

**UNIVERSIDADE FEDERAL DE SANTA CATARINA
DEPARTAMENTO DE INFORMÁTICA E ESTATÍSTICA
BACHARELADO EM CIÊNCIA DA COMPUTAÇÃO**

Pedro Artur Figueiredo Vitti

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

Prof. Dr. Carlos Becker Westphall
Orientador

Bel. Rafael Brundo Uriarte
Co-Orientador

Florianópolis, junho de 2012

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

Pedro Artur Figueiredo Vitti

Este Trabalho de Conclusão de Curso foi julgado adequado para a obtenção do título de Bacharel em Ciência da Computação e aprovada em sua forma final pelo Departamento de Informática e Estatística da Universidade Federal de Santa Catarina.

Banca Examinadora

Prof. Dr. Carlos Becker Westphall

Bel. Rafael Brundo Uriarte

Profa. Dra. Carla Merkle Westphall

*Uma grama de exemplos vale mais
que uma tonelada de conselhos.*

Provérbio Popular

Ofereço este trabalho a uma pessoa em especial, que me motiva e me ensina todos os dias a ser uma pessoa melhor. Em quem me espelho e tenho maior orgulho e admiração. Meu melhor amigo. Meu pai, Dr. Walter Vitti Junior.

Agradecimentos

Muitas foram as pessoas que de alguma forma especial contribuíram para que esse trabalho fosse realizado. A todos vocês, devo um pouco dessa conquista. Gostaria de deixar registrado alguns agradecimentos especiais.

A toda a minha família, em especial ao meu pai Walter e minha mãe Helaine e aos meus irmãos Vinicius, Tamarah e Isadora por me darem todas as oportunidades de estudo e estarem sempre ao meu lado me ajudando e apoiando em todas as decisões da minha vida.

A Monique Vicente, minha eterna companheira e amiga, por existir e estar ao meu lado, sempre disposta a ouvir, me incentivando com palavras de conforto e carinho.

Gostaria de agradecer a todos do Laboratório de Redes e Gerência. Em especial, ao professor e meu orientador Carlos Westphall, pela oportunidade, por me disponibilizar toda a infraestrutura necessária para o desenvolvimento de minhas atividades e por estar sempre pronto a ajudar. Ao Rafael Uriarte, por toda orientação, e por grande ajuda na definição e no desenvolvimento desse trabalho.

Sumário

Lista de Figuras	8
Lista de Tabelas	10
Lista de Siglas	11
Resumo	12
Abstract	13
1 Introdução	1
1.1 Motivação	2
1.2 Objetivo Geral	2
1.3 Objetivos Específicos	3
1.4 Organização do Trabalho	3
2 Computação em Nuvem	5
2.1 Classificações para Computação em Nuvem	6
2.1.1 Classificação Quanto ao Modelo de Serviço	6
2.1.2 Classificação quanto ao Modelo de Implantação	10
2.2 Padronizações para Computação em Nuvem	11
2.2.1 <i>Open Cloud Computing Interface (OCCI)</i>	11
2.2.2 <i>Open Cloud Consortium (OCC)</i>	12
2.3 Infraestrutura para Computação em Nuvem	13
2.3.1 OpenNebula	13
2.3.2 Eucalyptus	16
2.3.3 Conclusão	19

	7
3 Monitoramento em Computação em Nuvem	20
3.1 Introdução	20
3.2 Nagios	22
3.3 PCMONS	24
3.4 Conclusão	26
4 Desenvolvimento	27
4.1 Introdução	27
4.2 Implementação	29
4.3 Estudo de Caso	32
4.3.1 Ambiente	32
4.3.2 Criação da nuvem privada	33
4.4 Conclusão	42
5 Conclusão	45
5.1 Trabalhos Futuros	46
Referências	48
A Template CentOS 5.7.	50
B Template Ubuntu 10.04.03 LTS.	51
C Script para contextualização da imagem Ubuntu 10.04.03	52
D Script para contextualização da imagem CentOS 5.7	56
E Script para criação da base de dados e tabelas.	60
F Script para execução do PCMONS na inicialização do sistema.	62
G Script para execução do PCMONS na inicialização do sistema.	66

Lista de Figuras

1.1	Abstração da Internet representada por uma nuvem.	2
2.1	Modelos de serviço em computação em nuvem. Inter-relação entre provedores e consumidores. Traduzido de [BRI 09].	6
2.2	Modelos de implantação em computação em nuvem. Traduzido de [Mic 10].	7
2.3	Principais versões do OpenNebula desde 2008 [Ope 11].	15
2.4	Componentes da arquitetura baseada em <i>drivers</i> do OpenNebula. Traduzido de [SOT 09].	17
2.5	Arquitetura principal do <i>Eucalyptus</i> [Euc 11].	19
3.1	Arquitetura de três camadas para monitoramento de nuvens privadas [CHA 10].	21
3.2	Arquitetura da ferramenta PCMONS e interação entre seus componentes. Adaptado de [DC 11].	25
4.1	Modificações feitas no arquivo <code>cluster_config.py</code> do módulo Cluster Integrador de Dados.	29
4.2	Método responsável pelo mapeamento dos nós em cada um dos clusters da nuvem.	30
4.3	Método responsável pela atualização de informações das máquinas virtuais.	31
4.4	Método responsável pela coleta de informações sobre máquinas virtuais no nó.	31
4.5	Arquivo de configuração do OpenNebula EC2 Query.	36
4.6	Arquivo de configuração do OpenNebula Sunstone.	36
4.7	Tela inicial da interface gráfica OpenNebula Sunstone-Server.	38
4.8	Tela de gerenciamento de imagens através do OpenNebula Sunstone-Server.	39
4.9	Script executado na inicialização das máquinas virtuais.	40
4.10	PCMONS: Listagem das máquinas virtuais monitoradas pelo sistema.	41
4.11	Métricas monitoradas no estudo de caso.	42

4.12 PCMONS: Serviços sendo monitorados no momento.	43
4.13 PCMONS: Status dos serviços monitorados em cada uma das máquinas virtuais.	43
4.14 PCMONS: <i>Hostgroup</i> próprio e mapeamento entre máquinas virtuais e suas respectivas máquinas físicas.	44

Lista de Tabelas

4.1	Descrição do ambiente utilizado para o estudo de caso.	32
-----	--	----

Lista de Siglas

PCMONS	<i>Private Cloud Monitoring System</i> ou Sistema de Monitoramento para Nuvens Privadas
IaaS	<i>Infrastructure as a Service</i> ou Infraestrutura como Serviço
PaaS	<i>Platform as a Service</i> ou Plataforma como Serviço
SaaS	<i>Software as a Service</i> ou Software como Serviço
EaaS	<i>Everything as a Service</i> ou Tudo como Serviço
HaaS	<i>Hardware as a Service</i> ou Hardware como Serviço
NIST	<i>National Institute Standards and Technology</i> ou Instituto Nacional de Tecnologia e Padrões
IT	<i>Information Technology</i> ou Tecnologia da Informação
API	<i>Application Programming Interface</i> ou Interface de Programação de Aplicativos
EC2	<i>Elastic Compute Cloud</i> ou Computação em Nuvem Elástica
OCCI	<i>Open Cloud Computing Interface</i> ou Interface Aberta para Computação em Nuvem
OGF	<i>Open Grid Forum</i>
KVM	<i>Kernel-based Virtual Machine</i> ou Máquina Virtual baseada em Kernel
GNU	<i>GNU Not Unix</i>
IP	<i>Internet Protocol</i> ou Protocolo de Internet
XML	<i>eXtensive Markup Language</i>
RPC	Remote Procedure Call ou Chamada Remota de Procedimento
CLI	<i>Command Line Interface</i> ou Interface de Linha de Comando
Eucalyptus	Elastic Utility Computing Architecture Linkin You Programs To Useful Systems
S3	<i>Simple Storage Service</i> ou Serviço de Armazenamento Simples
AWS	<i>Amazon Web Services</i> ou Serviços Web Amazon
SLA	<i>Service Level Agreement</i> ou Acordo de Nível de Serviço
SOAP	<i>Simple Object Access Protocol</i> ou Protocolo Simples de Acesso a Objetos

Resumo

Considerando a falta de soluções genéricas e de código aberto para o gerenciamento e monitoramento de nuvens privadas foi desenvolvido o PCMONS (Private Cloud Monitoring System ou Sistema de Monitoramento de Nuvem Privada). Destinado a ser um sistema de monitoramento extensível e modular para nuvens privadas, o PCMONS atua principalmente na recuperação, coleta e preparação de informação relevante para a visualização dos dados de monitoramento e é especialmente focado em máquinas virtuais.

Esse projeto visa ampliar a compatibilidade desta ferramenta, através do desenvolvimento de uma extensão para integração a umas das principais ferramentas para infraestrutura em computação em nuvem atualmente: o OpenNebula.

Ao longo do trabalho, será apresentada uma revisão sobre o conceito de computação em nuvem, analisando suas principais categorizações, esforços de padronização, ferramentas e questões a respeito do gerenciamento e do monitoramento de nuvens privadas.

Para a validação da implementação e apresentação dos resultados obtidos, foi implantado no Laboratório de Redes e Gerência (LRG) na Universidade Federal de Santa Catarina um ambiente de computação em nuvem privada, utilizando a ferramenta OpenNebula, com monitoramento das máquinas virtuais gerenciadas, empregando a ferramenta PCMONS.

Palavras-chave: computação em nuvem, monitoramento, pcmons, opennebula.

Abstract

Considering the lack of generic and open-source solutions for management and monitoring of private clouds, PCMONS (Private Cloud Monitoring System) was developed. Intended to be an extensible and modular monitoring system for private clouds, PCMONS primarily operates in retrieving, gathering and preparing relevant information for monitoring data visualization and is specially focused on virtual machines.

This project aims to increase the compatibility of this tool by developing an extension to one of the currently leading IaaS tool for cloud computing: OpenNebula.

Throughout the work, will be presented a review of the concept of cloud computing, analyzing their main categorizations, standardization efforts, tools and issues concerning the management and monitoring of private clouds.

To validate the implementation and presents results, it was implemented at the Network and Management Laboratory (LRG) at the Federal University of Santa Catarina a private cloud computing environment using OpenNebula tool. The monitoring of the virtual machines was done using PCMONS.

Keywords: cloud computing, monitoring, pcmons, opennebula.

Capítulo 1

Introdução

Com o avanço da sociedade humana, serviços básicos e essenciais de utilidade pública como eletricidade, água e telefone tornaram-se fundamentais para a vida diária. As infraestruturas existentes atualmente permitem a entrega desses serviços de forma transparente ao usuário por meio de um modelo de pagamento baseado no uso [VEC 09]. O uso desses serviços é cobrado de acordo com uma política de tarifação aos usuários. A computação em nuvem utiliza essa mesma ideia, aplicada na computação. O termo "nuvem" é uma metáfora em relação à forma como a Internet é comumente representada nos diagramas de rede. Nesses diagramas, as nuvens representam todas as tecnologias que fazem a Internet funcionar, abstraindo infraestrutura e complexidade [VEL 09]. Um exemplo dessa abstração é demonstrado na Figura 1.1. Basicamente, a computação em nuvem pode ser entendida como recursos de computação altamente escaláveis fornecidos como um serviço externo através da rede. A cobrança por esses serviços é feita de acordo com a sua utilização, ou seja, sob demanda, assim, o usuário que utiliza um serviço na nuvem paga somente por aquilo que utilizar.

Os sistemas baseados em computação em nuvem se tornam mais comuns a cada dia. Muitas empresas estão migrando totalmente ou parcialmente seus recursos de computação e ou seus serviços e aplicações para ambientes na nuvem. Diversos estudos comparativos estão sendo feitos em relação às vantagens e desvantagens que estas soluções podem trazer.

Muitas das tecnologias envolvidas e que possibilitam a viabilização do paradigma da computação em nuvem já são bastante consolidadas, como a computação em grade (*grid computing*), computação distribuída, acordos de nível de serviço e virtualização. Porém, outras ainda precisam ser arquitetadas ou estendidas de modo a prover seu melhor uso e gerenciamento, como é o caso dos sistemas de monitoramento, instrumentos fundamentais para se



Figura 1.1: Abstração da Internet representada por uma nuvem.

acompanhar o comportamento de um sistema, detectar problemas e prover os dados necessários para atividades de manutenção, suporte e planejamento [CHA 10].

1.1 Motivação

A necessidade de sistemas de monitoramento em ambiente de computação em nuvem é cada vez maior. Com monitoramento é possível assistir diferentes atividades no sistema, e assim se utilizar de registros e outras informações para acompanhar sua infraestrutura, avaliando seu desempenho, comparando sua eficiência e com isso, encontrar maneiras de aperfeiçoar um sistema específico ou um determinado serviço.

1.2 Objetivo Geral

Esse trabalho tem como objetivo ampliar a compatibilidade da ferramenta PC-MONS proposta e desenvolvida por [CHA 10] com o desenvolvimento de uma extensão para integrá-lo à ferramenta de Infraestrutura como Serviço (IaaS) OpenNebula.

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) da Universidade Federal de Santa Catarina;
- Desenvolver e integrar um módulo para o sistema de monitoramento de nuvens privadas PCMONS;
- Testar o sistema desenvolvido através de um estudo de caso.

1.4 Organização do Trabalho

O trabalho está organizado em cinco capítulos.

Capítulo 1 – Introdução - Apresenta introdução e contextualização ao tema. São apresentados também: a motivação, objetivo geral e os objetivos específicos do trabalho.

Capítulo 2 – Computação em Nuvem - Apresenta algumas formas de categorizações de computação em nuvem, como em relação ao seu modelo de serviço, e modelo de implantação e são apresentadas algumas tentativas de padronizações utilizadas. São mostradas duas ferramentas para implantação de computação em nuvem na modalidade IaaS: Eucalyptus e OpenNebula.

Capítulo 3 – Monitoramento em computação em Nuvem - Esse capítulo apresenta características de monitoramento para computação em nuvem. Necessidade e importância do monitoramento. Apresenta também a ferramenta Nagios, utilizada para a visualização dos dados de monitoramento e a ferramenta PCMONS, empregada para o monitoramento de nuvens privadas. Ambas foram utilizadas no presente trabalho.

Capítulo 4 – Desenvolvimento - Nesse capítulo são apresentados os passos para a implantação de uma nuvem privada, utilizando a ferramenta OpenNebula no Laboratório de Redes e Gerência, para fins de estudo e conhecimento prático. São demonstradas as etapas utilizadas para o desenvolvimento de uma adaptação da ferramenta PCMONS para torná-la

compatível com o OpenNebula. A implementação é testada, utilizando esse mesmo ambiente em nuvem.

Capítulo 5 – Conclusão e Trabalhos Futuros - Este capítulo encerra o trabalho com algumas conclusões e considerações finais. São apresentadas algumas perspectivas para trabalhos futuros.

Capítulo 2

Computação em Nuvem

O paradigma de computação em nuvem, por estar envolvido com diversas outras tecnologias computacionais, muitas vezes tem a sua definição incorporada pelas definições já consolidadas dessas tecnologias, como a computação em grade, a computação distribuída, a virtualização, entre outras. Assim, uma definição padrão aceita universalmente para a computação em nuvem se torna difícil. O [NIS 11], National Institute Standards and Technology, do governo dos Estados Unidos define computação em nuvem como um modelo que possibilita acesso conveniente e sob demanda, através da rede, a um conjunto compartilhado de recursos computacionais configuráveis (rede, servidores, armazenamento, aplicações e serviços). Esses recursos podem ser providos rapidamente e liberados com um mínimo de esforço de gerenciamento ou interação com o provedor do serviço.

Uma definição mais simples do termo seria o provimento de recursos computacionais para um cliente a partir de uma demanda. O fornecedor desses recursos abstrai as tecnologias envolvidas e a procedência do recurso para os usuários finais. Desse modo, um cliente, ao contratar um serviço, se atém apenas à utilização do recurso em si, abstraindo-se da tecnologia envolvida para o recebimento desses recursos e até mesmo a origem ou localização geográfica do recurso. Com isso, a obtenção de recursos, como servidores para armazenamento de arquivos ou servidores web acaba tornando-se desnecessária, evitando na maioria das vezes, uma subutilização de recursos computacionais. Uma empresa, ao comprar a quantidade certa de recursos de TI (Tecnologia da Informação) sob demanda, pode evitar a compra de equipamentos desnecessários. Bancos de dados, redes, aplicativos, plataformas e até infraestruturas completas são alguns exemplos de recursos computacionais que podem ser fornecidos.

Grandes empresas, como Amazon, Google, Microsoft, Apple, IBM, investem em

pesquisas e desenvolvimento na área na tentativa de ganhar espaço e se tornar referência no mercado de computação em nuvem.

2.1 Classificações para Computação em Nuvem

Os ambientes de computação em nuvem podem ser classificados seguindo mais de um critério. A seguir são listadas duas das possíveis classificações para esses ambientes em relação ao seu modelo de serviços e ao seu modelo de implantação. As Figuras 2.1 e 2.2 mostram exemplos de modelos de serviços e a inter-relação entre os fornecedores desses serviços e seus consumidores e exemplos de modelos de implantação, respectivamente.

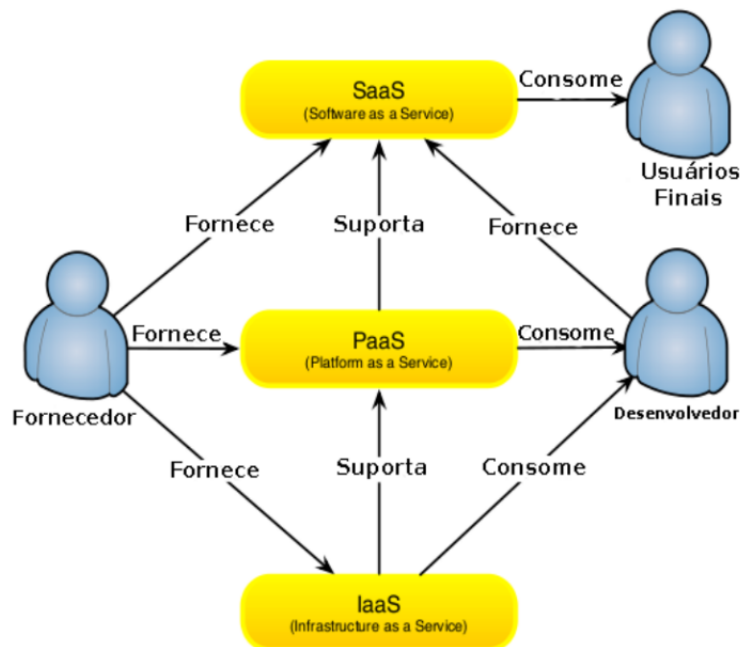


Figura 2.1: Modelos de serviço em computação em nuvem. Inter-relação entre fornecedores e consumidores. Traduzido de [BRI 09].

