ -z "$SERVICE_TOKEN" ]; then
  echo "Keystone service token not set. Please set with -T option or set
SERVICE_TOKEN variable." >&2
  missing_args="true"
fi

if [ -z "$SERVICE_ENDPOINT" ]; then
  echo "Keystone service endpoint not set. Please set with -E option or set
SERVICE_ENDPOINT variable." >&2
  missing_args="true"
fi

if [ -z "$MYSQL_PASSWORD" ]; then
  echo "MySQL password not set. Please set with -p option or set
MYSQL_PASSWORD variable." >&2
  missing_args="true"
fi

if [ -n "$missing_args" ]; then
  exit 1
fi

keystone service-create --name nova --type compute --description 'OpenStack
Compute Service'
keystone service-create --name volume --type volume --description
'OpenStack Volume Service'
keystone service-create --name glance --type image --description 'OpenStack
Image Service'
keystone service-create --name swift --type object-store --description
'OpenStack Storage Service'
keystone service-create --name keystone --type identity --description
'OpenStack Identity'
keystone service-create --name ec2 --type ec2 --description 'OpenStack EC2
service'

create_endpoint () {
  case $1 in
    compute)
    keystone endpoint-create --region $KEYSTONE_REGION --service_id $2
--publicurl 'http://'"$MASTER"':8774/v2/%(tenant_id)s' --adminurl
'http://'"$MASTER"':8774/v2/%(tenant_id)s' --internalurl
'http://'"$MASTER"':8774/v2/%(tenant_id)s'
    ;;
    volume)
    keystone endpoint-create --region $KEYSTONE_REGION --service_id $2
--publicurl 'http://'"$MASTER"':8776/v1/%(tenant_id)s' --adminurl
```

```
'http://'"$MASTER"':8776/v1/%(tenant_id)s' --internalurl
'http://'"$MASTER"':8776/v1/%(tenant_id)s'
    ;;
    image)
    keystone endpoint-create --region $KEYSTONE_REGION --service_id $2
--publicurl 'http://'"$MASTER"':9292/v1' --adminurl
'http://'"$MASTER"':9292/v1' --internalurl 'http://'"$MASTER"':9292/v1'
    ;;
    object-store)
    if [ $SWIFT_MASTER ]; then
      keystone endpoint-create --region $KEYSTONE_REGION --service_id $2
--publicurl 'http://'"$SWIFT_MASTER"':8080/v1/AUTH_%(tenant_id)s'
--adminurl 'http://'"$SWIFT_MASTER"':8080/v1' --internalurl
'http://'"$SWIFT_MASTER"':8080/v1/AUTH_%(tenant_id)s'
    else
      keystone endpoint-create --region $KEYSTONE_REGION --service_id $2
--publicurl 'http://'"$MASTER"':8080/v1/AUTH_%(tenant_id)s' --adminurl
'http://'"$MASTER"':8080/v1' --internalurl
'http://'"$MASTER"':8080/v1/AUTH_%(tenant_id)s'
    fi
    ;;
    identity)
    keystone endpoint-create --region $KEYSTONE_REGION --service_id $2
--publicurl 'http://'"$MASTER"':5000/v2.0' --adminurl
'http://'"$MASTER"':35357/v2.0' --internalurl
'http://'"$MASTER"':5000/v2.0'
    ;;
    ec2)
    keystone endpoint-create --region $KEYSTONE_REGION --service_id $2
--publicurl 'http://'"$MASTER"':8773/services/Cloud' --adminurl
'http://'"$MASTER"':8773/services/Admin' --internalurl
'http://'"$MASTER"':8773/services/Cloud'
    ;;
  esac
}

for i in compute volume image object-store identity ec2; do
  id=`mysql -h "$MYSQL_HOST" -u "$MYSQL_USER" -p"$MYSQL_PASSWORD"
"$MYSQL_DATABASE" -ss -e "SELECT id FROM service WHERE type='"$i"';"` ||
exit 1
  create_endpoint $i $id
done
```

Apêndice C

Artigo

Integração do PCMONS com o OpenStack para Gerência e Monitoramento de Nuvens Privadas

Alec Augusto Gonçalves Ventura
Dr. Carlos Becker Westphall

***Abstract**. This project focuses primarily on the installation and monitoring of a private cloud using OpenStack and measure the performance of this new tool under development using PCMONS (Private Cloud Monitoring System) is necessary integration between the two.To test, validate the integration of components and presentation of results, will be deploying at the Network and Management Laboratory (LRG) at Federal University of Santa Catarina a private cloud model using OpenStack by monitoring virtual machines using PCMONS.*

***Keywords:** Cloud computing, OpenStack, PCMONS.*

***Resumo**. Este trabalho tem como foco principal a instalação e monitoramento de uma cloud privada utilizando o OpenStack e a medição do desempenho dessa nova ferramenta em desenvolvimento utilizando o PCMONS (Private Cloud Monitoring System) sendo necessário a integração entre ambos.Para teste, validação da integração dos componentes e apresentação dos resultados obtidos será implantando no Laboratório de Redes e Gerência (LRG) da Universidade Federal de Santa Catarina um modelo de nuvem privada utilizando o OpenStack monitorando-se as máquinas virtuais utilizando-se o PCMONS.*

***Palavras-chave:** Computação em nuvem, OpenStack, PCMONS.*

1 INTRODUÇÃO

No dia a dia utilizamos serviços básicos como água, eletricidade e telefone, e pagamos por esses recursos conforme utilizamos, ou seja, pagamos conforme nossas demandas pessoais ou empresariais. A computação em nuvem utiliza o mesmo conceito, fazendo com que o usuário pague pelos serviços de computação que ele utiliza.

A maior vantagem nesse modelo é disponibilidade dos recursos já que em um ambiente onde temos um servidor local, o desempenho dos serviços são limitados pela capacidade do servidor que rapidamente se torna obsoleto, já num ambiente onde se utiliza a computação em nuvem se for necessário maior alocação de recursos esses são disponibilizados dinamicamente, ou seja, sob demanda e são limitados apenas pela capacidade de um grupo de servidores com um potencial muito superior a um servidor local.

Assim, o presente trabalho tem como objetivo a criação e implementação de uma rede privada utilizando o OpenStack que é uma nova ferramenta open-source, para a análise de seu desempenho.

Segundo a definição do [NIST] a computação em nuvem permite o acesso à rede sob demanda a um conjunto compartilhado de recursos computacionais configuráveis (redes, servidores, armazenamento, aplicativos e serviços) que podem ser rapidamente provisionados e liberados com o esforço de gerenciamento mínimo ou interação com o provedor de serviços.

Se faz necessário uma ferramenta de monitoramento das novas tecnologias emergentes no mercado para que sejamos capazes de avaliar as melhores ferramentas e seus desempenhos.

A motivação para este trabalho vem da vontade de integrar o PCMONS[CHA 10] ao OpenStack já que esta nova ferramenta de Infraestrutura como Serviço (IaaS) carece deste tipo de funcionalidade. E também de trabalhar em uma ferramenta de monitoramento que aos poucos vem ganhando espaço na comunidade acadêmica com a implementação de novos módulos sendo desenvolvidos por graduandos, mestrandos e doutorandos.

2 COMPUTAÇÃO EM NUVEM

Hoje a computação em nuvem utiliza a Internet como um meio para a distribuição de seus recursos e serviços. Com a evolução de tecnologias como virtualização, computação em grade e computação distribuída a computação em nuvem é capaz de oferecer recursos e serviços com qualidade e segurança rapidamente e com mínimo esforço do provedor de serviço.

Uma definição mais simples seria dizer que a computação em nuvem é o fornecimento de recurso ou serviço sob demanda do cliente. O cliente tem a ilusão de disponibilidade de recursos infinitos diferentemente de se ter um servidor próprio em sua empresa limitado pelo *hardware.* A computação em nuvem permite que empresas usem recursos na quantidade que forem necessários, aumentando ou diminuindo a capacidade computacional dinamicamente.

Bancos de dados, redes, aplicativos, plataformas e até infraestruturas completas são alguns exemplos de recursos computacionais que podem ser fornecidos.

2.1 **Classificação quanto ao modelo de serviço**

Existem três classificações para os ambientes de computação em nuvem [NIST].

- Software como Serviço (SaaS).
- Plataforma como Serviço (PaaS).
- Infraestrutura como Serviço (IaaS).

2.1.1 Software como Serviço (SaaS)

É um modelo onde softwares de aplicativos são oferecidos por um provedor ou fornecedor de serviço e toda parte de gerenciamento e manutenção é de responsabilidade do provedor, o cliente apenas acessa e utiliza o serviço, sem a necessidade de instalação no cliente.

2.1.2. Plataforma como Serviço (PaaS)

PaaS é um modelo que oferece um conjunto de ferramentas, softwares e middlewares que um desenvolvedor necessita para fazer sua aplicação. O modelo permite que o desenvolvedor acompanhe todas as etapas do desenvolvimento, desde a criação dos casos de uso até a implementação e validação dos testes.

2.1.3 Infraestrutura como Serviço (IaaS)

Neste modelo é oferecido ao usuário a estrutura completa de hardware, desde processamento, armazenamento e rede. O cliente tem total controle sobre quais sistemas operacionais e tecnologias de virtualização estão instalados sem precisar se preocupar em administrar a infraestrutura.

Este modelo é referido também por alguns autores como Hardware as a Service (HaaS).

A figura 1 representa um diagrama de modelos de serviços.

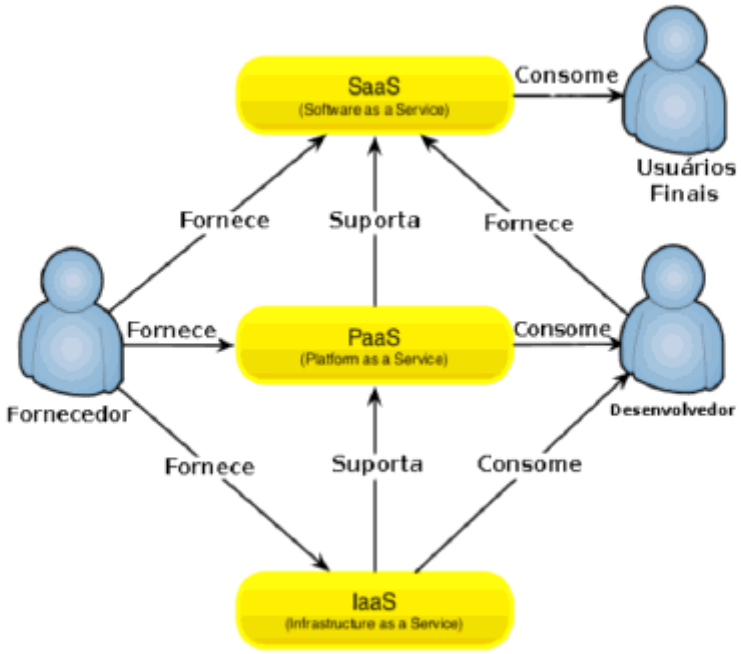


Figura 1 – Modelos de serviço em computação em nuvem. Inter-relação entre fornecedores e consumidores. [VIT 12].

2.1 Classificação quanto ao modelo de serviço

Além da classificação por modelo de serviço a computação em nuvem pode ser também classificada por modelo de implantação, nuvem pública, nuvem privada, nuvem híbrida e nuvem comunitária.

2.2.1 Nuvem Pública

Neste modelo de nuvem o prestador oferece seu serviço pela Internet, pode ser gratuito ou utilizando um modelo de pagamento baseado no uso (pay-per-use). A infraestrutura da *cloud* é projetada para atender a muitos usuários onde eles tem o mesmo nível de acesso aos recursos.

Um dos benefícios da nuvem pública é que elas tendem a ser muito maiores que as redes privadas e portanto mantém uma maior quantidade de recursos, isso permite uma maior escalabilidade de recursos. Essa característica evita a compra de equipamentos adicionais para resolver alguma necessidade temporária.