2.1.1 Classificação Quanto ao Modelo de Serviço

2.1.1.1 Software como Serviço (SaaS)

É um modelo de distribuição de *software* em que os aplicativos são hospedados por um provedor ou um fornecedor de serviço e disponibilizados aos clientes, através de uma rede, geralmente a Internet. Algumas das vantagens da utilização do modelo SaaS são:

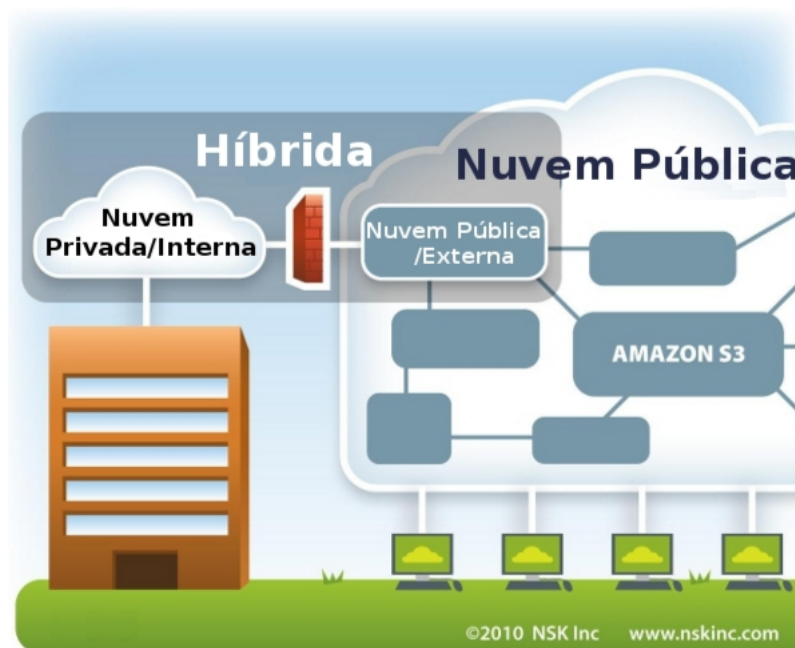


Figura 2.2: Modelos de implantação em computação em nuvem. Traduzido de [Mic 10].

- ✓ Facilidade de administração;
- ✓ Compatibilidade: todos os clientes utilizarão a mesma versão do aplicativo;
- ✓ Atualizações automáticas, sem necessidade de envolvimento do usuário;
- ✓ Facilidade para criação de ambientes colaborativos;
- ✓ Acessibilidade global.

Pode-se citar como exemplos: o Google (<http://www.google.com>), com a sua suíte de aplicativos Google Apps (<http://www.google.com/apps>), incluindo o servidor de e-mails Gmail (<http://mail.google.com>), e seu conjunto de aplicativos para escritório Google Docs (<http://docs.google.com>), entre outros serviços.

2.1.1.2 Plataforma como Serviço (PaaS)

Nesse modelo, um provedor oferece além de uma infraestrutura, um conjunto de soluções e ferramentas que um desenvolvedor necessita para criar uma aplicação. Esse modelo oferece a capacidade de gerenciamento de todas as fases do desenvolvimento de uma aplicação, desde a modelagem e o planejamento, até a construção, implantação para testes e manutenção. Esse modelo é uma consequência direta do modelo de Software como Serviço, descrito na seção 2.1.1.1. Pode-se citar, como vantagens da utilização desse modelo:

- ✓ Não há necessidade de comprar todo o sistema, aplicativos, plataformas, e ferramentas necessárias para construir, executar e implantar o aplicativo;
- ✓ Recursos do sistema operacional podem ser alterados e atualizados com frequência;
- ✓ Equipes de desenvolvimento, geograficamente distribuídas, podem trabalhar juntas em projetos de desenvolvimento;
- ✓ Despesas gerais minimizadas pela unificação dos esforços no desenvolvimento.

Uma das desvantagens de utilização desse modelo é a confiança na segurança dos dados que o desenvolvedor ou a empresa devem ter na provedora do serviço. A ideia de que dados pessoais ou corporativos privados, críticos ou não, não estão armazenados em domínios próprios pode não ser tolerado por algumas pessoas e organizações.

Pode-se citar como exemplos de provedores de PaaS: o Google App Engine (<http://appengine.google.com>) que permite o desenvolvimento de aplicações nas linguagens de programação Python e Java, o Heroku (<http://www.heroku.com>), da empresa Salesforce.com que suporta múltiplas linguagens como Ruby, Node.js, Clojure, Java, Python e Scala e o Cloud Foundry (<http://www.cloudfoundry.com>), ferramenta de código aberto da empresa VMware que fornece uma plataforma para desenvolvimento utilizando diversas linguagens.

2.1.1.3 Infraestrutura como Serviço (IaaS)

Nessa modalidade é oferecida ao cliente uma infraestrutura de hardware completa. Armazenamento, hardware (memória, processamento, etc.), servidores e componentes de rede são oferecidos aos usuários. O consumidor é capaz de implantar e executar aplicativos arbitrários, que podem incluir sistemas operacionais completos e tecnologias de virtualização para o gerenciamento dos recursos [HUR 10]. O cliente não administra ou controla a infraestrutura da nuvem, porém tem controle sobre os sistemas operacionais, armazenamento, uso de aplicativos e controle dos componentes de rede.

Esse modelo, referido por alguns autores como *Hardware as a Service (HaaS)* possui inúmeras vantagens, dentre elas:

- ✓ Permite que uma empresa mantenha o foco nos produtos e serviços que oferece, delegando o gerenciamento de tecnologia da organização à provedora de infraestrutura;
- ✓ A infraestrutura pode ser rapidamente ampliada em caso de maior demanda e reduzida caso a demanda recue;

- ✓ Redução de custos com utilização ótima de recursos. Uso dos serviços com base na exigência e somente enquanto forem necessários;
- ✓ Flexibilidade. Permite o acesso a infraestrutura a partir de qualquer lugar, utilizando diversos dispositivos;
- ✓ Permite economia de energia, diminuindo o impacto ambiental, contribuindo para iniciativas da chamada TI Verde ou *Green IT*.

O pioneiro e melhor exemplo de IaaS atualmente é o Amazon EC2 (Elastic Compute Cloud), oferecido pela empresa Amazon. São oferecidas ao cliente infraestruturas completas virtualizadas e todo o controle e gerenciamento pode ser feito remotamente, utilizando uma API de serviços web. Outros exemplos são: o SmartCloud (<http://www.ibm.com/SmartCloud>), da empresa IBM, o RightScale (<http://www.rightscale.com>) e GoDaddy (<http://www.godaddy.com/>).

2.1.1.4 Tudo como Serviço (AaaS)

Conforme aumentam os serviços prestados por provedores utilizando o paradigma da computação em nuvem, diversos novos termos são criados. A sigla AaaS ou XaaS, refere-se à frase em inglês *Anything as a Service*, ou Tudo como Serviço, onde a letra X pode ser substituída por diversas letras e assumir vários significados diferentes. Esses modelos, apesar de utilizarem outras nomenclaturas, acabam se derivando de modelos já consolidados e citados anteriormente. Alguns exemplos:

- Banco de Dados como Serviço: capacidade de utilizar os serviços de um banco de dados hospedado remotamente;
- Segurança como Serviço: fornecimento de serviços de segurança essenciais remotamente via Internet, como por exemplo, gerenciamento de identidades;
- Teste como serviço: provimento de serviços de teste hospedados remotamente para testar sistemas locais;
- Informação como Serviço: capacidade de utilizar de qualquer tipo de informação remota por meio de uma interface bem definida, como uma API;
- Processo como Serviço, Gestão como Serviço, etc.;

2.1.2 Classificação quanto ao Modelo de Implantação

Ambientes de computação em nuvem, além da categorização por modelos de serviço, podem ser classificados em relação ao seu modelo de implantação, ou seja, sua abrangência de público. Abaixo são citados os quatro modelos mais citados na literatura.

2.1.2.1 Nuvem Pública

A nuvem pública, ou nuvem externa, como alguns autores citam, descreve o significado convencional da computação em nuvem, onde um prestador de serviços disponibiliza recursos computacionais, tais como aplicativos e armazenamento para o público em geral, através da Internet. Serviços de nuvem pública podem ser livres ou oferecidos utilizando um modelo de pagamento baseado no uso (*pay-per-use* - modalidade onde o contratante solicita os serviços e recursos de acordo com sua necessidade e disponibilidade e paga apenas pelo que for utilizado) [SMO 12]. Empresas como Salesforce.com, Amazon EC2 e Flexiscale oferecem esse tipo de serviço.

2.1.2.2 Nuvem Privada

Em uma nuvem privada, uma infraestrutura é disponibilizada para uso exclusivo de uma única organização que compreende vários consumidores (por exemplo, unidades de negócios). Pode ser de propriedade, gerenciados e operados pela organização, um terceiro, ou uma combinação deles, e podem existir dentro ou fora das instalações da empresa [NIS 11]. Ferramentas como Eucalyptus e OpenNebula permitem a implantação desse tipo de modelo.

2.1.2.3 Nuvem Híbrida

Uma nuvem híbrida acontece quando recursos computacionais de nuvens públicas e de nuvens privadas são utilizados. Uma empresa pode optar por usar um serviço de nuvem pública para a utilização de recursos em geral, porém, pode armazenar suas informações críticas de negócios em uma nuvem privada, dentro do seu próprio domínio com a finalidade de aumentar a segurança dos dados. Para [SOT 09] uma nuvem privada, no entanto, pode dar suporte à uma nuvem híbrida, através da complementação da capacidade da infraestrutura local com a capacidade computacional de uma nuvem pública.

2.1.2.4 Nuvem Comunitária

Para [NIS 11], nesse modelo a infraestrutura da nuvem é compartilhada por diversas organizações, dando suporte a uma comunidade específica, com preocupações ou atividades em comum, podendo ser gerenciada pela própria organização ou por terceiros e se localizar dentro ou fora dos limites da organização.

2.2 Padronizações para Computação em Nuvem

A indústria da computação em nuvem está nos estágios iniciais de implantação de padrões, desde o armazenamento de rede à segurança. Hoje não há uma maneira padrão para as empresas formatarem seus dados para que possam ser facilmente movidos entre uma variedade de provedores de computação em nuvem. Se um cliente têm os seus dados em uma nuvem AWS S3, da Amazon, por exemplo, não pode necessariamente pegar esses dados e colocá-los em uma nuvem de outra provedora, como Rackspace, usando as mesmas chamadas da API. Existe uma falta de padrões na computação em nuvem. Sem padrões, a indústria acaba criando diversos sistemas proprietários fechados e assim uma interoperabilidade entre fornecedores de serviços se torna difícil. Como os clientes não querem ficar atrelados a um único sistema ou fornecedor existe uma forte pressão da indústria para a criação de padrões.

Uma série de organizações e grupos informais abordam esse problema. O site Cloud Standards (<http://www.cloud-standards.org>) reúne através de projeto colaborativo iniciativas de diversas organizações para a padronização em computação em nuvem. Algumas dessas iniciativas são descritas na próxima seção.

2.2.1 *Open Cloud Computing Interface (OCCI)*

O *Open Cloud Computing Interface*, ou em português, Interface para Computação em Nuvem Aberta é um conjunto de especificações de propósito geral, baseadas em computação em nuvem, para interações com recursos de uma forma que é explicitamente independente de fornecedor, e pode ser estendido para resolver uma ampla variedade de problemas em computação em nuvem.

O conjunto de especificações OCCI é um produto do *Open Grid Forum (OGF)*, uma organização de desenvolvimento de padrões abertos na área de redes distribuídas, computação e armazenamento, com ênfase em tecnologias de grande escala em computação distribuída. O

OGF desenvolve seus padrões através de um processo aberto que reúne entradas e contribuições da comunidade e realiza um refinamento através de revisões e comentários do público, a fim de produzir normas, orientações e informações de valor para a comunidade.

O OCCI fornece um protocolo e uma API para todos os tipos de tarefas de gerenciamento de nuvem. O trabalho foi iniciado originalmente para criar uma API de gerenciamento remoto para serviços baseados em modelo IaaS, permitindo o desenvolvimento de ferramentas interoperáveis para tarefas comuns, incluindo dimensionamento, implantação e monitoramento. Desde então, evoluiu para uma API flexível, com foco na integração, portabilidade, interoperabilidade e inovação, oferecendo ainda um alto grau de extensibilidade. A versão atual da interface é adequada para servir muitos outros modelos, além de IaaS, incluindo, por exemplo, PaaS e SaaS.

2.2.2 *Open Cloud Consortium (OCC)*

O Open Cloud Consortium, de acordo com o seu site principal, foi formado em 2008, e seus principais objetivos são:

- Suporte ao desenvolvimento de padrões para computação em nuvem e de *arcabouços* para interoperabilidade entre nuvens;
- Desenvolvimento de *benchmarks* para computação em nuvem;
- Suporte às implementações de referências para computação em nuvem, de preferência implementações de código aberto;
- Gerenciamento de infraestrutura para computação em nuvem para suporte a pesquisas científicas, como a *Open Science Data Cloud*.

Essas tarefas estão divididas em grupos de trabalho que apoiam o interesse e as atividades dos membros da OCC. Os grupos de trabalho atuais incluem:

The Open Science Data Cloud (OSDC) - Grupo de trabalho que administra e opera uma nuvem com grande quantidade de informações para dados científicos. Entre os membros deste grupo de trabalho incluem o Yahoo, que contribuiu com equipamentos para a prova de conceito do OSDC e a Cisco, que fornece equipamento para conectar os vários centros OSDC distribuídos geograficamente.

Projeto Matsu - Este grupo de trabalho está desenvolvendo uma nuvem que pode ajudar em momentos de desastres naturais, proporcionando uma capacidade elástica para processar dados geoespaciais. Armazenamento baseado em nuvem e serviços de computação estão disponíveis para o projeto e podem ser usados, por exemplo, para auxiliar o processamento de imagens de modo que essas imagens possam ser disponibilizadas para aqueles que fornecem assistência a desastres.

OCC Virtual Network Testbed - É uma vasta área distribuída de testes para redes virtuais. O foco inicial é comparar e contrastar várias tecnologias para criação e gerenciamento de redes virtuais.

The Open Cloud Testbed - Este grupo utiliza equipamentos e várias redes internacionais de pesquisa nos EUA para testar as diferentes tecnologias para nuvens de amplo alcance. A participação nesse grupo de trabalho é limitado aos membros OCC que contribuem com recursos computacionais, como rede e poder de processamento.

2.3 Infraestrutura para Computação em Nuvem

Atualmente, existem diversas soluções para implantação de um ambiente de computação em nuvem na modalidade IaaS. Nas próximas seções são apresentadas as ferramentas OpenNebula e Eucalyptus, por serem ferramentas de código aberto amplamente utilizadas atualmente.

2.3.1 OpenNebula

2.3.1.1 Introdução

O OpenNebula é um conjunto de ferramentas de código aberto para implantação de computação em nuvem na modalidade IaaS, oferecendo diversos recursos e soluções que facilitam e flexibilizam o gerenciamento completo de infraestruturas virtualizadas. Pode ser utilizado para o desenvolvimento de nuvens privadas, públicas, comunitárias e híbridas, possuindo a capacidade de combinar uma infraestrutura local com uma infraestrutura baseada em nuvem pública, permitindo ambientes altamente escaláveis. Oferece uma interface compatível com as interfaces EC2 Query, OGF OCCI e vCloud, largamente utilizadas e consideradas como padrões quando se trata de serviços para a nuvem. Inclui recursos para integração, gerenciamento, escalabilidade, segurança e contabilidade. Enfatiza a interoperabilidade, padronização

e portabilidade, fornecendo aos usuários e administradores da nuvem a possibilidade de escolha entre diversas interfaces e hipervisores (Xen, KVM e VMware ESX). Possui também uma arquitetura flexível que pode acomodar hardwares múltiplos e combinações diferentes de softwares. A ferramenta pode ser instalada em qualquer distribuição GNU Linux a partir do seu código fonte ou através dos repositórios oficiais de algumas distribuições específicas (Debian, openSUSE, Ubuntu). Muitos de seus recursos foram desenvolvidos para atender aos requisitos de casos de uso de negócio de empresas líderes de diversos setores em computação em nuvem, tais como Reservoir, StratusLab, BonFIRE, or 4CaaS. É considerada referência para a computação em nuvem em pesquisas de vários grandes projetos de infraestrutura. No momento da realização deste trabalho, o OpenNebula encontra-se na versão 3.2.1.

2.3.1.2 História

De acordo com [Ope 11] o OpenNebula foi criado, inicialmente, como um projeto de pesquisa, em 2005, por Ignacio M. Llorente e Rubén S. Montero. Desde sua primeira versão pública do software, de março de 2008, evoluiu através de versões de código aberto e agora opera como um projeto *open-source*. O OpenNebula é o resultado de muitos anos de pesquisa e desenvolvimento em gestão eficiente e escalável de máquinas virtuais em infraestruturas distribuídas em estreita colaboração com a comunidade de usuários.

A tecnologia do OpenNebula amadureceu graças a uma comunidade ativa de usuários e desenvolvedores. Diferentes projetos, grupos de pesquisa e empresas construíram novos componentes para complementar e melhorar suas funcionalidades. Em março de 2010, os principais autores de OpenNebula fundaram a C12G Labs para permitir que o projeto OpenNebula não fosse vinculado exclusivamente ao financiamento público [Maa 11]. A Figura 2.3 mostra os principais lançamentos da ferramenta desde de sua primeira versão pública no ano de 2008.

2.3.1.3 Arquitetura

A arquitetura interna do OpenNebula engloba vários componentes especializados em diferentes aspectos para o gerenciamento da infraestrutura virtual e pode ser dividida em três camadas.

Core (núcleo) - Responsável por controlar o ciclo de vida das máquinas virtuais, o núcleo do OpenNebula possui três diferentes áreas de gerenciamento, são elas:

OpenNebula 3.4 Released	2012/04/10
OpenNebula 3.4 Beta Release	2012/03/29
OpenNebula 3.2 Released	2012/01/17
OpenNebula 3.2 Beta Release	2011/12/16
OpenNebula 3.0 Released	2011/10/03
OpenNebula 3.0 Beta 2 Release	2011/09/08
OpenNebula 3.0 Beta 1 Release	2011/07/20
OpenNebula 2.2 Released	2011/03/28
OpenNebula 2.2 Beta1 Release	2011/03/02
OpenNebula 2.0 Released	2010/10/24
OpenNebula 2.0 Beta 1 Release	2010/07/28
OpenNebula Cloud Toolkit Goes Commercial	2010/05/05
New Web Site for OpenNebula.org	2010/02/24
OpenNebula 1.4 Released	2009/12/16
OpenNebula Cloud Announcement	2009/11/18
OpenNebula 1.4 RC Released!	2009/11/18
OpenNebula 1.4 Beta 2 Released!	2009/10/30
New Research Grants to Fund OpenNebula until 2012	2009/09/04
OpenNebula Tarantula Stable version 1.2.1	2009/07/29
OpenNebula 1.4 Beta 1 Codename Hourglass out for Testing	2009/07/23
Ubuntu 9.04 (Jaunty Jackalope) has been released today bringing OpenNebula	2009/04/23
OpenNebula Wins the Best Demo Award at OGF25/4th EGEE-UF	2009/03/09
OpenNebula 1.2 release	2009/02/06
OpenNebula 1.2 beta release	2008/12/05
Release of OpenNebula 1.0 for Data Center Virtualization & Cloud Solutions	2008/07/24
New Technology Preview (TP2) of the OpenNebula (ONE) Virtual Infrastructure Engine	2008/07/24
Technology Preview of the OpenNebula (ONE) Virtual Infrastructure Engine	2008/07/24

Figura 2.3: Principais versões do OpenNebula desde 2008 [Ope 11].