2.2.2 Nuvem Privada

As nuvens privadas são aquelas que são construídas para uma única empresa, normalmente optam por esse tipo de nuvem as empresas que estão mais preocupadas com a segurança ou a garantia de disponibilidade sem o atraso da Internet. A infraestrutura pode ser administrada pela própria empresa, por terceiros ou por ambos e ela pode existir dentro ou fora das instalações da empresa.

Ferramentas como Eucalyptus, OpenNebula e OpenStack permitem a implantação desse tipo de modelo.

2.2.3 Nuvem Híbrida

Nuvens híbridas combinam os recursos das redes públicas e privadas, ela permite que redes privadas tenham uma reserva de recursos acessíveis via rede pública, isso permite manter o nível dos serviços mesmo que haja uma flutuação rápida da demanda. Uma empresa também pode escolher utilizar uma rede privada apenas para armazenar seus dados críticos, garantindo assim a segurança de seus dados em data centers dentro do domínio da própria empresa.

2.2.4 Nuvem Comunitária

Nuvens comunitárias são *clouds* mantidas por um conjunto de empresas que normalmente tem a mesma área de atuação e preocupações como segurança, política, entre outros. A nuvem pode ser administrada pelas empresas ou por um terceiro e ela pode existir no ambiente da empresa ou fora dele.

3 O OPENSTACK

OpenStack é uma colaboração global de desenvolvedores e tecnólogos de computação em nuvem que produzem em código aberto implementações para nuvens públicas e privadas. O projeto visa oferecer soluções para todos os tipos de nuvens por ser simples de implementar, altamente escalável, e rico em recursos. A ferramenta consiste em uma série de projetos inter- relacionados que oferecem vários componentes como solução de infraestrutura de nuvem. Suporta a implementação de nuvens privadas, públicas ou híbridas e oferece uma interface compatível com as interfaces EC2 e S3 da Amazon AWS.

No presente trabalho foi utilizado o projeto com codinome Essex, que no presente momento é o mais atual [OPE 12].

A tabela 1 apresenta todas as versões do OpenStack.

Nome do release	**Data de lançamento**
Austin	21 de Outubro de 2010
Bexar	03 de Fevereiro de 2011
Cactus	15 de Abril de 2011
Diablo	22 de Setembro de 2011
Essex	05 de Abril de 2012
Folsom	Estimado para ser lançado no final de 2012

Tabela 3.1 - nome e a data dos releases[1].

[1]http://docs.openstack.org/essex/openstack-compute/install/apt/content/version.html

3.1 Arquitetura

O OpenStack foi desenhado de forma modular e é basicamente formado de três grandes projetos de software livre que formam o seu núcleo: Nova, Glance e Swift. Existem ainda projetos adicionais, Horizon e Keystone, que foram utilizados no presente trabalho.

A figura 2 detalha a arquitetura do OpenStack e a inter-relação entre seus módulos.

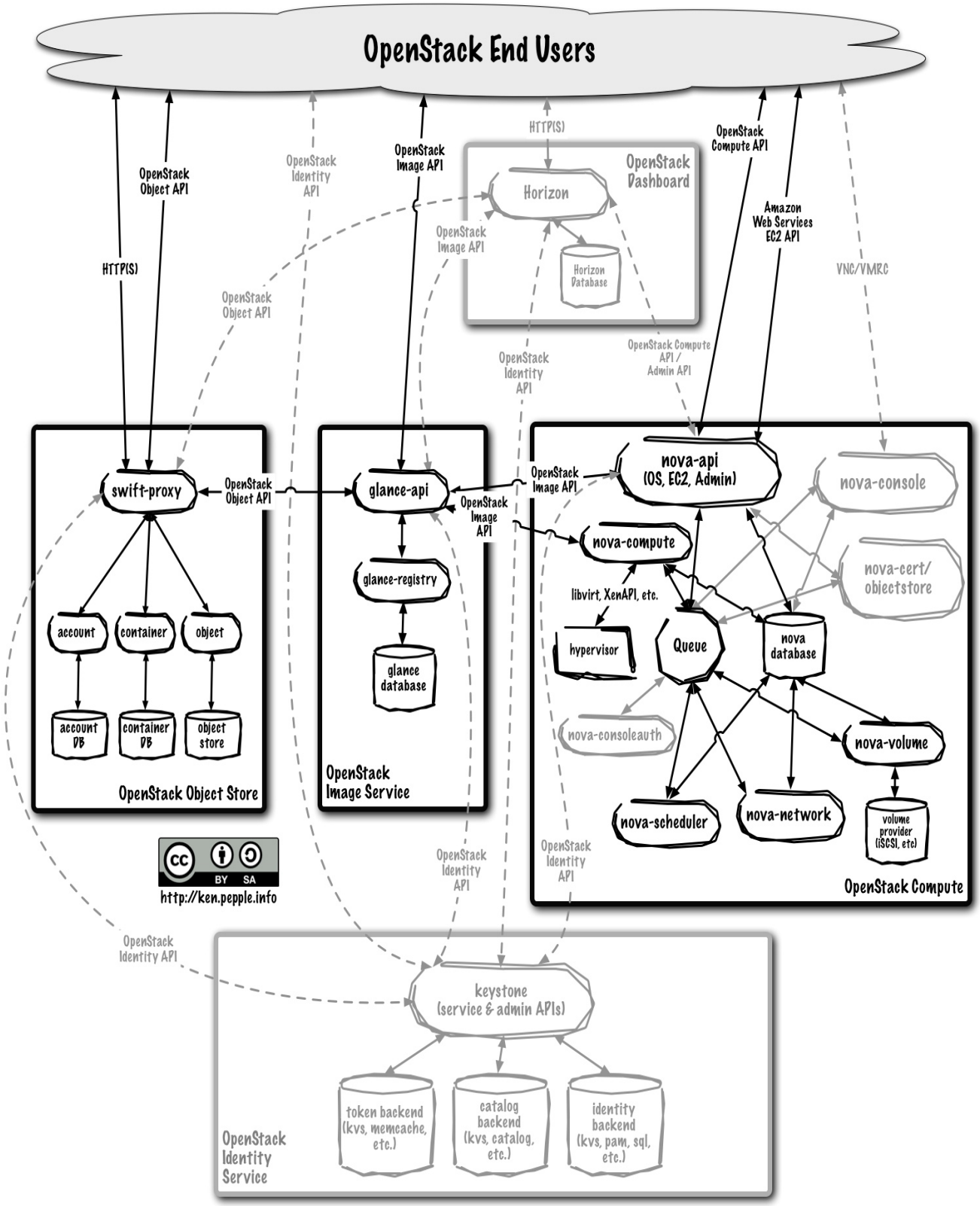


Figura 2 – Arquitetura do OpenStack Essex [DOC 12].

3.1.1 Nova

Nova é a aplicação que gerencia toda a infraestrutura do OpenStack, todas as atividades necessárias para manter o ciclo de vida das instâncias de uma nuvem OpenStack são controlados pelo Nova. Ele gerencia todos os recursos computacionais como a rede, autorização, escalabilidade e recursos computacionais da nuvem. O Nova em si não possui nenhuma capacidade de virtualização, ao invés disso ele faz o uso de APIs para interagir com os *hypervisors* suportados.

No presente trabalho foi utilizado como *hypervisor* o KVM [KVM 12].

3.1.2 Glance

O Glance é o serviço de gerenciamento de imagens do OpenStack, é um sistema de busca, controle e armazenamento das imagens de máquinas virtuais da cloud.

Ele pode ser configurado para fazer isso localmente utilizando o Swift, remotamente utilizando o Amazon S3 diretamente, ou utilizando o Amazon S3 com o Swift como intermediário.

Ele suporta uma grande variedade de padrões de imagens, incluindo VDI (VirtualBox), VHD (Microsoft Hyper-V), QCOW2 (QEMU/KVM), VMDK/OVF (VMware) e bruto.

3.1.3 Swift

Equivalente ao serviço S3 da Amazon (Simple Storage Service), o Swift implementa um sistema de armazenamento de objetos provendo redundância e tolerância a falhas, ele é extremamente escalável tanto em termos de tamanho (vários petabytes) como capacidade (bilhões de objetos).

3.1.4 Keystone

É um projeto que provê um único ponto de integração para geração e gerenciamento de identidades, tokens, catálogos e políticas para os serviços da família OpenStack. Ele implementa a API de identidade do OpenStack

3.1.5 Horizon

É a interface do OpenStack, apresentando um dashboad de fácil visualização e manuseio, o Horizon fornece algumas opções para que seja possível gerenciar e monitorar a cloud de forma simples. Ela permite visualizar, criar, destruir e gerenciar VMs, instâncias, a rede, usuários e algumas outras funcionalidades básicas.

4 MONITORAMENTO EM COMPUTAÇÃO EM NUVEM E O PCMONS

O monitoramento é o processo de colher, analisar e apresentar informações. Para se analisar cada recurso é necessário que se tenha técnicas adequadas para obter informações sobre cada um deles.

4.1 NAGIOS

Nagios é um sistema baseado em Unix e uma ferramenta de monitoramento de praticamente qualquer serviço que possa ser conectado a rede [NAG 13]. Ele possui uma licença GPL e vem sendo desenvolvido há mais de 10 anos e portanto possui ampla documentação e suporte da comunidade de código aberto. Ele é composto por três partes: seu núcleo, plugins e sua interface web. Os plugins são responsáveis por realizar as verificações e repassar os estados ao Nagios.

4.2 PCMONS

PCMONS (Private Cloud Monitoring System) ou em português, Sistema de Monitoramento de Nuvens Privadas, é uma ferramenta modular e flexível desenvolvida por [CHA 10].

Após a realização deste trabalho o PCMONS garante compatibilidade com o Nagios em sua camada de visualização, e com o OpenStack, OpenNebula e Eucalyptus em sua camada de infraestrutura. O PCMONS basicamente coleta e prepara os dados para a camada de visualização e ele é dividido em vários módulos para simplificar futuras adaptações para ferramentas específicas e para facilitar seu estudo e aplicação, de acordo com [DC 11].

Em [VIT 12] ele específica cada um dos módulos listados abaixo:

Coletor de Informações do Nó (Node Information Gatherer) - Este módulo é responsável por coletar as informações locais em cada um dos nós da nuvem. Reúne informações sobre as máquinas virtuais locais e envia para o Data Integrator Cluster.

Cluster Integrador de Dados (Data Integrator Cluster) - Como a maioria das ferramentas organiza seus nós em clusters, há um agente específico que reúne e prepara os dados para a próxima camada. Este agente evita a transferência de dados desnecessários de cada nó para o Monitoring Data Integrator.

Integrador de Dados do Monitoramento (Monitoring Data Integrator) - Coleta e armazena dados da nuvem no banco de dados para fins históricos, e fornece esses dados para o Configuration Generator.

Monitor de Máquina Virtual (Virtual Machine Monitor) - Este módulo injeta scripts nas máquinas virtuais que enviam dados úteis a partir dela para o sistema de monitoramento. Exemplos desses dados são carga do processador e da memória.

Gerador de Configuração (Configuration Generator) - Recupera informações do banco de dados, por exemplo, para gerar

os arquivos de configurações necessários para ferramentasde visualização a serem utilizadas na camada de visualização.

Servidor de Monitoramento (Monitoring Tool Server) - Este módulo é responsável por receber dados de monitoramento de recursos diferentes (por exemplo, do Monitor de Máquinas Virtuais). Seu objetivo é receber informações de monitoramento e tomar ações como armazená- las no módulo de banco de dados para fins históricos.

Interface de Usuário (User Interface) - O PCMONS utiliza a própria interface da ferramenta Nagios como interface do usuário.

Banco de Dados (Database) - Armazena dados necessários para o Gerador de Configuração e do Integrador de Dados do Monitoramento.

A figura 3 mostra os módulos descritos e a interação entre eles.