- Tecnologias de imagens e armazenamento, para a preparação de imagens de disco para as máquinas virtuais;
- Tecnologias de rede, para o provimento de ambientes de rede para as máquinas virtuais;
- Hipervisores, para a criação e gerenciamento das máquinas virtuais.

O núcleo realiza as operações específicas para armazenamento, rede e virtualização, por meio de *drivers* conectáveis. Assim, o OpenNebula não está vinculado a qualquer ambiente específico, fornecendo uma camada de gerenciamento uniforme, independentemente da infraestrutura utilizada. Além de gerenciar as máquinas virtuais como uma unidade, o núcleo também lida com a configuração e entrega de informações de contexto (como endereço IP, *hostname*, certificados digitais, licenças de software, etc.) para cada uma delas [COR 10].

Escalonador e outras ferramentas - Essa camada contém diversas ferramentas fornecidas com o OpenNebula, como uma interface de linha de comando, ou CLI (Command Line Interface), o escalonador, e também ferramentas de terceiros que podem ser facilmente criadas

utilizando as interfaces XML-RPC ou a API do OpenNebula. Um componente escalonador separado toma as decisões de alocação das máquinas virtuais. Mais especificamente, o escalonador tem acesso às informações sobre todas as requisições que o OpenNebula recebe e, com base nessas requisições, acompanha as alocações atuais e futuras, criando e atualizando o escalonador de recursos e enviando os comandos de implantação adequados ao núcleo do OpenNebula. O escalonador padrão do OpenNebula fornece uma política de escalonamento que coloca as máquinas virtuais em recursos físicos de acordo com um algoritmo de classificação que o administrador pode configurar. Ele se baseia em dados em tempo real de ambas as máquinas virtuais em execução e os recursos físicos disponíveis [SOT 09]. O OpenNebula oferece nessa camada interfaces de gerenciamento para integrar as funcionalidades do núcleo dentro de outras ferramentas de gerenciamento do *datacenter*, como contabilidade e *arcabouços* para monitoramento. Para este fim, OpenNebula implementa a API libvirt (<http://libvirt.org>), uma interface aberta para o gerenciamento de máquinas virtuais, e também uma interface de linha de comando. Estas funcionalidades são expostas aos usuários externos através de uma interface para nuvem.

Drivers - O OpenNebula fornece nessa camada um conjunto de módulos conectáveis para interagir com *middlewares* específicos (por exemplo, hipervisores de virtualização, mecanismos de transferência de arquivos ou serviços de informação). Estes módulos são chamados de Drivers de Acesso Middleware. O suporte para nuvem híbridas é feito utilizando *Drivers* para comunicação com nuvens externas. Isso permite que as organizações complementem sua infraestrutura local, com a capacidade de computação de uma nuvem pública para atender às demandas de pico, atendendo melhor os pedidos de acesso (por exemplo, movendo o serviço para locais mais próximos do usuário), ou implementando estratégias de alta disponibilidade [Maa 11].

O OpenNebula foi projetado para ser modular, a fim de permitir a sua integração com o maior número possível de hipervisores e diferentes ambientes e arquiteturas. [COR 10]. A Figura 2.4 ilustra os componentes da arquitetura geral da ferramenta OpenNebula.

2.3.2 Eucalyptus

O Eucalyptus, do inglês *Elastic Utility Computing Architecture Linkin Your Programs To Useful Systems* ou, numa tradução livre, Arquitetura de Computação Utilitária e Elástica Para Vincular seus Programas a Sistemas Úteis é uma ferramenta baseada no sistema ope-

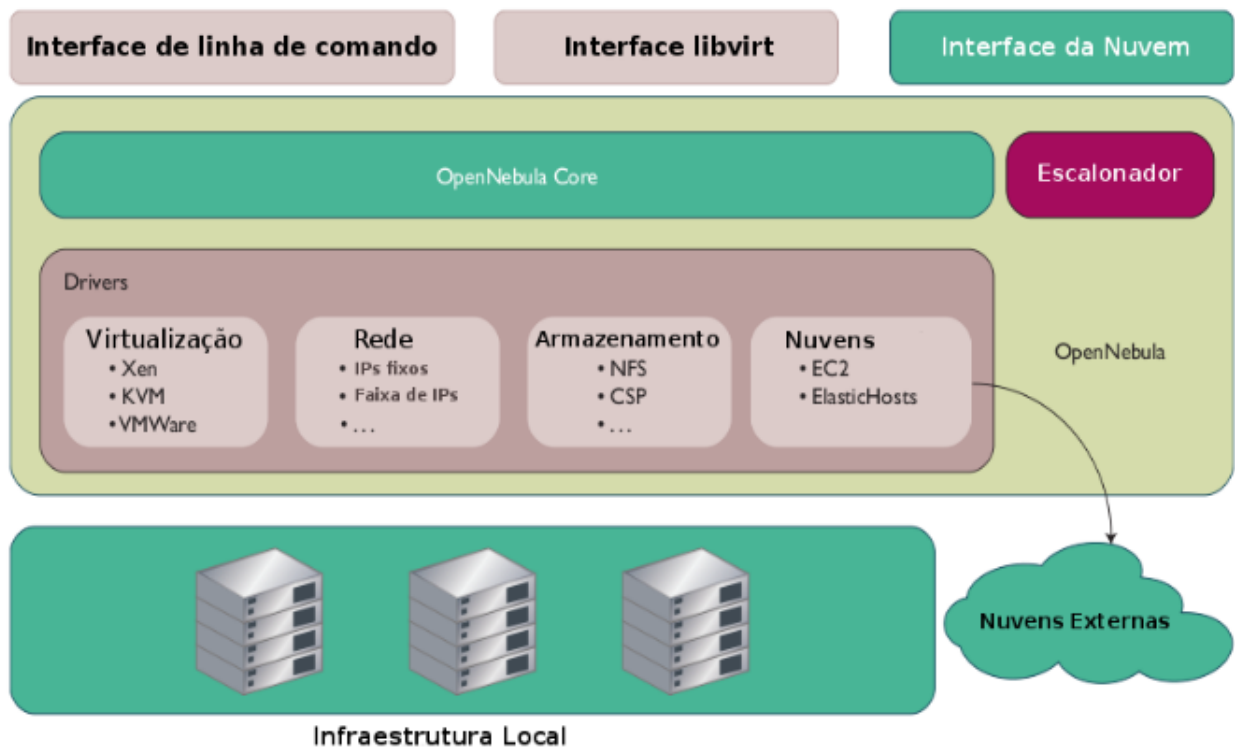


Figura 2.4: Componentes da arquitetura baseada em *drivers* do OpenNebula. Traduzido de [SOT 09].

racional GNU/Linux que implementa nuvens privadas e híbridas de forma escalável e com eficiência dentro de uma infraestrutura de TI de uma organização. Possui uma arquitetura modular que facilita o seu entendimento e extensão para o uso acadêmico.

O Eucalytus começou como um projeto de pesquisa na área de computação de alta performance no departamento de Ciências da Computação na Universidade da Califórnia em Santa Barbara, EUA. Em Janeiro de 2009, uma empresa denominada Eucalyptus Systems, Inc. foi fundada para apoiar a comercialização do Eucalyptus. A ferramenta possui uma versão empresarial e uma versão gratuita e de código aberto que possui algumas limitações em relação à versão empresarial.

A ferramenta implementa o conceito de IaaS e, assim, aborda questões do paradigma de computação em nuvem como instanciamento de máquinas virtuais e definições e execuções de acordos de níveis de serviço. Provê também uma interface única que permite que os usuários acessem recursos computacionais disponíveis tais como máquinas virtuais, rede, armazenamento. Possui uma interface compatível com a interface EC2/S3, da Amazon, interfaces mais aceitas no mercado. No momento do desenvolvimento desde trabalho a versão de código aberto do Eucalyptus encontra-se na versão 2.0.3, enquanto que sua versão empresarial

encontra-se na versão 3.0. Algumas das características da ferramenta são:

- Compatibilidade com a Amazon Web Services API (AWS);
- Instalação a partir do código fonte ou a partir de pacotes binários DEB e RPM;
- Suporte aos hipervisores Xen e KVM;
- Suporte para múltiplos clusters como uma única nuvem;
- Interface Web e linha de comando para administração e configuração da nuvem;
- IPs elásticos e grupos de segurança;
- Gestão de usuários e grupos;
- Políticas de escalonamento configuráveis e SLAs.

O Eucalyptus é dividido em cinco camadas principais de alto nível que funcionam juntas para fornecer os serviços em nuvem solicitados. Cada uma delas possui a sua própria interface de serviço *web* conforme mostrado na Figura 2.5 e se comunicam entre si com segurança usando sistema de mensagens SOAP com WS-Security. São elas:

Controlador de Nuvem (Cloud Controller - CLC) - É o controlador da camada principal responsável por gerenciar todo o sistema. É o ponto de entrada na nuvem para usuários e administradores. Ele indaga os gerenciadores de nodos sobre informações a respeito de recursos, toma decisões de agendamento de alto nível e as implementa através de requisições aos controladores de clusters. É a interface pública da nuvem.

Walrus - Controlador que gerencia o acessos aos serviços de armazenamento no Eucalyptus provendo um mecanismo para acesso e armazenamento de imagens de máquinas virtuais e dados de usuários. Este serviço é similar e compatível com o EBS (Elastic Block Storage) da Amazon.

Controlador de Cluster (Cluster Controller - CC) - Controlador responsável por gerar toda a rede de instância virtual. Geralmente é executado na máquina principal de um cluster. Faz o agendamento e a coleta de informações da execução das máquinas virtuais em um controlador de nodo.

Controlador de Armazenamento (Storage Controller - SC) - Serviço de armazenamento baseado em put/get que implementa a interface S3 (Simple Storage Service) da Amazon, provendo um mecanismo para acesso e armazenamento de imagens de máquinas virtuais e dados de usuários.

Controlador de Nó (Node Controller - NC) - Controla o sistema operacional hospedeiro e o hipervisor correspondente. É executado em todos os nós designados para hospedar máquinas virtuais. Controla a execução, inspeção e término das instâncias de máquinas virtuais.

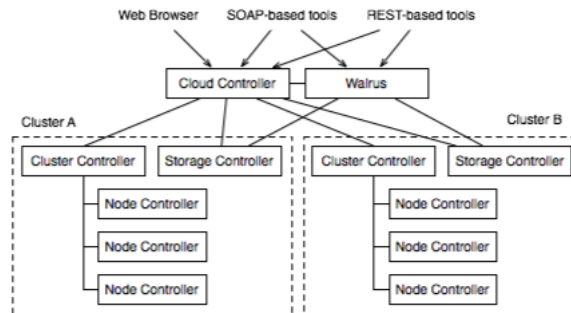


Figura 2.5: Arquitetura principal do *Eucalyptus* [Euc 11].

O Eucalyptus é capaz de gerar arquivos de configuração para aplicativos especializados em monitoramento como o Nagios e o Glanglia. Nenhuma ferramenta para o controle e monitoramento de seus componentes e das máquinas virtuais é disponibilizado.

2.3.3 Conclusão

Neste capítulo foi apresentada uma introdução sobre o paradigma de computação em nuvem. Foram mostrados também as categorizações em relação ao modelo de implantação e ao modelo de serviços e os esforços de padronizações em ambientes de nuvem. Foram introduzidas as ferramentas de código aberto que implementam computação em nuvem na modalidade IaaS, OpenNebula e Eucalyptus.

Capítulo 3

Monitoramento em Computação em Nuvem

3.1 Introdução

Monitoramento é o processo de obter informações sobre elementos de um sistema computacional. Estas informações ajudam a entender a situação do sistema, sua configuração, estatísticas de uso e desempenho, informações sobre erros e sobre a topologia do sistema. O processo de monitoramento depende de técnicas para coletar, processar, armazenar e disponibilizar estas informações. A variedade de elementos que compõem um sistema computacional exige técnicas adequadas para cada classe de elemento [VER 09].

Ambientes de computação em nuvem, especialmente os que atuam no modelo de IaaS, fornecem recursos computacionais a seus clientes utilizando a tecnologia de máquinas virtuais. Estas podem ser definidas como uma implementação de software de um ambiente computacional em que um sistema operacional ou programa pode ser instalado e executado. A máquina virtual emula um ambiente de computação física. Desse modo, tal como um ambiente de computação tradicional que emprega máquinas físicas, necessita de um monitoramento do sistema utilizando métricas tais como: processamento, memória, disco rígido, rede, entre outros, um ambiente virtual utilizando máquinas virtuais também deve ser monitorado. Na maioria das vezes ambientes virtualizados possuem suas próprias ferramentas de monitoramento que são dependentes de plataforma como o XenCenter para o Citrix XenServer e o VMware vCenter para o VMware vSphere [HUA 09]. De acordo com [SHA 09] sistemas genéricos existentes de monitoramento de desempenho de máquinas virtuais são inadequados, com interfaces de ge-

renciamento fracas e sistemas para coleta de informações incompreensíveis. Desse modo, fica difícil para administradores encontrarem pontos de estrangulamento do sistema através dos dados obtidos, e portanto, torna-se complicado o gerenciamento dos recursos de forma a melhorar o desempenho de todo o sistema.

Para companhias e gerentes de TI não importa se a organização mantém uma infraestrutura de TI tradicional ou se possui uma infraestrutura virtual baseada em um ambiente de computação em nuvem. O monitoramento em busca de informações, dados e estatísticas é crucial. É necessária uma estratégia de monitoramento adaptada a recursos de computação em nuvem, aplicações e infraestrutura.

Em [CHA 10] é proposto um modelo de arquitetura genérica para monitoramento de nuvens privadas dividido em três camadas conforme ilustrado na Figura 3.1.

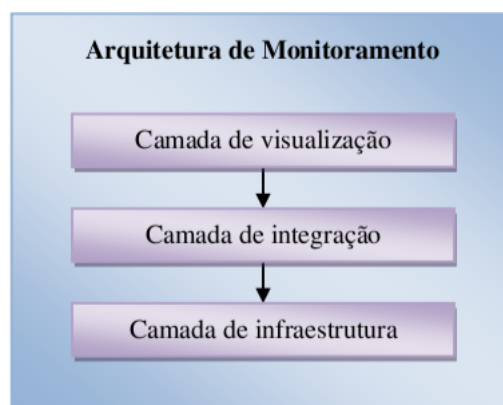


Figura 3.1: Arquitetura de três camadas para monitoramento de nuvens privadas [CHA 10].

Camada de infraestrutura - É a camada mais baixa da arquitetura e a base do modelo. Consiste em toda a infraestrutura utilizada para criar o ambiente de nuvem. Pode-se citar como pertencentes a essa camada toda a infraestrutura de hardware, tais como: dispositivos de armazenamento, processamento, rede, entre outros, e de software necessárias, tais como: sistemas operacionais, aplicativos diversos, licenças, hipervisores, entre outros [CHA 10].

Camada de Integração - A camada de infraestrutura é composta, principalmente, de recursos heterogêneos e, portanto, requer uma interface comum para acesso. Uma situação típica é um usuário que solicita a instanciação de uma máquina virtual. Este tipo de solicitação pode ser controlada por diferentes hipervisores, tais como Xen, KVM, VMWare,

dependendo de como a camada de infraestrutura é implantada. Outro cenário possível é a presença de múltiplas plataformas de computação em nuvem, como Eucalyptus e Open-Nebula. Assim, ações de controle a serem executadas na camada de infraestrutura devem ser sistematizadas antes de serem passadas para o serviço apropriado. Portanto, a camada de integração é responsável por abstrair quaisquer detalhes da infraestrutura [DC 11].

Camada de Visualização - A camada de visualização serve de interface de gerenciamento de alto nível, através da qual é possível verificar informações como o cumprimento de políticas organizacionais e SLAs [DC 11]. Usuários dessa camada estão interessados em averiguar máquinas virtuais disponíveis ou obter informações de monitoramento de máquinas virtuais em execução.

Citando [CHA 10], essa arquitetura em três camadas foi projetada para atender às necessidades de monitoramento de uma nuvem privada. Entre essas necessidades pode-se citar a integração aos sistemas de monitoramento já utilizados na empresa, como o Nagios, que além de ser extensível para atender a novos recursos, é simples de operar. Para validação da arquitetura proposta, a autora propõe o desenvolvimento de uma solução intitulada PCMONS (Private Cloud Monitoring Systems), descrita na seção 3.3.

3.2 Nagios

O Nagios é um sistema de código aberto baseado em Unix e uma ferramenta para monitoramento de redes. Com ele, pode-se monitorar servidores, dispositivos de rede e aplicações e essencialmente qualquer dispositivo ou serviço que tenha um endereço e possa ser contactado via TCP/IP, emitindo alertas aos usuários quando alguma anomalia é detectada [BAR 08].

Alguns dos recursos do Nagios são:

- Monitoramento de serviços de rede (SMTP, POP3, HTTP, NNTP, PING);
- Monitoramento dos recursos das máquinas (carga do processador, uso de disco);
- Monitoramento de *hosts* executando diversos sistemas operacionais como Microsoft Windows, Unix/Linux, Novell NetWare, e outros;
- Sistema simples de *plugins* que permite aos usuários desenvolver suas próprias verificações de serviços;

- Verificação dos serviços paralelamente;
- Habilidade para definir hierarquias da rede, permitindo a detecção e a distinção entre as máquinas que estão inativas e as que são inalcançáveis;
- Notificações quando problemas de serviços ou de máquinas ocorrerem ou são resolvidas (por e-mail, SMS, *pager* ou algum método definido pelo usuário);
- Capacidade para definir manipuladores de eventos a serem executados durante eventos de serviços para uma resolução pró-ativa do problema;
- Rotação automática de arquivos de *logs*;
- Suporte para implantação de servidores de monitoramento redundantes;
- Suporte para implantação em um modelo distribuído com vários servidores, coletando dados e notificando a um servidor central;
- Interface web opcional para visualização de rede atual, histórico de notificações de status, problemas;
- Pode ser configurado para funcionar através de *firewalls*, túneis VPN, túneis SSH, e através da Internet;
- Capacidade de utilizar uma variedade de protocolos de rede, incluindo HTTP, SNMP, e SSH, para realizar o monitoramento;

[dC 10] faz uma adaptação da ferramenta Nagios para o monitoramento de máquinas virtuais, com a finalidade de integrar informações de monitoramento de máquinas físicas com informações de monitoramento de máquinas virtuais e seus relacionamentos.

Em [Wil 11] é sugerido a utilização da ferramenta Nagios em ambientes de computação em nuvem e ambientes virtualizados, onde o número de servidores e serviços não é fixa e sua escala pode aumentar e diminuir. São discutidos os desafios enfrentados neste ambiente e é proposta a utilização de soluções como o DNX (Distributed Nagios eXecutor - Extensão modular para o Nagios, que permite sua execução de forma distribuída) e Mod-Gearman como forma de lidar com instalações do Nagios em ambientes de computação em nuvem.