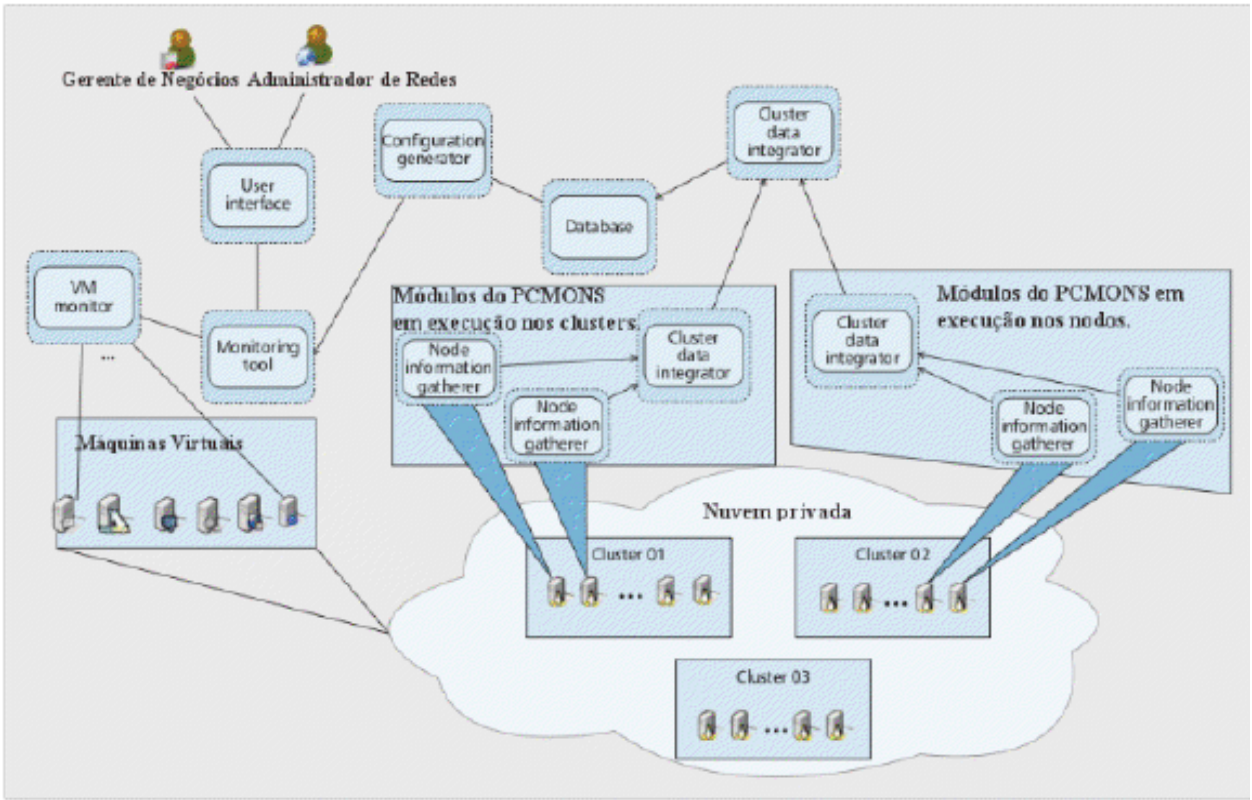


Figura 3 – Arquitetura da ferramenta PCMONS e interação entre seus componentes. [VIT 12].

O PCMONS funciona da seguinte maneira: a ferramenta colhe informações básicas sobre as máquinas virtuais como seu endereço IP, o usuário que a instanciou, e seu número de identificação e envia para a camada de visualização que em seguida cria os arquivos necessários para a monitoração destas máquinas virtuais. Neste processo é feito também o mapeamento entre a máquina virtual e seu host, característica importante do PCMONS já que para as ferramentas disponíveis para o gerenciamento de sistemas do tipo IaaS, esse mapeamento é abstraído, dificultando assim a resolução de problemas [URI 10].

5 ESTUDO DE CASO

Foi implementado no LRG (Laboratório de Rede e Gerencia) da Universidade Federal de Santa Catarina um ambiente de computação em nuvem utilizando a ferramenta de IaaS chamada OpenStack. O OpenStack foi escolhido como ferramenta de implementação por ser de código aberto e por estar conquistando o topo entre as melhores ferramentas para implementação em nuvem no ramo de IaaS.

5.1 Implementação

Foi necessário estender alguns módulos do PCMONS para que este suportasse o monitoramento do OpenStack. As alterações de código feitas em cada um dos módulos foram:

Cluster Integrador de Dados

- Arquivo de configuração do agente;
- Mapeamento dos nós de acordo com a ferramenta de IaaS utilizada;
- Atualização das informações das máquinas virtuais que estão sendo gerenciadas.

Coletor de informações do Nó

- Obtenção de informações das máquinas virtuais locais e envio para o Cluster Integrador de Dados.

Adaptação para nova versão da biblioteca BOTO (python – AWS)

- Mudanças no código para adequar o PCMONS ao BOTO.

5.2 Ambiente

OpenStack permite que em uma mesma máquina sejam instanciados todos os serviços necessários para a infraestrutura da nuvem, tais como: controlador da nuvem, controlador de imagem, armazenamento de imagem e serviço de autenticação. A figura 4 descreve os principais hardwares e softwares utilizados para a implementação deste trabalho.

Hardware	Software
AMD Phenom(tm) II X4 965 Processor 4GB DDR2 667MHz HD Samsung HD502HJ 500GB ATA	Ubuntu 12.04 OpenStack Essex Nagios Core 3.2.0

Figura 4 – Hardwares e Softwares utilizados no estudo de caso.

O hipervisor escolhido foi o KVM, pois como o hardware utilizado apresenta suporte a virtualização, foi possível efetuar a virtualização completa, não sendo necessário assim efetuar nenhuma alteração no kernel do sistema operacional.

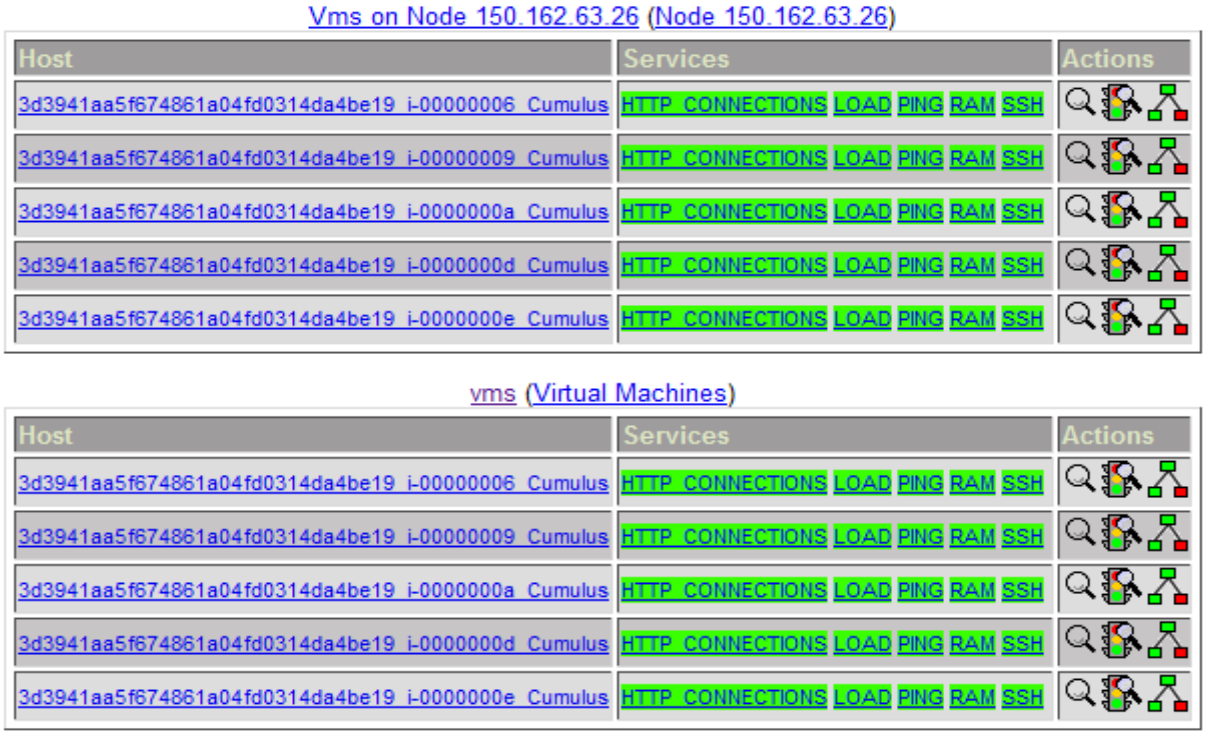
5.3 Resultados, Nagios e PCMONS

A figura 5 mostra a interface da ferramenta Nagios sendo utilizada pelo PCMONS, que mostra a lista de todas as instâncias de máquinas virtuais sendo utilizadas no momento. A nomenclatura utilizada para representar cada uma delas é composta do identificador do usuário que iniciou a VM, do identificador da VM, e o nome da máquina física onde está a máquina virtual.

Host	Status	Last Check	Duration	Status Information
3d3941aa5f674861a04fd0314da4be19_i-00000006_Cumulus	UP	2012-11-23 20:04:53	0d 19h 18m 21s	PING OK - Packet loss = 0%, RTA = 0.56 ms
3d3941aa5f674861a04fd0314da4be19_i-00000009_Cumulus	UP	2012-11-23 20:05:53	0d 19h 18m 0s	PING OK - Packet loss = 0%, RTA = 0.58 ms
3d3941aa5f674861a04fd0314da4be19_i-0000000a_Cumulus	UP	2012-11-23 20:06:53	0d 19h 16m 48s	PING OK - Packet loss = 0%, RTA = 0.42 ms
3d3941aa5f674861a04fd0314da4be19_i-0000000d_Cumulus	UP	2012-11-23 20:02:43	0d 19h 17m 38s	PING OK - Packet loss = 0%, RTA = 0.48 ms
3d3941aa5f674861a04fd0314da4be19_i-0000000e_Cumulus	UP	2012-11-23 20:06:53	0d 17h 48m 28s	PING OK - Packet loss = 0%, RTA = 0.48 ms
localhost	UP	2012-11-23 20:03:13	17d 6h 9m 35s	PING OK - Packet loss = 0%, RTA = 0.11 ms

Figura 5 – Nagios: Lista das máquinas virtuais monitoradas pelo sistema.

Neste estudo de caso as informações monitoradas pelo PCMONS englobam dois conjuntos de monitoramento, um em relação ao serviço que está sendo executado (número de conexões HTTP), e outro em relação ao desempenho e disponibilidade da máquina virtual (memória, ssh, ping e carga). A figura 6 mostra todos esses serviços passivos (HTTP CONNECTIONS, LOAD, RAM) e ativos (PING, SSH) que estão sendo monitorados pelo PCMONS.