A facilidade de adaptação do Nagios, que se utiliza de *plugins* que executam verificações ao invés de monitorar os *hosts* por conta própria, o torna uma solução modular e flexível. Esses fatores aliados a uma extensa documentação e suporte através da comunidade faz com que diversos trabalhos na área de monitoramento o utilizem como principal ferramenta de suporte.

3.3 PCMONS

O PCMONS, sigla para (*Private Cloud Monitoring System*), ou em português, Sistema de Monitoramento de Nuvens Privadas é uma ferramenta desenvolvida por [CHA 10] para o monitoramento de ambientes de nuvens privadas. Implementado baseando-se na arquitetura genérica para o monitoramento de ambientes de nuvens privadas descrita na seção 3.1, o PCMONS age principalmente na camada de integração, em atividades como coleta e preparo das informações relevantes para a camada de visualização. De acordo com [DC 11], o sistema está dividido em vários módulos, com o propósito de simplificar futuras adaptações para ferramentas específicas e facilitando o seu estudo e aplicação:

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 3.2 mostra os módulos da ferramenta descritos e a interação entre eles.

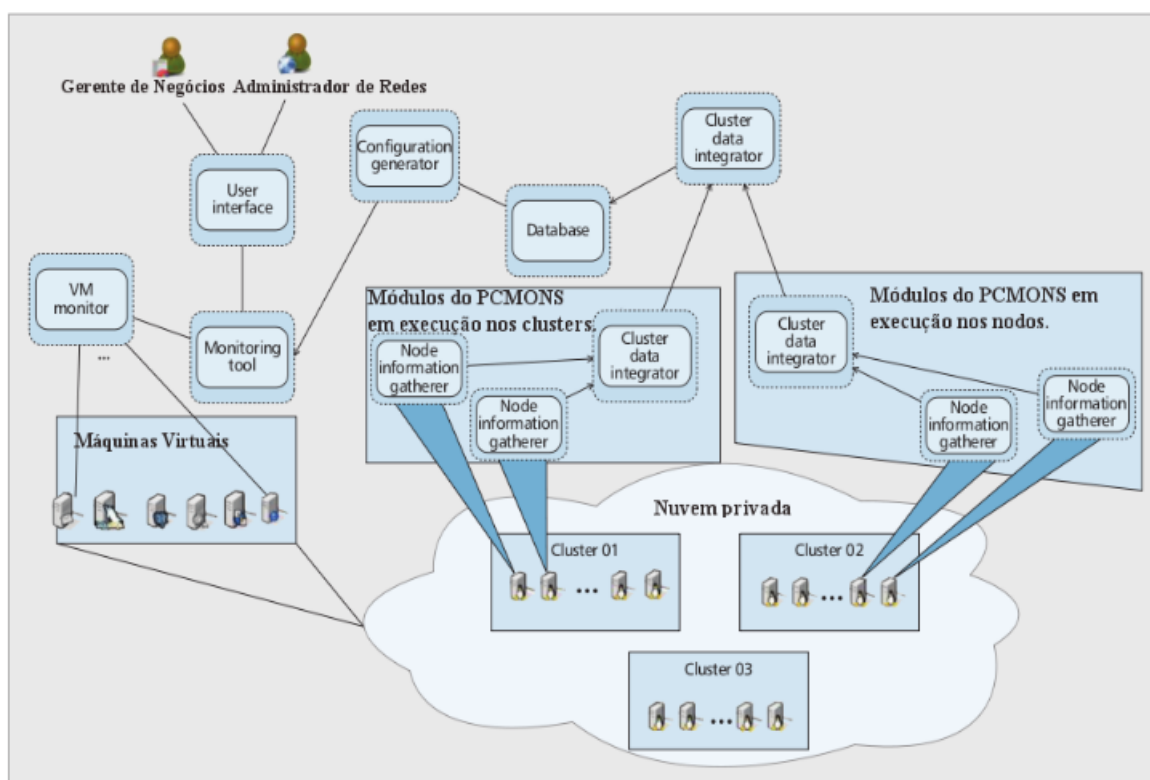


Figura 3.2: Arquitetura da ferramenta PCMONS e interação entre seus componentes. Adaptado de [DC 11].

No atual estágio de desenvolvimento, o PCMONS é compatível com o *Eucalyptus*, descrito na seção 2.3.2 na camada de infraestrutura e o Nagios, descrito na seção 3.2, na camada de visualização. A ênfase do monitoramento é dada as máquinas virtuais inicialmente, devido à falta de soluções nessa área. De acordo com [CHA 10] seu desenvolvimento considera a facilidade de integração com outras ferramentas para computação em nuvem, através do desenvolvimento de extensões. Isto é, não há um forte acoplamento ao Eucalyptus ou ao Nagios, os quais são um módulo do sistema, utilizados apenas quando configurados.

3.4 Conclusão

O presente capítulo apresentou uma introdução sobre monitoramento e sua importância na computação. Um modelo de arquitetura genérica para monitoramento de nuvens privadas foi apresentado. Foram introduzidas as ferramentas para monitoramento de *hosts* e serviços e monitoramento de um ambiente de nuvem privada Nagios e PCMONS.

Capítulo 4

Desenvolvimento

4.1 Introdução

O PCMONS, em sua versão atual, age principalmente na camada de integração, em atividades como coleta e preparo das informações relevantes para a camada de visualização. Para provar a facilidade de integração da ferramenta a outras soluções que provêm computação em nuvem na modalidade IaaS, foi implementada uma adaptação para aumentar sua compatibilidade e permitir sua utilização juntamente com a ferramenta OpenNebula.

Atualmente existem diversas soluções para a implantação de ambientes de computação em nuvem. Além das ferramentas Eucalyptus e OpenNebula descritas nas seções 2.3.2 e 2.3.1 respectivamente, pode-se citar as ferramentas de código aberto OpenStack (<http://openstack.org>), XCP (Xen Cloud Platform - <http://xen.org/products/cloudxen.html>), Nimbus (<http://www.nimbusproject.org>)

Como motivação para escolha da ferramenta OpenNebula no desenvolvimento desse trabalho, foi feita uma avaliação das funcionalidades das principais ferramentas de código aberto para provimento de computação em nuvem na modalidade IaaS. Essa análise possibilitou a elaboração de uma lista, que segue abaixo, contendo as principais vantagens encontradas em relação a outras soluções.

- ✓ Possui uma interface de administração superior onde pode-se executar diversas ações sobre as máquinas virtuais como migrar, suspender e reiniciar. Outras ferramentas permitem apenas a utilização das funcionalidades oferecidas pela interface EC2.
- ✓ Melhores políticas para alocações de recursos. Seu algoritmo pode ser ajustado para permitir capacidades de reservas avançadas. Existem algumas pesquisas para o desen-

volvimento de escalonadores utilizando os princípios de tecnologia verde, voltada para o impacto dos recursos tecnológicos no meio ambiente [SOT 09].

- ✓ Permite o gerenciamento completo dos serviços, incluindo redes privadas para interconectar máquinas de serviço. Como o Eucalyptus está ligado à interface EC2, não é possível definir redes virtuais [SOT 09].
- ✓ Possui suporte para contextualização antecipada. Com isso é possível incluir dados de contexto, para que uma máquina virtual se auto-configure durante sua inicialização, utilizando esses dados (IP, hostname, licenças de software, chaves SSH).
- ✓ Possui uma API para permitir a extensão de seus recursos. Tanto para desenvolvimento de aplicações sobre a plataforma, como para integrar suas tecnologias de armazenamento, virtualização e rede [COR 10].
- ✓ Pode ser utilizado para a implantação de nuvens híbridas. Com isso é possível aumentar a capacidade de uma nuvem privada, instanciando máquinas virtuais tanto localmente como em uma nuvem pública.
- ✓ Possui uma arquitetura completamente modular, facilitando sua integração com diferentes tecnologias [COR 10].
- ✓ A versão de código aberto do Eucalyptus não vem recebendo suporte adequado por parte dos desenvolvedores e não recebe atualizações há quase um ano.

Desde 2010, com o lançamento da versão 2.0 do OpenNebula, a cada dois meses em média, uma nova versão da ferramenta é lançada pelos desenvolvedores, com diversas melhorias e novas funcionalidades. Esse grande esforço por parte dos desenvolvedores, aliado a uma comunidade adepta e uma vasta documentação, tornam o OpenNebula um solução completa e adequada para pesquisas, estudos e desenvolvimentos em geral, na área de computação em nuvem.

Nas próximas seções são apresentadas as modificações realizadas nos módulos da ferramenta PCMONS. Todos esses módulos são descritos em detalhes na seção 3.3.

4.2 Implementação

O funcionamento básico do PCMONS ocorre da seguinte maneira: o sistema reúne diversas informações a respeito das máquinas virtuais em cada um dos nós da nuvem, como IPs, usuários que as instanciaram, e números de identificação das mesmas e os envia para a camada de visualização que tem como responsabilidade criar os arquivos necessários para o monitoramento destas máquinas virtuais.

A) Cluster Integrador de Dados

No arquivo de configuração desse módulo foram feitas algumas alterações para facilitar a integração com outras ferramentas de IaaS. Uma nova variável denominada INFRA foi criada e é responsável por guardar o nome da ferramenta de infraestrutura sendo utilizada. A Figura 4.1 mostra o arquivo `cluster_setting.py`, onde essa variável é definida. Para que todos os nós que estão sendo utilizados para armazenamento de máquinas virtuais pelo OpenNebula sejam mapeados corretamente, esse módulo fica responsável por examinar a variável INFRA definida pelo usuário, e executar o trecho de código adequado, correspondente a ferramenta definida. A Figura 4.2 mostra o método responsável pela escolha da ferramenta de infraestrutura e pelo mapeamento dos nós.

```
#You can choose here your IaaS Software (eucalyptus, opennebula)
INFRA = 'opennebula'
#Configuration path for OpenNebula (self-contained installation).
#Must have a folder $PATH_TO_OPENNEBULA/bin/ with commands like onehost, onelist, etc.
#If you are using a system-wide installation, use PATH_TO_OPENNEBULA = "/usr instead.
PATH_TO_OPENNEBULA = "/srv/cloud/one"
#PATH_TO_OPENNEBULA = "/usr"
```

Figura 4.1: Modificações feitas no arquivo `cluster_config.py` do módulo Cluster Integrador de Dados.

Nesse mesmo módulo foi criado um método responsável pela atualização das informações das máquinas virtuais que estão sendo gerenciadas. A Figura 4.3 mostra o trecho de código responsável por essa atualização. As classes responsáveis pelo acesso ao banco de dados para armazenamento das informações das máquinas virtuais não sofreram modificações, já que essas informações e o acesso a base de dados para inserções e atualizações se mantiveram iguais para ambas as ferramentas de IaaS.

B) Coletor de Informações do Nó

```

def grepNodes(self, infra=cluster_config.INFRA,
file=cluster_config.PATH_TO_EUCALYPTUS_CONFIG_FILE,
pattern="NÓDES"):
    """
    provides the list eucalyptus nodes
    """
    if infra == 'eucalyptus':
        fileConf = open(file,'r')
        Nodes = ''
        for line in fileConf:
            if pattern in line:
                Nodes = Nodes + line
                Nodes = Nodes.split('\n')
                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

    else:
        return 'not implemented yet'

```

Figura 4.2: Método responsável pelo mapeamento dos nós em cada um dos clusters da nuvem.

Este módulo é o responsável por coletar as informações locais em cada um dos nós da nuvem e reunir informações sobre as máquinas virtuais locais e enviar para o *Data Integrator Cluster*. A Figura 4.4 mostra a implementação da função responsável por obter todas essas informações.

C) Interface Com Usuário

No módulo interface, responsável pela organização e apresentação das informações coletadas alguns trechos de código foram alterados para adaptar o PCMONS a última versão estável da ferramenta de monitoramento Nagios, a 3.2.0, descrita em detalhes na seção 3.2.

D) Instalação

Os primeiros passos para instalação do PCMONS são: a criação de um usuário no banco de dados MySQL e a criação da base de dados e tabelas. Para facilitar o processo de criação de todas as tabelas para armazenamento das informações das máquinas virtuais para fins de monitoramento e de dados históricos, foi desenvolvido um *script* que pode ser executado através de alguma ferramenta para gerenciamento de bancos de dados MySQL, como

```

def update_opennebula_vms_info(self):
    """
    update informations about an opennebula vms
    """
    for node in self.nodes:
        vms_on_node = self.get_vm_list_running_on_node
        (node)
        if len(vms_on_node) > 0:
            for vm on node in vms_on_node:
                self.db.update_vm_ip_hostname(node,
                hostname=vm_on_node['node_hostname'],
                instance_id=vm_on_node['instance_id'])

```

Figura 4.3: Método responsável pela atualização de informações das máquinas virtuais.

```

def opennebula(self):
    """
    This function get information about vms managed by opennebula
    """
    hostname = commands.getoutput('hostname')
    vms = [ ]
    vm = { }
    vms_running_host = commands.getoutput("virsh list | cut -c5-14")
    vms_running_host = vms_running_host.split()[3:]
    for vm in vms_running_host:
        vm = {'node_hostname':hostname,'instance_id':vm.replace
        {'one','i'},}
        vms.append(vm)
    if len(vms) > 0:
        logging.debug("Vms returned :%s"%vms)
    else:
        logging.debug("No vm was returned")

    return vms

```

Figura 4.4: Método responsável pela coleta de informações sobre máquinas virtuais no nó.

o PhpMyAdmin, ou através do terminal do MySQL. Esse *script* encontra-se anexado a esse trabalho no APÊNDICE E.

A instalação do PCMONS é simples. Todos os arquivos que se encontram na pasta `running_vms/node/` são responsáveis pelo módulo Node e devem ser executados em todos os nós do sistema. Todos os outros arquivos devem ser executados na máquina cluster, que possui a instalação do Nagios. Para facilitar o trabalho de execução dos módulos do PCMONS todas as vezes que as máquinas são reiniciadas, foram criados dois *scripts* que devem ser adicionados na inicialização do sistema operacional para que sejam executados automaticamente. Esses *scripts* verificam se os arquivos necessários para a execução do PCMONS estão presentes no sistema e os executa em *background*. Os arquivos chamados de `pcmons-cluster.sh` e `pcmons-node.sh` devem ser adicionados na inicialização da máquinas que farão o papel de cluster e nós do sistema e podem ser vistos no APÊNDICE F e APÊNDICE G.

4.3 Estudo de Caso

Para testar a implementação da ferramenta PCMONS compatível com o OpenNebula, foi realizado um estudo de caso, onde foi implantado um ambiente de nuvem privada no Laboratório de Redes e Gerência (LRG) no Departamento de Informática e Estatística (INE) da Universidade Federal de Santa Catarina, utilizando esta ferramenta. Para exemplificar uma situação prática para uso da estratégia de monitoramento e ferramenta desenvolvida, foram disponibilizadas na nuvem imagens de máquinas virtuais, para que um usuário possa instanciar simulando assim um ambiente de provedor de hospedagem.

4.3.1 Ambiente

A tabela 4.1 abaixo mostra a infraestrutura de software e hardware utilizadas nessa implantação. Uma das máquinas foi empregada como *front-end*, responsável por controlar a nuvem, enquanto a outra foi responsável pela execução do hipervisor e pela alocação das máquinas virtuais.

Tabela 4.1: Descrição do ambiente utilizado para o estudo de caso.

Hardware	Software	Papel no sistema
AMD Phenom(tm) 9650 Quad-Core, 2310 MHZ 4GB DDR2 667MHz HD SAMSUNG HD753LJ 750GB ATA-8-AC	Ubuntu 10.04.4 LTS OpenNebula 3.2 Nagios Core 3.2.0	OpenNebula <i>Front-end</i>
Intel(R) Core(TM) 2 Quad Q8200, 2.33 GHz 3GB DDR2 333MHz RAM HD SAMSUNG HD322HJ 320GB ATA	CentOS 5.8 (Final) Xen 3.1.2	Nó

A escolha do sistema operacional Ubuntu na máquina *front-end* do OpenNebula se deu devido a grande popularidade dessa distribuição GNU/Linux e sua extensa documentação e suporte por parte dos fabricantes e da comunidade. Isso fez com que a maioria das dependências de ferramentas e bibliotecas necessárias fossem facilmente instaladas utilizando o gerenciador de pacotes do sistema, o APT (*Advanced Packaging Tool* - Gerenciador de pacotes para o Sistema Operacional GNU/Linux Debian e derivados). Já a escolha do sistema operacional CentOS na máquina nó, responsável pelo gerenciamento e armazenamento das máquinas virtuais, se deu

devido a facilidade de instalação e configuração, principalmente do hipervisor Xen, utilizando o gerenciador de pacotes semelhante ao APT, chamado YUM (*Yellow dog Updater, Modified.*).

A escolha do hipervisor de modo geral está atrelada ao software para computação em nuvem escolhido. O OpenNebula possui suporte para os hipervisores de código aberto Xen e KVM e possui suporte também para a solução proprietária VMWare ESX. O Xen é um monitor de máquina virtual para arquiteturas x86, x86-64, IA-32, IA-64 e PowerPC que permite a execução de múltiplos sistemas operacionais compartilhando o mesmo hardware de modo seguro, mas sem sacrificar o desempenho ou sua funcionalidade. Versões modificadas de Linux e NetBSD podem ser usadas como base. Diversos sistemas Unix modificados podem ser executados [BAR 03]. O KVM (Kernel-based Virtual Machine) é uma solução de virtualização completa para Linux em hardware x86 contendo extensões de virtualização (Intel VT ou AMD-V). É constituída por um módulo de *kernel* carregável, que fornece a infraestrutura de virtualização do núcleo e um módulo de processador específico. Usando o KVM, é possível executar várias máquinas virtuais rodando Linux ou Windows sem necessidade de modificação do kernel. Cada máquina virtual possui um hardware virtualizado privado: placa de rede, disco e placa de vídeo. O Xen é o mais indicado ao estudo de caso, pelo fato de implementar o mecanismo de paravirtualização, técnica na qual uma camada de hardware virtual muito similar ao hardware real é adicionada ao hipervisor [Tim 09], não precisando de suporte nativo à virtualização pelo hardware, porém existe a necessidade de adaptação do núcleo do sistema operacional. No caso do KVM, não há a necessidade de se alterar o núcleo do sistema operacional, pois ele está disponível como um módulo no núcleo padrão na maioria das distribuições Linux, porém existe a necessidade de utilização de tecnologia de hardware especializado, não implementada nas máquinas utilizadas nesse estudo de caso. O Nagios foi escolhido como ferramenta de monitoramento por ser uma solução de código aberto amplamente documentada com grande suporte por parte da comunidade. Outro fator é sua arquitetura modular, que pode ser complementada através da criação de plugins (módulo adicionais), aumentando sua possibilidade e flexibilidade de uso.