Vms on Node 150.162.63.26 (Node 150.162.63.26)

Host	Services	Actions
3d3941aa5f674861a04fd0314da4be19_i-00000006_Cumulus	HTTP_CONNECTIONS LOAD PING RAM SSH	
3d3941aa5f674861a04fd0314da4be19_i-00000009_Cumulus	HTTP_CONNECTIONS LOAD PING RAM SSH	
3d3941aa5f674861a04fd0314da4be19_i-0000000a_Cumulus	HTTP_CONNECTIONS LOAD PING RAM SSH	
3d3941aa5f674861a04fd0314da4be19_i-0000000d_Cumulus	HTTP_CONNECTIONS LOAD PING RAM SSH	
3d3941aa5f674861a04fd0314da4be19_i-0000000e_Cumulus	HTTP_CONNECTIONS LOAD PING RAM SSH	

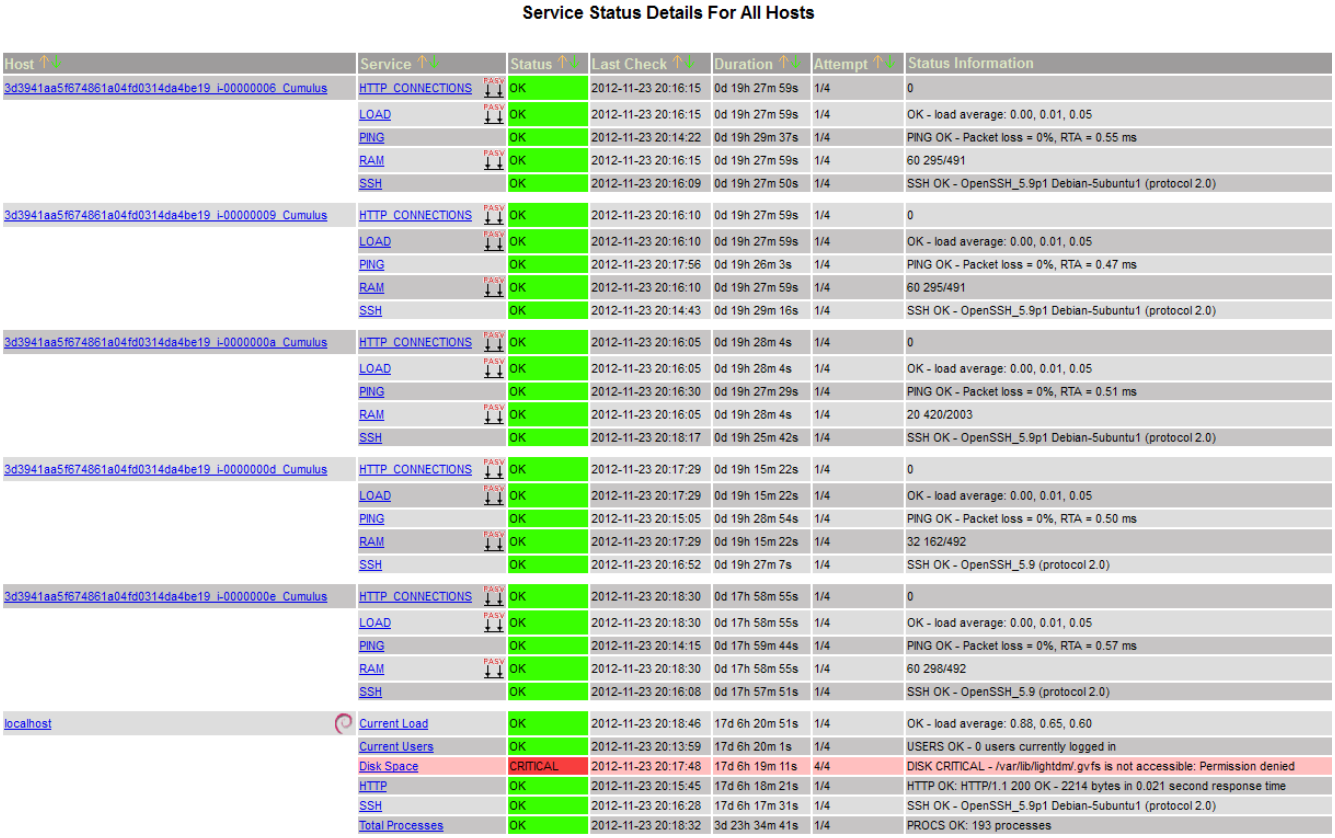
vms (Virtual Machines)

Host	Services	Actions
3d3941aa5f674861a04fd0314da4be19_i-00000006_Cumulus	HTTP_CONNECTIONS LOAD PING RAM SSH	
3d3941aa5f674861a04fd0314da4be19_i-00000009_Cumulus	HTTP_CONNECTIONS LOAD PING RAM SSH	
3d3941aa5f674861a04fd0314da4be19_i-0000000a_Cumulus	HTTP_CONNECTIONS LOAD PING RAM SSH	
3d3941aa5f674861a04fd0314da4be19_i-0000000d_Cumulus	HTTP_CONNECTIONS LOAD PING RAM SSH	
3d3941aa5f674861a04fd0314da4be19_i-0000000e_Cumulus	HTTP_CONNECTIONS LOAD PING RAM SSH	

Figura 6 – Nagios: Serviços sendo monitorados pelo sistema.

Na figura 6 também mostra que o Nagios faz o agrupamento de máquinas virtuais por nodo, facilitando assim a identificação de falhas no caso de um nodo com problemas.

O Nagios ainda apresenta uma tela com as informações detalhadas de cada serviço de todas as máquinas virtuais, como mostra a figura 7:



Service Status Details For All Hosts

Host	Service	Status	Last Check	Duration	Attempt	Status Information
3d3941aa5f674861a04fd0314da4be19_i-00000006_Cumulus	HTTP_CONNECTIONS	OK	2012-11-23 20:16:15	0d 19h 27m 59s	1/4	0
	LOAD	OK	2012-11-23 20:16:15	0d 19h 27m 59s	1/4	OK - load average: 0.00, 0.01, 0.05
	PING	OK	2012-11-23 20:14:22	0d 19h 29m 37s	1/4	PING OK - Packet loss = 0%, RTA = 0.55 ms
	RAM	OK	2012-11-23 20:16:15	0d 19h 27m 59s	1/4	60 295/491
	SSH	OK	2012-11-23 20:16:09	0d 19h 27m 50s	1/4	SSH OK - OpenSSH_5.9p1 Debian-5ubuntu1 (protocol 2.0)
3d3941aa5f674861a04fd0314da4be19_i-00000009_Cumulus	HTTP_CONNECTIONS	OK	2012-11-23 20:16:10	0d 19h 27m 59s	1/4	0
	LOAD	OK	2012-11-23 20:16:10	0d 19h 27m 59s	1/4	OK - load average: 0.00, 0.01, 0.05
	PING	OK	2012-11-23 20:17:56	0d 19h 26m 3s	1/4	PING OK - Packet loss = 0%, RTA = 0.47 ms
	RAM	OK	2012-11-23 20:16:10	0d 19h 27m 59s	1/4	60 295/491
	SSH	OK	2012-11-23 20:14:43	0d 19h 29m 16s	1/4	SSH OK - OpenSSH_5.9p1 Debian-5ubuntu1 (protocol 2.0)
3d3941aa5f674861a04fd0314da4be19_i-0000000a_Cumulus	HTTP_CONNECTIONS	OK	2012-11-23 20:16:05	0d 19h 28m 4s	1/4	0
	LOAD	OK	2012-11-23 20:16:05	0d 19h 28m 4s	1/4	OK - load average: 0.00, 0.01, 0.05
	PING	OK	2012-11-23 20:16:30	0d 19h 27m 29s	1/4	PING OK - Packet loss = 0%, RTA = 0.51 ms
	RAM	OK	2012-11-23 20:16:05	0d 19h 28m 4s	1/4	20 420/2003
	SSH	OK	2012-11-23 20:18:17	0d 19h 25m 42s	1/4	SSH OK - OpenSSH_5.9p1 Debian-5ubuntu1 (protocol 2.0)
3d3941aa5f674861a04fd0314da4be19_i-0000000d_Cumulus	HTTP_CONNECTIONS	OK	2012-11-23 20:17:29	0d 19h 15m 22s	1/4	0
	LOAD	OK	2012-11-23 20:17:29	0d 19h 15m 22s	1/4	OK - load average: 0.00, 0.01, 0.05
	PING	OK	2012-11-23 20:15:05	0d 19h 28m 54s	1/4	PING OK - Packet loss = 0%, RTA = 0.50 ms
	RAM	OK	2012-11-23 20:17:29	0d 19h 15m 22s	1/4	32 162/492
	SSH	OK	2012-11-23 20:16:52	0d 19h 27m 7s	1/4	SSH OK - OpenSSH_5.9 (protocol 2.0)
3d3941aa5f674861a04fd0314da4be19_i-0000000e_Cumulus	HTTP_CONNECTIONS	OK	2012-11-23 20:18:30	0d 17h 58m 55s	1/4	0
	LOAD	OK	2012-11-23 20:18:30	0d 17h 58m 55s	1/4	OK - load average: 0.00, 0.01, 0.05
	PING	OK	2012-11-23 20:14:15	0d 17h 59m 44s	1/4	PING OK - Packet loss = 0%, RTA = 0.57 ms
	RAM	OK	2012-11-23 20:18:30	0d 17h 58m 55s	1/4	60 298/492
	SSH	OK	2012-11-23 20:16:08	0d 17h 57m 51s	1/4	SSH OK - OpenSSH_5.9 (protocol 2.0)
localhost	Current Load	OK	2012-11-23 20:18:46	17d 6h 20m 51s	1/4	OK - load average: 0.88, 0.65, 0.60
	Current Users	OK	2012-11-23 20:13:59	17d 6h 20m 1s	1/4	USERS OK - 0 users currently logged in
	Disk Space	CRITICAL	2012-11-23 20:17:48	17d 6h 19m 11s	4/4	DISK CRITICAL - /var/lib/lightdm/.gvfs is not accessible: Permission denied
	HTTP	OK	2012-11-23 20:15:45	17d 6h 18m 21s	1/4	HTTP OK: HTTP/1.1 200 OK - 2214 bytes in 0.021 second response time
	SSH	OK	2012-11-23 20:16:28	17d 6h 17m 31s	1/4	SSH OK - OpenSSH_5.9p1 Debian-5ubuntu1 (protocol 2.0)
	Total Processes	OK	2012-11-23 20:18:32	3d 23h 34m 41s	1/4	PROCS OK: 193 processes

Figura 7 – Nagios: Status dos serviços monitorados em cada uma das VMs.

6 CONCLUSÃO

Na área de monitoramento de nuvens privadas pouco se tem feito, é possível encontrar algumas ferramentas que fazem o monitoramento de ferramentas de IaaS especificas, mas quase nenhuma com o propósito genérico do PCMONS, já que a maioria dos esforços estão voltados para implementação de tecnologias de fornecimento e padronização.

Este trabalho comprovou mais uma vez a facilidade de extensão do PCMONS, realizando sua integração com o OpenStack. Essa facilidade é decorrente de sua estrutura modular proposta e desenvolvida por [CHA 10]. O código fonte do PCMONS está disponível sob a licença GPL (General Public License ou Licença Pública Geral) e pode ser encontrado em http://code.google.com/p/pcmons/.

REFERÊNCIAS

[CHA 10] CHAVES, S. Arquitetura e Sistema de Monitoramento para Computação em Nuvem Privada. Florianópolis: Universidade Federal De Santa Catarina, Instituto de Informática e Estatística, Programa de Pós-Graduação em Ciências da Computação, 2010. Dissertação de Mestrado

[DC 11] DE CHAVES, S.; URIARTE, R.; WESTPHALL, C. Toward an architecture for monitoring private clouds. Communications Magazine, IEEE, [S.l.], v.49, n.12, p.130 –137, december, 2011.

[DOC 12] Documentação do OpenStack. Disponível em <http://docs.openstack.org/> Acesso em: 23 de Novembro de 2012.

[HUR 10] HURWITZ, Judith. **Cloud Computing for DUMMIES.** Wiley, Indianapolis, Indiana, USA, 2010.

[KVM 13] Site do KVM. Disponível em <http://www.linux-kvm.org/page/Main_Page> Acesso em: 13/02/2013

[LOS 12] Instalação passo a passo do OpenStack Essex. Disponível em <http://www.hastexo.com/resources/docs/installing-openstack-essex-20121-ubuntu-1204-precise-pangolin> Acesso em: 23 de Novembro de 2012.

[NAG 13] Site do Nagios. Disponível em <http://http://www.nagios.org/> Acesso em: 14/02/2013

[TAU 03] TAURION, Cezar. **Cloud Computing - Computação em Nuvem.** Rio de Janeiro: Brasport, 2009.

[VIT 12] VITTI, Pedro A. F. **Integração do PCMONS com o OpenNebula para Gerência e Monitoramento de Nuvens Privadas** 2012. 69p. Trabalho de conclusão de curso do curso de Ciência da Computação. UFSC. Florianópolis - SC.