4.3.2 Criação da nuvem privada

4.3.2.1 OpenNebula 3.2.1

A máquina que mantém a instalação OpenNebula é chamada de *front-end*. Esta máquina precisa ter acesso ao repositório de armazenamento de imagens (montagem direta ou

através da rede), e conectividade de rede para cada nó. A base de instalação do OpenNebula possui aproximadamente 10MB e inclui:

- *Daemon* de monitoramento (**oned**)
- Escalonador (**mm sched**)
- *Daemon* de monitoramento e contabilidade (**onecctd**)
- Servidor de interface *web* (**sunstone**)
- Servidores de API para nuvem (Amazon EC2 e OCCI)

Estes componentes se comunicam através do protocolo XML-RPC (Extensible Markup Language e Remote Procedure Call) e pode ser instalado em máquinas diferentes por razões de segurança ou desempenho.

Os requisitos de software necessários para a instalação do OpenNebula são:

- compilador g++ (>= 4.0)
- bibliotecas xmlrpc-c (>= 1.06)
- ferramenta de compilação scons (>= 0.98)
- bibliotecas sqlite3 (>= 3.6)
- bibliotecas mysql (>= 5.1)
- bibliotecas libxml2 (>= 2.7)
- bibliotecas libssl (>= 0.9.8)
- interpretador ruby (>= 1.8.7)

Na distribuição Ubuntu todas essas dependências podem ser instaladas utilizando a ferramenta apt-get, utilizando o seguinte comando no terminal:

```
$ sudo apt-get install g++ libxmlrpc-c3-dev scons libsqlite3-dev \
libmysqlclient-dev libxml2-dev libssl-dev ruby
```

A instalação foi feita a partir da compilação do código fonte da aplicação sem a utilização de pacotes binários prontos, pois não havia uma versão de 32 bits compatível com o hardware utilizado. O OpenNebula pode ser baixado através da sua página oficial de *downloads* (<http://downloads.opennebula.org/>). Nessa implantação foi utilizada a versão 3.2.1 datado de 30 de Janeiro de 2012, por ser a última versão estável desenvolvida até a realização deste trabalho. Os passos seguidos para a compilação do código fonte foram:

Download e descompressão do *tarball* (Tape ARchive - Formato para compressão de arquivos) contendo o OpenNebula:

```
$ tar xzf opennebula-3.2.1.tar.gz
```

Executar o comando *scons* dentro da pasta onde a instalação será feita para compilação do OpenNebula:

```
$ cd opennebula-3.2.1/  
$ scons -j2  
$ ./install.sh -d /srv/cloud/one
```

A instalação do OpenNebula cria um novo usuário e grupo chamado *oneadmin* na máquina *front-end*. Essa conta será utilizada para executar o OpenNebula e para tarefas de administração e manutenção. Alguns componentes de OpenNebula necessitam de bibliotecas Ruby extras. Existe um *script* localizado na pasta */usr/share/one/install_gems* para facilitar a instalação dessas bibliotecas. Para este estudo de caso, foi necessária a instalação das bibliotecas para implementação da interface EC2 e da interface gráfica SunStone utilizando o seguinte comando:

```
$ ./install_gems sunstone cloud
```

O OpenNebula EC2 Query é um serviço web que permite iniciar e gerenciar máquinas virtuais em sua instalação do OpenNebula através da interface de consulta Amazon EC2. Desta forma, pode-se usar qualquer ferramenta de consulta EC2 ou utilitário para acessar sua nuvem privada. Esse servidor web é implementado sobre a camada OpenNebula API (OCA) que expõe todos os recursos de uma nuvem OpenNebula privada e o Sinatra, um *arcabouço* web leve amplamente utilizado. O serviço é configurado através do arquivo */etc/one/econe.conf*, onde deve-se configurar os parâmetros operacionais básicos para o servidor web EC2. A Figura 4.5 mostra todas configurações utilizadas para esse estudo de caso.

O OpenNebula Sunstone é uma interface gráfica destinada a usuários e administradores que simplifica as operações de gerenciamento típicos em infraestruturas de nuvens privadas e híbridas. Permite o gerenciamento de todos os recursos do OpenNebula e realizar várias operações sobre eles [Ope 11]. Através dessa interface é possível instanciar, desligar, reiniciar máquinas virtuais, gerenciar e fazer *upload* de imagens, adicionar e remover nós e gerenciar contas de usuários. Para utilização da interface SunStone é necessária sua configuração através do arquivo */etc/sunstone-server.conf*. A configuração utilizada para esse estudo de caso pode ser vista na Figura 4.6.


```

# OpenNebula sever contact information
:one_xmlrpc: http://localhost:2633/RPC2

# Host and port where econe server will run
:server: 150.162.63.32
:port: 4567

# SSL proxy that serves the API (set if is being used)
#:ssl_server: fqdm.of.the.server

# Authentication protocol for the econe server:
#   ec2, default Access key and Secret key scheme
#   x509, for x509 certificates based authentication
:auth: ec2

# VM types allowed and its template file (inside templates directory)
:instance_types:
  :m1.small:
    :template: m1.small.erb
  :m1.large:
    :template: m1.large.erb

```

Figura 4.5: Arquivo de configuração do OpenNebula EC2 Query.

```

# OpenNebula sever contact information
:one_xmlrpc: http://localhost:2633/RPC2

# Server Configuration
:host: 150.162.63.32
:port: 9869

:auth: sunstone

# VNC Configuration
:vnc_proxy_base_port: 29876
:novnc_path: /srv/cloud/one/share/novnc

```

Figura 4.6: Arquivo de configuração do OpenNebula Sunstone.

Após as configurações, as interfaces EC2 e Sunstone podem ser inicializadas utilizando os seguintes comandos:

```
$ oned start
$ econe-server start
$ sunstone-server start
```

Não é necessária a instalação de nenhum componente do OpenNebula nos nós, porém para comunicação entre os componentes e para manuseio e distribuição das máquinas virtuais são necessárias as seguintes ferramentas:

- Servidor SSH
- Hipervisor
- Ruby >= 1.8.7

Para uma instalação do Xen bastante facilitada com a utilização do gerenciador de pacotes YUM, basta executar os comandos abaixo no terminal, reiniciar a máquina e iniciá-la utilizando o *kernel* Xen modificado:

```
# yum install xen virt-manager kernel-xen
# chkconfig xend on
# reboot
```

Também é necessária a criação do usuário e grupo *oneadmin* com o mesmo *uid* (User **id**entifier, código único que identifica cada usuários em um sistema baseado em Unix) e *gid* dos que foram criados na máquina *front-end*:

```
# groupadd --gid 1001 oneadmin
# useradd --uid 1001 -g oneadmin -d /var/lib/one oneadmin
```

Existem várias possibilidades para lidar com o armazenamento e a transferência de imagens de máquinas virtuais utilizando o OpenNebula. Nesse estudo de caso utilizou-se um sistema de arquivos compartilhado pela rede como sistema de armazenamento onde todas as imagens são compartilhadas entre todos os nós utilizando um sistema de arquivos NFS (Network File System - Sistema de arquivos distribuídos desenvolvido pela *Sun Microsystems*). O NFS é uma maneira de compartilhar arquivos entre máquinas de uma rede, como se estes arquivos estivessem localizados no disco rígido local do cliente. É útil para compartilhar diretórios de arquivos entre múltiplos usuários da mesma rede [RHE 03]. Para utilização do NFS

é necessária sua instalação e configuração nas máquinas nós. Para instalação, basta utilizar o seguinte comando no terminal:

```
# yum install nfs-utils
```

Após a instalação, o arquivo `/etc/fstab` deve ser editado, para que o sistema operacional realize a montagem do sistema de arquivos utilizado pelo OpenNebula durante a inicialização, adicionando a seguinte entrada:

```
150.162.63.32:/srv/cloud/one /srv/cloud/one nfs defaults 0 0
```

onde 150.162.63.32 é o IP externo da máquina *front-end*.

A Figura 4.7 mostra a tela inicial da interface gráfica OpenNebula Sunstone-Server. Através dela é possível ver um sumário com informações de recursos como imagens disponíveis, nós monitorados, quantidade de máquinas virtuais instanciadas. É possível a visualização de gráficos da utilização de recursos (memória, CPU) e estatísticas de redes dos nós.

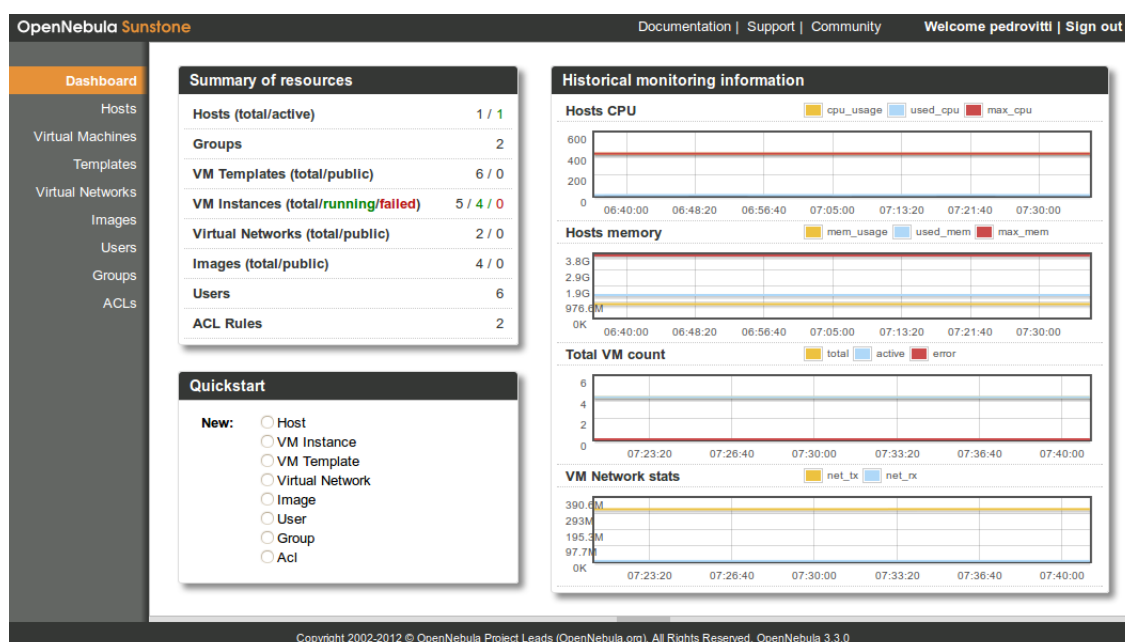


Figura 4.7: Tela inicial da interface gráfica OpenNebula Sunstone-Server.

Na figura 4.8 é mostrada a tela do SunStone-Server onde acontece toda a parte de gerenciamento de imagens.

Para esse estudo de caso foram criadas duas imagens com a instalação mínima dos sistemas operacionais baseados em Linux Ubuntu 10.04.03 LTS e CentOS 5.7. Para permitir a

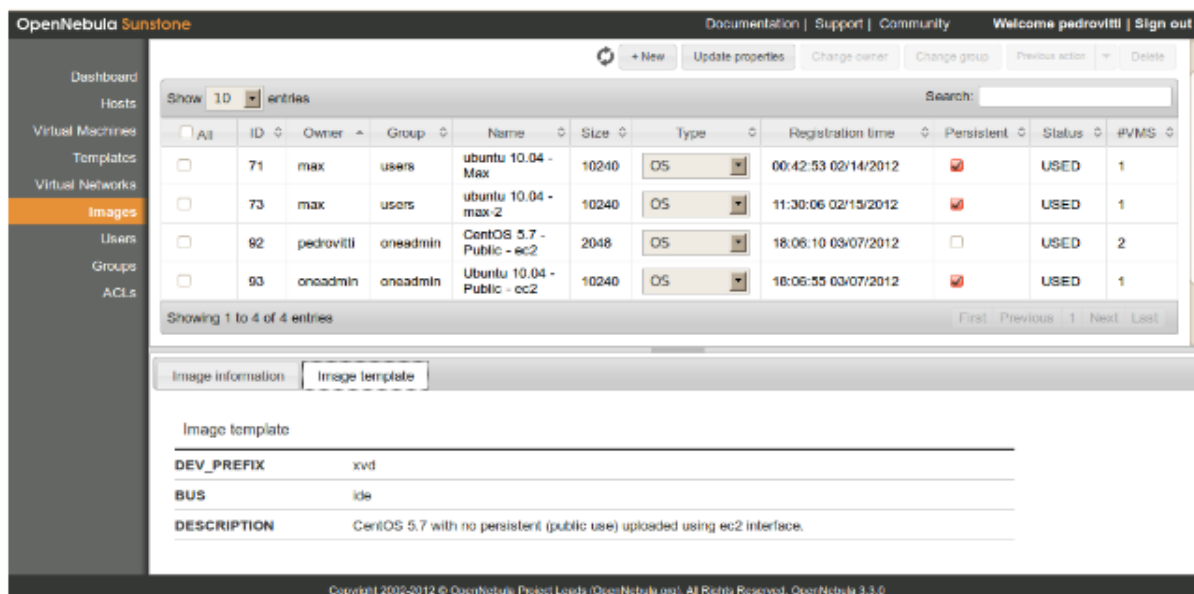


Figura 4.8: Tela de gerenciamento de imagens através do OpenNebula Sunstone-Server.

personalização e instalação de aplicativos nas imagens, foi utilizado o mecanismo de contextualização implementado pelo próprio OpenNebula. Esse mecanismo permite passar arquivos e parâmetros de configuração para uma máquina virtual durante sua instanciação utilizando uma imagem ISO. No *template* da máquina virtual pode-se especificar o conteúdo do arquivo ISO (arquivos e diretórios), e informar em qual dispositivo a imagem ISO estará acessível e especificar os parâmetros de configuração que serão gravados em um arquivo para uso posterior dentro da máquina virtual. Os *templates* utilizados nesse estudo de caso para instanciação das imagens e passagem de parâmetros estão incluídos nos APÊNDICE A e APÊNDICE B. O *scripts* que são executados na inicialização das máquinas virtuais e são responsáveis por todas as configurações e instalações de ferramentas extras estão incluídos nos APÊNDICE C e APÊNDICE D. Uma única modificação nas máquinas virtuais foi feita para a utilização desse mecanismo. Foi adicionado um *script* na inicialização do sistema operacional da máquina virtual que tem a função de montar a partição ISO e executar o *script* de configuração. Esse uso de chamada indireta, onde um *script* chama o outro facilita bastante a instalação e adaptação das máquinas virtuais, já que a modificação do *script* de configuração pode ser feita localmente antes da instanciação da máquina virtual, ao invés da necessidade de alterar todas as máquinas virtuais. A Figura 4.9 mostra o trecho de código utilizado nas máquinas virtuais para execução do *script* de configuração.

```
#!/bin/sh -e
mount -t iso9660 /dev/sdc /mnt
if [ -f /mnt/context.sh ]; then
    . /mnt/init.sh
fi
umount /mnt
exit 0
```

Figura 4.9: Script executado na inicialização das máquinas virtuais.

4.3.2.2 Nagios 3.2.0 e PCMONS

Para instalação da ferramenta de monitoramento Nagios na mesma máquina onde está instalado o OpenNebula, novamente foi utilizado o gerenciador de pacotes das distribuições GNU/Linux baseadas no *debian*, o APT e seu repositório padrão. Para iniciar a instalação basta digitar os seguintes comandos no terminal do sistema:

```
$ sudo apt-get install apache2
$ sudo apt-get install mysql-server mysql-client
$ sudo apt-get install php5 libapache2-mod-php5
$ sudo apt-get install nagios3
```

Com isso, será instalada e configurada a ferramenta Nagios, versão 3.2.0, denominada Nagios3, juntamente com todas as dependências necessárias, incluindo o servidor Apache2, Postfix e outros. O Postfix é uma agente de transferência de e-mails, nesse caso utilizado para enviar alertas de monitoramento. Durante a instalação, o usuário deve selecionar como deseja configurar o Postfix. Nesse caso, selecionou-se a opção 'Internet Site', já marcada por padrão. É solicitado ao usuário também a definição de uma senha para o usuário "nagiosadmin", utilizado para acesso e administração da interface de monitoramento da ferramenta. Os arquivos de configurações podem ser encontrados na pasta */etc/nagios3* e os *plugins* responsáveis pelo monitoramento e verificação dos serviços na pasta */etc/nagios-plugins*.

Desse modo, uma instalação básica do Nagios está instalada no sistema e sua interface pode ser acessada localmente acessando a página <http://localhost/nagios3>. Para que cada nova máquina virtual que entra ou sai da nuvem seja mostrada na interface do Nagios, um *script* de geração de configuração para o Nagios deve ser adicionado no *cron* (programa que executa comandos agendados nos sistemas operacionais do tipo Unix) do sistema operacional. Se o arquivo de configuração das máquinas virtuais tiver sofrido alguma alteração, é feito uma recarga do serviço do Nagios.

A execução da ferramenta PCMONS é descrita em detalhes na seção 4.2. A figura 4.10 mostra a interface da ferramenta Nagios, sendo utilizada pelo PCMONS, onde apresenta uma listagem de todas as máquinas virtuais sendo monitoradas pelo sistema no momento. É possível observar também a nomenclatura especial utilizada para o *hostname* das máquinas virtuais. Esse nome consiste na junção do nome do usuário que instanciou a máquina, com o id da máquina virtual fornecido pela ferramenta de IaaS, mais o nome da máquina física onde a máquina virtual esta sendo executada.

Host ↑↓	Status ↑↓	Last Check ↑↓	Duration ↑↓	Status Information
gateway	UP	2012-05-12 22:21:23	45d 14h 54m 27s	PING OK - Packet loss = 0%, RTA = 0.32 ms
localhost	UP	2012-05-12 22:17:53	214d 23h 45m 26s	PING OK - Packet loss = 0%, RTA = 0.03 ms
oneadmin_i-243_stratus	UP	2012-05-12 22:18:03	0d 0h 53m 59s	PING OK - Packet loss = 0%, RTA = 0.14 ms
oneadmin_i-244_stratus	UP	2012-05-12 22:19:33	0d 0h 53m 19s	PING OK - Packet loss = 0%, RTA = 0.17 ms
oneadmin_i-278_stratus	UP	2012-05-12 22:21:03	0d 0h 6m 35s	PING OK - Packet loss = 0%, RTA = 0.15 ms
oneadmin_i-279_stratus	UP	2012-05-12 22:20:13	0d 0h 6m 25s	PING OK - Packet loss = 0%, RTA = 0.21 ms

Figura 4.10: PCMONS: Listagem das máquinas virtuais monitoradas pelo sistema.

As métricas de monitoramento utilizadas nesse estudo de caso provêm um conjunto mínimo de informações sobre os recursos monitorados e englobam dois conjuntos de monitoramento: um referente ao desempenho/disponibilidade da máquina virtual (carga, memória, ping e SSH) e outro em relação ao serviço que está sendo executado (número de conexões HTTP). A figura 4.11 demonstra todas as métricas utilizadas para monitoramento no estudo de caso e uma breve explicação a respeito de cada uma. A figura 4.12 mostra a interface do Nagios com todos os serviços, ativos (PING e SSH) e passivos (HTTP_CONNECTIONS, LOAD, RAM) que estão sendo monitorados pelo PCMONS.

Na figura 4.13 é possível visualizar detalhadamente o *status* de todas as métricas que estão sendo monitoradas. A figura 4.14 mostra a organização das máquinas virtuais num *hostgroup* próprio e a organização das máquinas virtuais por nós onde estão sendo executadas. Nota-se, que, para esse estudo de caso, todas as máquinas virtuais estão sendo executadas no mesmo nó. Esse mapeamento que informa quais máquinas virtuais estão hospedadas em cada nó é uma característica importante que facilita a detecção rápida de falhas e sua consequente correção.

Serviço	Motivo	Tipo de Monitoramento
Memória	Verificar utilização da memória e a necessidade de instanciar novos recursos.	Passivo
Carga no Sistema	Verificar carga do sistema e possíveis problemas de desempenho	Passivo
Acesso	Verificar disponibilidade de acesso à MV usando o protocolo SSH.	Ativo
Tempo de Resposta	Teste para verificar o desempenho da rede. A verificação é feita através da ferramenta PING.	Ativo
# de Conexões HTTP	Verificar a demanda e o número de conexões e comparar com o desempenho.	Passivo










Figura 4.11: Métricas monitoradas no estudo de caso.

4.4 Conclusão

Esse capítulo apresentou a motivação em relação à escolha da ferramenta OpenNebula e todos os passos seguidos para adaptá-lo ao PCMONS. Foi descrito um estudo de caso realizado no Laboratório de Redes em Gerência, onde foi feita a implantação e o monitoramento de uma nuvem privada.

Status Grid For All Host Groups

[Vms on Node 150.162.63.31 \(Node 150.162.63.31\)](#)

Host	Services	Actions
oneadmin_i-243_stratus	HTTP_CONNECTIONS LOAD PING RAM SSH	  
oneadmin_i-244_stratus	HTTP_CONNECTIONS LOAD PING RAM SSH	  
oneadmin_i-278_stratus	HTTP_CONNECTIONS LOAD PING RAM SSH	  
oneadmin_i-279_stratus	HTTP_CONNECTIONS LOAD PING RAM SSH	  

[vms \(Virtual Machines\)](#)












Host	Services	Actions
oneadmin_i-243_stratus	HTTP_CONNECTIONS LOAD PING RAM SSH	  
oneadmin_i-244_stratus	HTTP_CONNECTIONS LOAD PING RAM SSH	  
oneadmin_i-278_stratus	HTTP_CONNECTIONS LOAD PING RAM SSH	  
oneadmin_i-279_stratus	HTTP_CONNECTIONS LOAD PING RAM SSH	  

Figura 4.12: PCMONS: Serviços sendo monitorados no momento.





















































oneadmin_i-243_stratus	HTTP_CONNECTIONS  	OK	2012-05-12 22:19:31	0d 0h 1m 32s	1/4	0
	LOAD  	OK	2012-05-12 22:20:13	0d 0h 0m 50s	1/4	OK - load average: 0.00, 0.00, 0.00
	PING  	OK	2012-05-12 22:17:07	0d 0h 53m 56s	1/4	PING OK - Packet loss = 0%, RTA = 0.17 ms
	RAM  	OK	2012-05-12 22:20:41	0d 0h 0m 22s	1/4	64 81/83
	SSH  	OK	2012-05-12 22:19:44	0d 0h 51m 19s	1/4	SSH OK - OpenSSH_5.3p1 Debian-3ubuntu7 (protocol 2.0)
oneadmin_i-244_stratus	HTTP_CONNECTIONS  	OK	2012-05-12 22:19:43	0d 0h 1m 20s	1/4	0
	LOAD  	OK	2012-05-12 22:20:07	0d 0h 0m 56s	1/4	OK - load average: 0.00, 0.00, 0.00
	PING  	OK	2012-05-12 22:20:47	0d 0h 50m 16s	1/4	PING OK - Packet loss = 0%, RTA = 2.16 ms
	RAM  	OK	2012-05-12 22:20:50	0d 0h 1m 4s	1/4	94 70/83
	SSH  	OK	2012-05-12 22:20:51	0d 0h 50m 12s	1/4	SSH OK - OpenSSH_5.3p1 Debian-3ubuntu7 (protocol 2.0)
oneadmin_i-278_stratus	HTTP_CONNECTIONS  	OK	2012-05-12 22:19:41	0d 1h 17m 51s	1/4	0
	LOAD  	OK	2012-05-12 22:19:41	0d 1h 17m 51s	1/4	OK - load average: 0.00, 0.00, 0.00
	PING  	OK	2012-05-12 22:19:46	0d 0h 4m 39s	1/4	PING OK - Packet loss = 0%, RTA = 0.20 ms
	RAM  	OK	2012-05-12 22:19:41	0d 1h 17m 51s	1/4	97 81/83
	SSH  	OK	2012-05-12 22:16:21	0d 0h 4m 42s	1/4	SSH OK - OpenSSH_5.3p1 Debian-3ubuntu7 (protocol 2.0)
oneadmin_i-279_stratus	HTTP_CONNECTIONS  	OK	2012-05-12 22:19:52	0d 1h 16m 45s	1/4	0
	LOAD  	OK	2012-05-12 22:19:52	0d 1h 16m 45s	1/4	OK - load average: 0.01, 0.06, 0.02
	PING  	OK	2012-05-12 22:18:43	0d 0h 2m 20s	1/4	PING OK - Packet loss = 0%, RTA = 0.19 ms
	RAM  	OK	2012-05-12 22:19:52	0d 1h 16m 45s	1/4	69 44/64
	SSH  	OK	2012-05-12 22:20:01	0d 0h 6m 2s	1/4	SSH OK - OpenSSH_4.3 (protocol 2.0)

Figura 4.13: PCMONS: Status dos serviços monitorados em cada uma das máquinas virtuais.

Vms on Node 150.162.63.31 (Node 150.162.63.31)			
Host	Status	Services	Actions
oneadmin_i-243_stratus	UP	5 OK	  
oneadmin_i-244_stratus	UP	5 OK	  
oneadmin_i-278_stratus	UP	5 OK	  
oneadmin_i-279_stratus	UP	5 OK	  













vms (Virtual Machines)			
Host	Status	Services	Actions
oneadmin_i-243_stratus	UP	5 OK	  
oneadmin_i-244_stratus	UP	5 OK	  
oneadmin_i-278_stratus	UP	5 OK	  
oneadmin_i-279_stratus	UP	5 OK	  

Figura 4.14: PCMONS: *Hostgroup* próprio e mapeamento entre máquinas virtuais e suas respectivas máquinas físicas.

Capítulo 5

Conclusão

A viabilidade de otimização de recursos computacionais faz com que sistemas baseados em computação em nuvem se tornem mais comuns a cada dia. Muitas empresas estão migrando, totalmente ou parcialmente, seus recursos físicos de computação e seus serviços e aplicações para infraestruturas virtuais. Com isso, administradores e gerentes de rede possuem sob sua responsabilidade quantidade cada vez maior de máquinas e serviços. Neste cenário, ferramentas que auxiliem e automatizem tarefas de monitoramento e gerenciamento de máquinas virtuais e seus serviços são bastante convenientes e necessárias.

Na área de monitoramento em nuvens privadas, pouco foi publicado até o momento. Como a abordagem de computação em nuvem é recente, a maioria dos esforços estão voltados para o desenvolvimento de soluções para o fornecimento, padronização e aceitação geral do paradigma.

Atualmente, existem diversas soluções para a implantação de ambientes de computação em nuvem. O OpenNebula, por ser uma ferramenta de código aberto com uma vasta gama de recursos, aliados a um grande esforço por parte dos desenvolvedores e uma comunidade ativa, se mostrou uma das soluções mais completas e ideal para a construção desse ambiente de forma simples e sem maiores investimentos.

Com esse trabalho foi possível provar a facilidade de extensão do PCMONS e torná-lo compatível com a ferramenta que provê o modelo de IaaS em computação em nuvem OpenNebula. Grande parte dessa facilidade devido à sua arquitetura genérica e modular, desenvolvida considerando sua fácil adaptação. 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/>.

5.1 Trabalhos Futuros

Como trabalhos futuros pretende-se ampliar o suporte da ferramenta PCMONS a outras ferramentas de IaaS, como OpenStack ou Nimbus, e ferramentas de monitoramento como Zabbix e Cacti, além de facilitar sua adaptação por terceiros, através de documentação mais consistente e pelo fornecimento de um conjunto de rotinas e padrões para a utilização das suas funcionalidades, através de uma API.

Pretende-se também utilizar e testar a ferramenta em ambientes maiores, com um amplo número de *hosts* e uma grande quantidade de máquinas virtuais, sem limitações de recursos computacionais. Planeja-se também o suporte para a especificação e a verificação do cumprimento de Acordos de Níveis de Serviços, gerando relatórios e possivelmente respostas autônomas para determinados eventos, como a migração de máquinas virtuais em caso de falhas em determinadas máquinas físicas.

Referências

- [BAR 03] BARHAM, P. et al. Xen and the art of virtualization. **SIGOPS Oper. Syst. Rev.**, New York, NY, USA, v.37, n.5, p.164–177, Outubro, 2003.
- [BAR 08] BARTH, W. **Nagios: System and Network Monitoring**. No Starch Press Series. No Starch Press, 2008. 462 p.
- [BRI 09] BRISCOE, G.; MARINOS, A. Digital Ecosystems in The Clouds: Towards Community Cloud computing. **IEEE International Conference on Digital Ecosystems and Technologies**, Istambul, v.1, p.103–108, Junho, 2009.
- [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.
- [COR 10] CORDEIRO, T. et al. Open source cloud computing platforms. [S.l.], v., p.366–371, nov., 2010.
- [dC 10] DE CARVALHO, M. B. **Adaptação da Ferramenta Nagios para o Monitoramento de Servidores Virtuais**. Porto Alegre: Universidade Federal do Rio Grande do Sul, Instituto de Informática, Curso de Ciências da Computação, 2010. Trabalho de Conclusão de Curso.
- [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.
- [Euc 11] Eucalyptus . **Eucalyptus 2.0 Documentation**. Disponível em <http://open.eucalyptus.com/wiki/EucalyptusInstallation_v2.0>. Acesso em: 23 de Abril de 2012.
- [HUA 09] HUANG, H. et al. Building end-to-end management analytics for enterprise data centers. In: PROCEEDINGS OF THE 11TH IFIP/IEEE INTERNATIONAL CONFERENCE ON SYMPOSIUM ON INTEGRATED NETWORK MANAGEMENT, 2009. **Proceedings...** Piscataway, NJ, USA: IEEE Press, 2009. IM'09, p.661–675.
- [HUR 10] HURWITZ, J. et al. **Cloud Computing for Dummies**. 1. ed. EUA: Wiley Publishing, 2010.
- [Maa 11] MILOJIĆ ANDIĆ AND, D.; LLORENTE, I. M.; MONTERO, R. S. Opennebula: A cloud management tool. **Internet Computing, IEEE**, [S.l.], v.15, n.2, p.11 –14, march-april, 2011.
- [Mic 10] Michael Lupacchino on Fri, Jun 25, 2010. **Private or Public Cloud?** Disponível em <<http://blog.nskinc.com/IT-Services-Boston/bid/32590/Private-Cloud-or-Public-Cloud>>. Acesso em: 23 de Abril de 2012.

- [NIS 11] NIST SP 800-145. **The NIST Definition of Cloud Computing**. Disponível em <<http://csrc.nist.gov/publications/nistpubs/800-145/SP800-145.pdf>>. Acesso em: 23 de Abril de 2012.
- [Ope 11] OpenNebula.org. **OpenNebula Documentation**. Disponível em <<http://openebula.org/documentation/documentation>>. Acesso em: 23 de Abril de 2012.
- [RHE 03] RHEL: Red Hat Enterprise Linux 3. **Guia de Administração do Sistema**. Disponível em <http://web.mit.edu/rhel-doc/3/rhel-sag-pt_br-3/ch-nfs.html>. Acesso em: 24 de Abril de 2012.
- [SHA 09] SHAO, Z.; JIN, H.; LU, X. Pmonitor: A lightweight performance monitor for virtual machines. **Education Technology and Computer Science, International Workshop on**, Los Alamitos, CA, USA, v.3, p.689–693, 2009.
- [SMO 12] SMOOT, S. R.; TAN, N. K. **Private Cloud Computing**. 1. ed. EUA: Morgan Kauffman, 2012.
- [SOT 09] SOTOMAYOR, B. et al. Virtual infrastructure management in private and hybrid clouds. **IEEE Internet Computing**, Los Alamitos, CA, USA, v.13, p.14–22, 2009.
- [Tim 09] Tim Jones . **Anatomia de um Hypervisor Linux**. Disponível em <<http://www.ibm.com/developerworks/br/library/l-hypervisor/>>. Acesso em: 23 de Abril de 2012.
- [VEC 09] VECCHIOLLA, C.; CHU, X.; BUYYA, R. Aneka: A Software Platform for .NET-based Cloud Computing. p.267–295. **High Speed and Large Scale Scientific Computing**, IOS Press, Amsterdam, Netherlands., 2009.
- [VEL 09] VELTE, T.; VELTE, A.; ELSENPETER, R. C. **Cloud Computing, A Practical Approach**. 1. ed. EUA: Osborne/McGraw-Hill, 2009.
- [VER 09] VERMA, D. C. **Principles of Computer Systems and Network Management**. 1st. ed. Springer Publishing Company, Incorporated, 2009.
- [Wil 11] Willian Leibzon. **Willian Leibzon Nagios Addons and Plugins**. Disponível em <<http://wleibzon.bol.ucla.edu/nagios/presentations/nagios2011/presentation-nagiosconference2011.pdf>>. Acesso em: 23 de Abril de 2012.

Apêndice A

Template CentOS 5.7.

```
CONTEXT=[
  FILES="/srv/cloud/one/var/centOS/init.sh
        /opt/pcmons/booting_vms/monitoring.tar
        /opt/pcmons/booting_vms/opennebula_custom.tar",
  HOSTNAME=centos ,
  IP_PRIVATE=$NIC[IP ],
  IP_PUBLIC=$NIC[IP ],
  TARGET=sdc ,
  USERNAME=publicuser ]
CPU=1
DISK=[
  BUS=ide ,
  IMAGE_ID=92,
  TARGET=xvda ]
MEMORY=64
NAME="CentOS 5.7 - Public"
NIC=[
  NETWORK_ID=7 ]
OS=[
  BOOTLOADER=/usr/bin/pygrub ]
RAW=[
  TYPE=xen ]
TEMPLATE_ID=107
```

Apêndice B

Template Ubuntu 10.04.03 LTS.

```
CONTEXT=[
  FILES="/srv/cloud/one/var/centOS/init.sh
        /opt/pcmons/booting_vms/monitoring.tar
        /opt/pcmons/booting_vms/opennebula_custom.tar",
  HOSTNAME=ubuntu ,
  IP_PRIVATE=$NIC [ IP ] ,
  IP_PUBLIC=$NIC [ IP ] ,
  TARGET=sdc ,
  USERNAME=publicuser ]
CPU=1
DISK=[
  BUS=ide ,
  DRIVER=file : ,
  IMAGE_ID=93 ,
  TARGET=xvda1 ]
MEMORY=64
NAME="Ubuntu 10.04 - Public"
NIC=[
  NETWORK_ID=7 ]
OS=[
  INITRD=/srv/cloud/one/var/ubuntu-xen/initrd.img-2.6.32-342-ec2 ,
  KERNEL=/srv/cloud/one/var/ubuntu-xen/vmlinuz-2.6.32-342-ec2 ,
  KERNEL_CMD=ro ,
  ROOT=xvda1 ]
RAW=[
  TYPE=xen ]
TEMPLATE_ID=108
```


Apêndice C

Script para contextualização da imagem Ubuntu 10.04.03

```
#!/bin/bash
```

```
# init.sh Script #####  
# Description: Script to setup initial configuration of Ubuntu #  
# virtual machines based on OpenNebula context variables      #  
# Author: Pedro Vitti (pvitti@inf.ufsc.br)                   #  
#####
```

```
debug() {  
    echo "$(date +'[%m/%d/%y %H:%M]') [DEBUG] $@" >> /var/log/opennebula.log  
}
```

```
startMonitoring() {  
    debug "monitoring.tar founded..."  
    mkdir -p /opt/pcmons/booting_vms/  
    cp /mnt/monitoring.tar /opt/pcmons/booting_vms/  
    tar xf /opt/pcmons/booting_vms/monitoring.tar -C /opt/pcmons/booting_vms  
    /  
    python /opt/pcmons/booting_vms/Monitor.py > /dev/null 2>&1 &  
    debug "started Monitor.py..."  
    #removing unnecessary files  
    rm /root/monitoring.tar > /dev/null  
    rm /opt/pcmons/booting_vms/monitoring.tar > /dev/null
```

```

debug "removed /root/monitoring.tar"
debug "removed /opt/pcmons/booting_vms/monitoring.tar"
}

startOpenNebulaCustom() {
    mkdir -p /opt/pcmons/booting_vms/
    cp /mnt/opennebula_custom.tar /opt/pcmons/booting_vms/
    tar xf /opt/pcmons/booting_vms/opennebula_custom.tar -C /opt/pcmons/
        booting_vms/
    debug "File opennebula_custom decompressed at /opt/pcmons/booting_vms
        ..."
    debug "starting opennebula_custom script.."
    perl /opt/pcmons/booting_vms/opennebula_custom.pl
    debug "opennebula_custom script executed succesfully.."
    #removing unnecessary files
    rm /opt/pcmons/booting_vms/opennebula_custom.tar > /dev/null
    debug "removed /opt/pcmons/booting_vms/opennebula_custom.tar"
}

startVMConfiguration() {
if [ -n "$HOSTNAME" ]; then
    echo $HOSTNAME > /etc/hostname
    hostname $HOSTNAME
    echo 127.0.0.1 $HOSTNAME >> /etc/hosts
    debug "hostname configured to $hostname"
fi

if [ -n "$IP_PUBLIC" ]; then
    ifconfig eth0 $IP_PUBLIC netmask 255.255.255.0
    debug "ip configured to $ip_public"
fi

if [ -n "$NETMASK" ]; then
    ifconfig eth0 netmask $NETMASK
    debug "netmask configured to $netmask"
fi

if [ -f /mnt/$ROOT_PUBKEY ]; then
    mkdir -r /root/.ssh

```

```

    cat /mnt/$ROOT_PUBKEY >> /root/.ssh/authorized_keys
    chmod -R 600 /root/.ssh/
    debug "public key configuration done..."
fi

if [ -n "$USERNAME" ]; then
    adduser --shell /bin/bash --home /home/$USERNAME $USERNAME
    debug "user $username created..."
    if [ -f /mnt/$USER_PUBKEY ]; then
        mkdir -p /home/$USERNAME/.ssh/
        cat /mnt/$USER_PUBKEY >> /home/$USERNAME/.ssh/authorized_keys
        chown -R $USERNAME:$USERNAME /home/$USERNAME/.ssh
        chmod -R 600 /home/$USERNAME/.ssh/authorized_keys
    fi
fi
}

touch /var/log/opennebula.log

if [ -f /mnt/context.sh ]; then
    debug "context.sh founded..."
    . /mnt/context.sh
    debug "Starting VM configuration.."
    #start vm configuration
    startVMConfiguration

    #start monitoring services
    if [ -f /mnt/monitoring.tar ]; then
        startMonitoring
        debug "Starting Monitoring..."
    else
        debug "monitoring.tar NOT founded..."
    fi

    #start opennebula custom services
    if [ -f /mnt/opennebula_custom.tar ]; then
        startOpenNebulaCustom
        debug "Start OpenNebula Custom..."
    else

```

```
    debug "opennebula_custom.tar NOT founded.."
fi

else
    debug "context.sh NOT founded.."
fi
debug "init script executed succesfully.."
```

Apêndice D

Script para contextualização da imagem CentOS 5.7

```
#!/bin/bash

# init.sh Script #####
# Description: Script to setup initial configuration of CentOS #
# virtual machines based on OpenNebula context variables      #
# Author: Pedro Vitti (pvitti@inf.ufsc.br)                   #
#####

debug() {
    echo "$(date +%m/%d/%y %H:%M)' [DEBUG] $@ " >> /var/log/opennebula.log
}

startMonitoring() {
    debug "monitoring.tar founded..."
    mkdir -p /opt/pcmons/booting_vms/
    cp /mnt/monitoring.tar /opt/pcmons/booting_vms/
    tar xf /opt/pcmons/booting_vms/monitoring.tar -C /opt/pcmons/booting_vms
    /
    python /opt/pcmons/booting_vms/Monitor.py > /dev/null 2>&1 &
    debug "started Monitor.py..."
    #removing unnecessary files
    rm /root/monitoring.tar > /dev/null
    rm /opt/pcmons/booting_vms/monitoring.tar > /dev/null
    debug "removed /root/monitoring.tar"
```

```

    debug "removed /opt/pcmons/booting_vms/monitoring.tar"
}

startOpenNebulaCustom() {
    mkdir -p /opt/pcmons/booting_vms/
    cp /mnt/opennebula_custom.tar /opt/pcmons/booting_vms/
    tar xf /opt/pcmons/booting_vms/opennebula_custom.tar -C /opt/pcmons/
        booting_vms/
    debug "File opennebula_custom decompressed at /opt/pcmons/booting_vms
        ..."
    debug "starting opennebula_custom script.."
    perl /opt/pcmons/booting_vms/opennebula_custom.pl
    debug "opennebula_custom script executed succesfully.."
    #removing unnecessary files
    rm /opt/pcmons/booting_vms/opennebula_custom.tar > /dev/null
    debug "removed /opt/pcmons/booting_vms/opennebula_custom.tar"
}

startVMConfiguration() {
if [ -n "$HOSTNAME" ]; then
    sed -i "/HOSTNAME=/s/=.*$/=$HOSTNAME/" /etc/sysconfig/network
    debug "hostname configured to $(hostname)"
fi

if [ -n "$IP_PUBLIC" ]; then
    ifconfig eth0 $IP_PUBLIC
    debug "ip configured to $ip_public"
fi

ifconfig eth0 netmask 255.255.255.0

if [ -n "$NETMASK" ]; then
    ifconfig eth0 netmask $NETMASK
    debug "netmask configured to $netmask"
fi

if [ -f /mnt/$ROOT_PUBKEY ]; then
    mkdir -p /root/.ssh
    cat /mnt/$ROOT_PUBKEY >> /root/.ssh/authorized_keys

```

```

chmod -R 600 /root/.ssh/
debug "public key configuration done..."
fi

if [ -n "$USERNAME" ]; then
  useradd $USERNAME
  debug "user $username created..."
  if [ -f /mnt/$USER_PUBKEY ]; then
    mkdir -p /home/$USERNAME/.ssh/
    cat /mnt/$USER_PUBKEY >> /home/$USERNAME/.ssh/authorized_keys
    chown -R $USERNAME:$USERNAME /home/$USERNAME/.ssh
    chmod -R 600 /home/$USERNAME/.ssh/authorized_keys
  fi
fi
}

touch /var/log/opennebula.log

if [ -f /mnt/context.sh ]; then
  debug "context.sh founded..."
  . /mnt/context.sh
  debug "Starting VM configuration.."
  #start vm configuration
  startVMConfiguration

  #start monitoring services
  if [ -f /mnt/monitoring.tar ]; then
    startMonitoring
    debug "Starting Monitoring..."
  else
    debug "monitoring.tar NOT founded..."
  fi

  #start opennebula custom services
  if [ -f /mnt/opennebula_custom.tar ]; then
    startOpenNebulaCustom
    debug "Start OpenNebula Custom..."
  else
    debug "opennebula_custom.tar NOT founded.."

```

```
fi

else
    debug "context.sh NOT founded.."
fi

debug "init script executed succesfully.."
```


Apêndice E

Script para criação da base de dados e tabelas.

```
— phpMyAdmin SQL Dump
— version 3.3.2 deb1ubuntu1
— http://www.phpmyadmin.net
—
— Host: localhost
— Generation Time: Mar 01, 2012 at 04:46 PM
— Server version: 5.1.41
— PHP Version: 5.3.2-1ubuntu4.14

SET SQL_MODE="NO_AUTO_VALUE_ON_ZERO";

/*!40101 SET @OLD_CHARACTER_SET_CLIENT=@@CHARACTER_SET_CLIENT */;
/*!40101 SET @OLD_CHARACTER_SET_RESULTS=@@CHARACTER_SET_RESULTS */;
/*!40101 SET @OLD_COLLATION_CONNECTION=@@COLLATION_CONNECTION */;
/*!40101 SET NAMES utf8 */;

—
— Database: 'manager'
—
—
—
—
```

— Table structure **for** table 'vmmonitor_vm'
—

```
CREATE TABLE IF NOT EXISTS 'vmmonitor_vm' (  
  'instance_id' varchar(50) NOT NULL DEFAULT '',  
  'reservation' varchar(50) DEFAULT NULL,  
  'user' varchar(50) DEFAULT NULL,  
  'dns_name' varchar(50) DEFAULT NULL,  
  'private_dns_name' varchar(50) DEFAULT NULL,  
  'public_dns_name' varchar(50) DEFAULT NULL,  
  'state' varchar(50) DEFAULT NULL,  
  'instance_type' varchar(50) DEFAULT NULL,  
  'launch_time' varchar(50) DEFAULT NULL,  
  'availability_zone' varchar(50) DEFAULT NULL,  
  'kernel' varchar(50) DEFAULT NULL,  
  'ramdisk' varchar(50) DEFAULT NULL,  
  'last_check' varchar(50) DEFAULT NULL,  
  'node_hostname' varchar(50) DEFAULT NULL,  
  'node_ip' varchar(50) DEFAULT NULL,  
  PRIMARY KEY ('instance_id')  
) ENGINE=MyISAM DEFAULT CHARSET=latin1;
```

Apêndice F

Script para execução do PCMONS na inicialização do sistema.

pcmons-cluster

```
#!/bin/bash
#####
# Provides:          pcmons
# Required-Start:    $remote_fs $syslog
# Required-Stop:     $remote_fs $syslog
# Default-Start:     2 3 4 5
# Default-Stop:      0 1 6
# Short-Description: Checks if pcmons is running. If not, starts it.
#####

# Variables Initialization
PCMONS_PATH=/opt/pcmons/
NAGIOS_INIT_PATH=/etc/init.d/nagios3

#####
# start() function
# Description: Check if all scripts are in place
# Check if Nagios is running
# Start running_vms/Notification_Server.py
#####
start() {
    echo -e "Starting PCMONS... \n"
```

```

# Code here to start the program and check it's OK
if [ -d $PCMONS_PATH ] &&
    #Check if pcmoms directory exists
    [ -e ${PCMONS_PATH}running_vms/cluster/
      VM_Monitoring_Cluster_Plugin.py ] && #Check if pcmoms files
      exists
    [ -e ${PCMONS_PATH}interface/nagios/Nagios_Passive_Server.py ]
    && #Check if pcmoms files exists
    [ -e ${PCMONS_PATH}running_vms/Notification_Server.py ];
    #Check if pcmoms files exists
then
    nohup python $PCMONS_PATH'running_vms/cluster/
      VM_Monitoring_Cluster_Plugin.py' & # Starts pcmoms
    NAGIOS='ps -ef|grep -i "nagios" |grep -v grep |awk '{print $2
      }'
    if [ ! -z "$NAGIOS" ]; then
        nohup python $PCMONS_PATH'running_vms/
          Notification_Server.py' &
    else
        echo -e "Nagios is not running, PCMONS won't work
          properly \n Exiting"
    fi
    else
        echo -e "Required files not found. Verify the installation
          .\n"
        exit 0
    fi
    echo "Done"
    return 0
}

#####
# stop() function
# Description: Get the PID of each
#               service and kill them
#####
stop() {
    echo -e "Stopping PCMONS...\n "

```

```

VM_MONITORING_CLUSTER_PLUGIN='ps -ef | grep -i "
    vm_monitoring_cluster_plugin" | grep -v grep lawk '{print $2}''
NOTIFICATION_SERVER='ps -ef | grep -i "notification_server.py" | grep
    -v grep lawk '{print $2}''

if [ ! -z $NOTIFICATION_SERVER ]; then
    kill $NOTIFICATION_SERVER
fi

if [ ! -z $VM_MONITORING_CLUSTER_PLUGIN ]; then
    kill $VM_MONITORING_CLUSTER_PLUGIN
fi

echo "OK"
return 0
}

#####
# status() function
# Description: Check the status of pcmons
#####
status() {
    VM_MONITORING_CLUSTER_PLUGIN='ps -ef | grep -i "
        vm_monitoring_cluster_plugin" | grep -v grep lawk '{print $2}''
    NOTIFICATION_SERVER='ps -ef | grep -i "notification_server.py" | grep
        -v grep lawk '{print $2}''

    if [ ! -z $VM_MONITORING_CLUSTER_PLUGIN ]; then
        if [ ! -z $NOTIFICATION_SERVER ]; then
            echo "All running"
        else
            echo -e "NOTIFICATION_SERVER not running\n"
        fi
    else
        echo -e "VM_MONITORING_CLUSTER_PLUGIN not running\n"
    fi
    return 0
}

case "$1" in

```

```
start)  start    ;;
stop)   stop     ;;
status) status   ;;
restart) stop start ;;
*)
    echo $"Usage: $0 {start|stop|status|restart}"
    exit 1
esac
```

Apêndice G

Script para execução do PCMONS na inicialização do sistema.

pcmons-node

```
#!/bin/bash
#####
# Provides:          pcmons
# Required-Start:    $remote_fs $syslog
# Required-Stop:     $remote_fs $syslog
# Default-Start:     2 3 4 5
# Default-Stop:      0 1 6
# Short-Description: Checks if pcmons is running. If not, starts it.
#####

# Variables Initialization
PCMONS_PATH=/opt/pcmons/
NAGIOS_INIT_PATH=/etc/init.d/nagios

#####
# start() function
# Description: Check all scripts are in place
# Start running_vms/node/VM_Monitoring_Node_Plugin.py
#####
start() {
    echo -e "Starting PCMONS... \n"

    # Code here to start the program and check it's OK
```

```

if [ -d $PCMONS_PATH ] &&
    #Check if pcmons directory exists
    [ -e ${PCMONS_PATH}running_vms/node/VM_Monitoring_Node_Plugin.py
    ]; #Check if pcmons files exists

then
    python $PCMONS_PATH'running_vms/node/VM_Monitoring_Node_Plugin.
    py' & # Starts pcmons Node
else
    echo -e "Required files not found. Verify the installation
    .\n"
    exit 0
fi
echo "Done"
return 0

}

#####
# stop() function
# Description: Get the PID of each
# service and kill then
#####
stop() {
    echo -e "Stopping PCMONS...\n "

    VM_MONITORING_NODE_PLUGIN='ps -ef|grep -i "
    vm_monitoring_node_plugin" |grep -v grep |awk '{print $2}'

    if [ ! -z $VM_MONITORING_NODE_PLUGIN ]; then
        kill $VM_MONITORING_NODE_PLUGIN
    fi

    echo "OK"
    return 0
}

#####
# status() function
# Description: Check the status of pcmons
#####

```



```
status() {

    VM_MONITORING_NODE_PLUGIN='ps -ef | grep -i "
        vm_monitoring_node_plugin" | grep -v grep | awk '{print $2}'

    if [ ! -z $VM_MONITORING_NODE_PLUGIN ]; then
        echo "Node Plugin running.."
    else
        echo -e "VM_MONITORING_NODE_PLUGIN is not running\n"
    fi
    return 0
}

case "$1" in
    start)    start        ;;
    stop)     stop         ;;
    status)   status       ;;
    restart)  stop start   ;;
    *)
        echo $"Usage: $0 {start|stop|status|restart}"
        exit 1
esac
```

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

Pedro Vitti¹

¹Instituto de Informática e Estatística – Universidade Federal de Santa Catarina (UFSC)
Trindade – Florianópolis – SC – Brazil

`pvitti@inf.ufsc.br`

Abstract. *Considering the lack of generic and open-source solutions for management and monitoring of private clouds, PCMONS was developed. Intended to be an extensible and modular monitoring system for private clouds, primarily operates in retrieving, gathering and preparing relevant information for monitoring data visualization and is specially focused on virtual machines. This project aims to increase the compatibility of this tool by developing an extension to one of the currently leading IaaS tool for cloud computing: OpenNebula.*

Resumo. *Considerando a falta de soluções genéricas para o gerenciamento e monitoramento de nuvens privadas foi desenvolvido o PCMONS. Destinado a ser um sistema de monitoramento extensível e modular para nuvens privadas, atua principalmente na recuperação, coleta e preparação de informação relevante para a visualização dos dados de monitoramento e é especialmente focado em máquinas virtuais. Esse projeto visa ampliar a compatibilidade desta ferramenta, através do desenvolvimento de uma extensão para integração a umas das principais ferramentas para infraestrutura em computação em nuvem atualmente: o OpenNebula.*

1. Introdução

Com o avanço da sociedade humana, serviços básicos e essenciais de utilidade pública como eletricidade, água e telefone, tornaram-se fundamentais para a vida diária. As infraestruturas existentes atualmente permitem a entrega desses serviços de forma transparente ao usuário por meio de um modelo de pagamento baseado no uso [Vecchiolla et al. 2009]. O uso desses serviços é cobrado de acordo com uma política de tarifação aos usuários. A computação em nuvem utiliza essa mesma ideia, aplicada na computação. O termo "nuvem" é uma metáfora em relação à forma como a Internet é comumente representada nos diagramas de rede. Nesses diagramas, as nuvens representam todas as tecnologias que fazem a Internet funcionar, abstraindo a infraestrutura e complexidade [Velte et al. 2009]. Basicamente, a computação em nuvem pode ser entendida como recursos de computação altamente escaláveis fornecidos como um serviço externo através da rede. A cobrança por esses serviços é feita de acordo com a sua utilização, ou seja, sob demanda, assim, o usuário que utiliza um serviço na nuvem paga somente por aquilo que utilizar.

Os sistemas baseados em computação em nuvem se tornam mais comuns a cada dia. Muitas empresas estão migrando totalmente ou parcialmente seus recursos de computação e ou seus serviços e aplicações para ambientes na nuvem. Diversos estudos comparativos estão sendo feitos em relação às vantagens e desvantagens que estas soluções podem trazer.

Muitas das tecnologias envolvidas e que possibilitam a viabilização do paradigma da computação em nuvem já são bastante consolidadas, como a computação em grade (*grid computing*), computação distribuída, acordos de nível de serviço e virtualização. Porém, outras ainda precisam ser arquitetadas ou estendidas de modo a prover seu melhor uso e gerenciamento, como é o caso dos sistemas de monitoramento, instrumentos fundamentais para se acompanhar o comportamento de um sistema, detectar problemas e prover os dados necessários para atividades de manutenção, suporte e planejamento [Chaves 2010].

1.1. Motivação e Objetivos

A necessidade de sistemas de monitoramento em ambiente de computação em nuvem é cada vez maior. Com monitoramento é possível assistir diferentes atividades no sistema, e assim se utilizar de registros e outras informações para acompanhar sua infraestrutura, avaliando seu desempenho, comparando sua eficiência e com isso, encontrar maneiras de aperfeiçoar um sistema específico ou um determinado serviço.

Esse trabalho tem como objetivo ampliar a compatibilidade da ferramenta PCMONS proposta e desenvolvida por [Chaves 2010] com o desenvolvimento de uma extensão para integrá-lo à ferramenta de Infraestrutura como Serviço (IaaS) OpenNebula e implantar uma nuvem privada com propósitos acadêmicos no Laboratório para testar o sistema desenvolvido através de um estudo de caso.

1.2. Organização do Trabalho

A seção 1 apresenta introdução e contextualização ao tema. São apresentados também: a motivação, objetivo geral e os objetivos específicos do trabalho. A seção 2 apresenta algumas formas de categorizações de computação em nuvem, como em relação ao seu modelo de serviço, e modelo de implantação. É mostrada a ferramenta para implantação de computação em nuvem na modalidade IaaS: OpenNebula. Na seção 3 são apresentadas características de monitoramento para computação em nuvem e a necessidade e importância do monitoramento. A ferramenta PCMONS, empregada para o monitoramento de nuvens privadas é apresentada. A seção 4, apresenta os passos para a implantação de uma nuvem privada, utilizando a ferramenta OpenNebula no Laboratório de Redes e Gerência, para fins de estudo e conhecimento prático. São demonstradas as etapas utilizadas para o desenvolvimento de uma adaptação da ferramenta PCMONS para torná-la compatível com o OpenNebula. A implementação é testada, utilizando esse mesmo ambiente em nuvem. A seção 5 encerra o trabalho com algumas conclusões e considerações finais e são apresentadas algumas perspectivas para trabalhos futuros.

2. Computação em Nuvem

O paradigma de computação em nuvem, por estar envolvido com diversas outras tecnologias computacionais, muitas vezes tem a sua definição incorporada pelas definições já consolidadas dessas tecnologias, como a computação em grade, a computação distribuída, a virtualização, entre outras. Assim, uma definição padrão aceita universalmente para a computação em nuvem se torna difícil. O [NIST SP 800-145 2011], National Institute Standards and Technology, do governo dos Estados Unidos define computação em nuvem como um modelo que possibilita acesso conveniente e sob demanda, através da rede, a um conjunto compartilhado de recursos computacionais configuráveis (rede, servidores,

armazenamento, aplicações e serviços). Esses recursos podem ser providos rapidamente e liberados com um mínimo de esforço de gerenciamento ou interação com o provedor do serviço.

2.1. Classificação em Computação em Nuvem

Os ambientes de computação em nuvem podem ser classificados seguindo mais de um critério. Duas das possíveis classificações para esses ambientes são em relação ao seu modelo de serviços e ao seu modelo de implantação.

Em relação ao modelo de serviços, as nuvens podem ser classificadas em Software como Serviço (SaaS), Plataforma como Serviço (PaaS) e Infraestrutura como Serviço (IaaS). SaaS é um modelo de distribuição de *software* em que os aplicativos são hospedados por um provedor ou um fornecedor de serviço e disponibilizados aos clientes, através de uma rede, geralmente a Internet. No modelo de PaaS um provedor oferece além de uma infraestrutura, um conjunto de soluções e ferramentas que um desenvolvedor necessita para criar uma aplicação. Esse modelo oferece a capacidade de gerenciamento de todas as fases do desenvolvimento de uma aplicação, desde a modelagem e o planejamento, até a construção, implantação para testes e manutenção. Esse modelo é uma consequência direta do modelo de Software como Serviço. Já na modalidade IaaS é oferecido ao cliente uma infraestrutura de hardware completa. Armazenamento, hardware (memória, processamento, etc.), servidores e componentes de rede são oferecidos aos usuários. O consumidor é capaz de implantar e executar aplicativos arbitrários, que podem incluir sistemas operacionais completos e tecnologias de virtualização para o gerenciamento dos recursos [Hurwitz et al. 2010]. O cliente não administra ou controla a infraestrutura da nuvem, porém tem controle sobre o sistemas operacionais, armazenamento, uso de aplicativos e controle dos componentes de rede.

Outra categorização é em relação ao seu modelo de implantação, ou seja, sua abrangência de público. As nuvens podem ser públicas, privada, híbrida e comunitária. A nuvem pública descreve o significado convencional da computação em nuvem, onde um prestador de serviços disponibiliza recursos computacionais, tais como aplicativos e armazenamento para o público em geral, através da Internet. Serviços de nuvem pública podem ser livres ou oferecidos utilizando um modelo de pagamento baseado no uso *pay-per-use* [Smoot and Tan 2012]. Em uma nuvem privada, uma infraestrutura é disponibilizada para uso exclusivo de uma única organização que compreende vários consumidores. Pode ser de propriedade, gerenciados e operados pela organização, um terceiro, ou uma combinação deles, e podem existir dentro ou fora das instalações da empresa [NIST SP 800-145 2011]. Uma nuvem híbrida acontece quando recursos computacionais de nuvens públicas e de nuvens privadas são utilizados. Uma empresa pode optar por usar um serviço de nuvem pública para a utilização de recursos em geral, porém, pode armazenar suas informações críticas de negócios em uma nuvem privada, dentro do seu próprio domínio com a finalidade de aumentar a segurança dos dados. No modelo de nuvem comunitária a infraestrutura da nuvem é compartilhada por diversas organizações, dando suporte a uma comunidade específica, com preocupações ou atividades em comum, podendo ser gerenciada pela própria organização ou por terceiros e se localizar dentro ou fora dos limites da organização.

2.2. OpenNebula

O OpenNebula é um conjunto de ferramentas de código aberto para implantação de computação em nuvem na modalidade IaaS, oferecendo diversos recursos e soluções que facilitam e flexibilizam o gerenciamento completo de infraestruturas virtualizadas. Pode ser utilizado para o desenvolvimento de nuvens privadas, públicas, comunitárias e híbridas, possuindo a capacidade de combinar uma infraestrutura local com uma infraestrutura baseada em nuvem pública, permitindo ambientes altamente escaláveis. Oferece uma interface compatível com as interfaces EC2 Query, OGF OCCI e vCloud, largamente utilizadas e consideradas como padrões quando se trata de serviços para a nuvem. Inclui recursos para integração, gerenciamento, escalabilidade, segurança e contabilidade. Enfatiza a interoperabilidade, padronização e portabilidade, fornecendo aos usuários e administradores da nuvem a possibilidade de escolha entre diversas interfaces e hipervisores (Xen, KVM e VMware ESX). Possui também uma arquitetura flexível que pode acomodar hardwares múltiplos e combinações diferentes de softwares. A ferramenta pode ser instalada em qualquer distribuição GNU Linux a partir do seu código fonte ou através dos repositórios oficiais de algumas distribuições específicas (Debian, openSUSE, Ubuntu). Muitos de seus recursos foram desenvolvidos para atender aos requisitos de casos de uso de negócio de empresas líderes de diversos setores em computação em nuvem, tais como Reservoir, StratusLab, BonFIRE, or 4CaaS. É considerada referência para a computação em nuvem em pesquisas de vários grandes projetos de infraestrutura. No momento da realização deste trabalho, o OpenNebula encontra-se na versão 3.2.1.

A arquitetura interna do OpenNebula engloba vários componentes especializados em diferentes aspectos para o gerenciamento da infraestrutura virtual e pode ser dividida em três camadas: Core (núcleo), Escalonador e outras ferramentas e Drivers.

O núcleo é responsável por controlar o ciclo de vida das máquinas virtuais, e possui três diferentes áreas de gerenciamento, são elas:

- Tecnologias de imagens e armazenamento, para a preparação de imagens de disco para as máquinas virtuais;
- Tecnologias de rede, para o provimento de ambientes de rede para as máquinas virtuais;
- Hipervisores, para a criação e gerenciamento das máquinas virtuais.

O núcleo realiza as operações específicas para armazenamento, rede e virtualização, por meio de *drivers* conectáveis. Assim, o OpenNebula não está vinculado a qualquer ambiente específico, fornecendo uma camada de gerenciamento uniforme, independentemente da infraestrutura utilizada. Além de gerenciar as máquinas virtuais como uma unidade, o núcleo também lida com a configuração e entrega de informações de contexto (como endereço IP, *hostname*, certificados digitais, licenças de software) para cada uma delas [Cordeiro et al. 2010].

A segunda camada contém diversas ferramentas fornecidas com o OpenNebula, como uma interface de linha de comando, o escalonador, e também ferramentas de terceiros que podem ser facilmente criadas utilizando as interfaces XML-RPC ou a API do OpenNebula. Um componente escalonador separado toma as decisões de alocação das máquinas virtuais. Mais especificamente, o escalonador tem acesso às informações sobre todas as requisições que o OpenNebula recebe e, com base nessas requisições, acompanha

as alocações atuais e futuras, criando e atualizando o escalonador de recursos e enviando os comandos de implantação adequados ao núcleo do OpenNebula. O escalonador padrão do OpenNebula fornece uma política de escalonamento que coloca as máquinas virtuais em recursos físicos de acordo com um algoritmo de classificação que o administrador pode configurar. Ele se baseia em dados em tempo real de ambas as máquinas virtuais em execução e os recursos físicos disponíveis [Sotomayor et al. 2009]. O OpenNebula oferece nessa camada interfaces de gerenciamento para integrar as funcionalidades do núcleo dentro de outras ferramentas de gerenciamento do *datacenter*, como contabilidade e *arcabouços* para monitoramento. Para este fim, OpenNebula implementa a API libvirt (<http://libvirt.org>), uma interface aberta para o gerenciamento de máquinas virtuais, e também uma interface de linha de comando. Estas funcionalidades são expostas aos usuários externos através de uma interface para nuvem.

Na camada de drivers o OpenNebula fornece um conjunto de módulos conectáveis para interagir com *middlewares* específicos (por exemplo, hipervisores de virtualização, mecanismos de transferência de arquivos ou serviços de informação). Estes módulos são chamados de Drivers de Acesso Middleware. O suporte para nuvem híbridas é feito utilizando *Drivers* para comunicação com nuvens externas. Isso permite que as organizações complementem sua infraestrutura local, com a capacidade de computação de uma nuvem pública para atender às demandas de pico, atendendo melhor os pedidos de acesso (por exemplo, movendo o serviço para locais mais próximos do usuário), ou implementando estratégias de alta disponibilidade [Milojić andić and et al. 2011].

O OpenNebula foi projetado para ser modular, a fim de permitir a sua integração com o maior número possível de hipervisores e diferentes ambientes e arquiteturas. [Cordeiro et al. 2010].

3. Monitoramento em Computação em Nuvem

Monitoramento é o processo de obter informações sobre elementos de um sistema computacional. Estas informações ajudam a entender a situação do sistema, sua configuração, estatísticas de uso e desempenho, informações sobre erros e sobre a topologia do sistema. O processo de monitoramento depende de técnicas para coletar, processar, armazenar e disponibilizar estas informações. A variedade de elementos que compõem um sistema computacional exige técnicas adequadas para cada classe de elemento [Verma 2009].

Ambientes de computação em nuvem, especialmente os que atuam no modelo de IaaS, fornecem recursos computacionais a seus clientes utilizando a tecnologia de máquinas virtuais. Estas podem ser definidas como uma implementação de software de um ambiente computacional em que um sistema operacional ou programa pode ser instalado e executado. A máquina virtual emula um ambiente de computação física. Desse modo, tal como um ambiente de computação tradicional que emprega máquinas físicas, necessita de um monitoramento do sistema utilizando métricas tais como: processamento, memória, disco rígido, rede, entre outros, um ambiente virtual utilizando máquinas virtuais também deve ser monitorado. Na maioria das vezes ambientes virtualizados possuem suas próprias ferramentas de monitoramento que são dependentes de plataforma como o XenCenter para o Citrix XenServer e o VMware vCenter para o VMware vSphere [Huang et al. 2009]. De acordo com [Shao et al. 2009] sistemas genéricos existentes de monitoramento de desempenho de máquinas virtuais são inadequados, com interfaces de

gerenciamento fracas e sistemas para coleta de informações incompreensíveis. Desse modo, fica difícil para administradores encontrarem pontos de estrangulamento do sistema através dos dados obtidos, e portanto, se torna complicado o gerenciamento dos recursos de forma a melhorar o desempenho de todo o sistema.

Para companhias e gerentes de TI não importa se a organização mantém uma infraestrutura de TI tradicional ou se possui uma infraestrutura virtual baseada em um ambiente de computação em nuvem. O monitoramento em busca de informações, dados e estatísticas é crucial. É necessária uma estratégia de monitoramento adaptada a recursos de computação em nuvem, aplicações e infraestrutura.

Em [Chaves 2010] é proposto um modelo de arquitetura genérica para monitoramento de nuvens privadas dividido em três camadas conforme ilustrado na Figura 1.

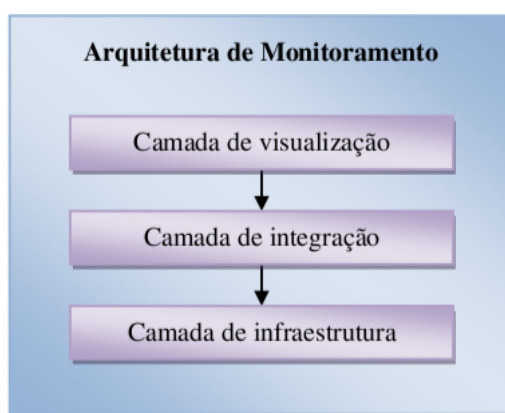


Figura 1. Arquitetura de três camadas para monitoramento de nuvens privadas [Chaves 2010].

Camada de infraestrutura - É a camada mais baixa da arquitetura e a base do modelo. Consiste em toda a infraestrutura utilizada para criar o ambiente de nuvem. Pode-se citar como pertencentes a essa camada toda a infraestrutura de hardware, tais como: dispositivos de armazenamento, processamento, rede, entre outros, e de software necessárias, tais como: sistemas operacionais, aplicativos diversos, licenças, hipervisores, entre outros [Chaves 2010].

Camada de Integração - A camada de infraestrutura é composta, principalmente, de recursos heterogêneos e, portanto, requer uma interface comum para acesso. Uma situação típica é um usuário que solicita a instanciação de uma máquina virtual. Este tipo de solicitação pode ser controlada por diferentes hipervisores, tais como Xen, KVM, VMWare, dependendo de como a camada de infraestrutura é implantada. Outro cenário possível é a presença de múltiplas plataformas de computação em nuvem, como Eucalyptus e OpenNebula. Assim, ações de controle a serem executadas na camada de infraestrutura devem ser sistematizadas antes de serem passadas para o serviço apropriado. Portanto, a camada de integração é responsável por abstrair quaisquer detalhes da infraestrutura [De Chaves et al. 2011].

Camada de Visualização - A camada de visualização serve de interface de gerenciamento de alto nível, através da qual é possível verificar informações como o cumprimento de políticas organizacionais e SLAs [De Chaves et al. 2011]. Usuários dessa camada estão interessados em averiguar máquinas virtuais disponíveis ou obter informações de monitoramento de máquinas virtuais em execução.

Citando [Chaves 2010], essa arquitetura em três camadas foi projetada para atender às necessidades de monitoramento de uma nuvem privada. Entre essas necessidades pode-se citar a integração aos sistemas de monitoramento já utilizados na empresa, como o Nagios, que além de ser extensível para atender a novos recursos, é simples de operar. Para validação da arquitetura proposta, a autora propõe o desenvolvimento de uma solução intitulada PCMONS (Private Cloud Monitoring Systems), descrita na seção 3.1.

3.1. PCMONS

O PCMONS, sigla para (*Private Cloud Monitoring System*), ou em português, Sistema de Monitoramento de Nuvens Privadas é uma ferramenta desenvolvida por [Chaves 2010] para o monitoramento de ambientes de nuvens privadas. Implementado baseando-se na arquitetura genérica para o monitoramento de ambientes de nuvens privadas descrita na seção 3, o PCMONS age principalmente na camada de integração, em atividades como coleta e preparo das informações relevantes para a camada de visualização. De acordo com [De Chaves et al. 2011], o sistema está dividido em vários módulos, com o propósito de simplificar futuras adaptações para ferramentas específicas e facilitando o seu estudo e aplicação:

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 2 mostra os módulos da ferramenta descritos e a interação entre eles.

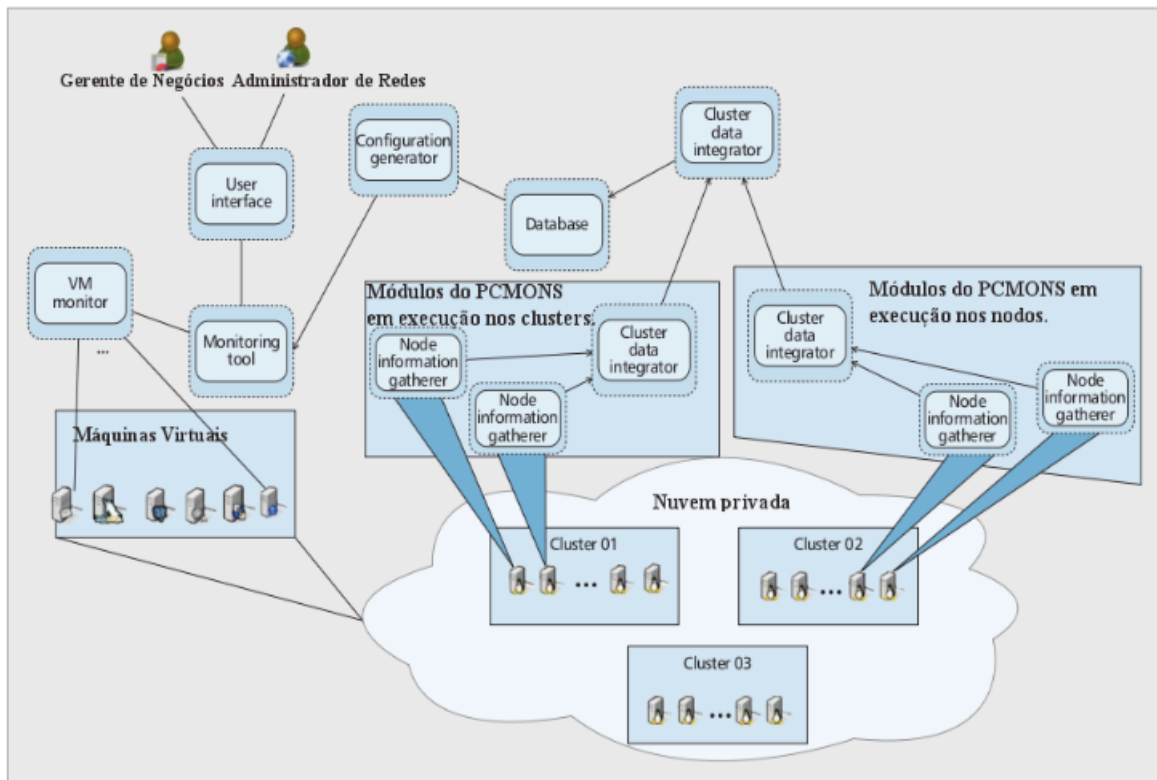


Figura 2. Arquitetura da ferramenta PCMONS e interação entre seus componentes. Adaptado de [De Chaves et al. 2011].

No atual estágio de desenvolvimento, o PCMONS é compatível com a ferramenta *Eucalyptus*, na camada de infraestrutura e o Nagios, na camada de visualização. A ênfase do monitoramento é dada as máquinas virtuais inicialmente, devido à falta de soluções nessa área. De acordo com [Chaves 2010] seu desenvolvimento considera a facilidade de integração com outras ferramentas para computação em nuvem, através do desenvolvimento de extensões. Isto é, não há um forte acoplamento ao Eucalyptus ou ao Nagios, os quais são um módulo do sistema, utilizados apenas quando configurados.

4. Desenvolvimento

O PCMONS, em sua versão atual, age principalmente na camada de integração, em atividades como coleta e preparo das informações relevantes para a camada de visualização. Para provar a facilidade de integração da ferramenta a outras soluções que provêm computação em nuvem na modalidade IaaS, foi implementada uma adaptação para aumentar sua compatibilidade e permitir sua utilização juntamente com a ferramenta OpenNebula.

Atualmente existem diversas soluções para a implantação de ambientes de computação em nuvem. Além da ferramenta OpenNebula descrita na seção 2.2 pode-se citar as ferramentas de código aberto Eucalyptus, OpenStack, Xen Cloud Platform, Nimbus.

Como motivação para escolha da ferramenta OpenNebula no desenvolvimento desse trabalho, foi feita uma avaliação das funcionalidades das principais ferramentas de código aberto para provimento de computação em nuvem na modalidade IaaS. Essa análise possibilitou a elaboração de uma lista, que segue abaixo, contendo as principais

vantagens encontradas em relação a outras soluções.

- Possui uma interface de administração superior onde pode-se executar diversas ações sobre as máquinas virtuais como migrar, suspender e reiniciar. Outras ferramentas permitem apenas a utilização das funcionalidades oferecidas pela interface EC2.
- Melhores políticas para alocações de recursos. Seu algoritmo pode ser ajustado para permitir capacidades de reservas avançadas. Existem algumas pesquisas para o desenvolvimento de escalonadores utilizando os princípios de tecnologia verde, voltada para o impacto dos recursos tecnológicos no meio ambiente [Sotomayor et al. 2009].
- Permite o gerenciamento completo dos serviços, incluindo redes privadas para interconectar máquinas de serviço. Como o Eucalyptus está ligado à interface EC2, não é possível definir redes virtuais [Sotomayor et al. 2009].
- Possui suporte para contextualização antecipada. Com isso é possível incluir dados de contexto, para que uma máquina virtual se auto-configure durante sua inicialização, utilizando esses dados (IP, hostname, licenças de software, chaves SSH).
- Possui uma API para permitir a extensão de seus recursos. Tanto para desenvolvimento de aplicações sobre a plataforma, como para integrar suas tecnologias de armazenamento, virtualização e rede [Cordeiro et al. 2010].
- Pode ser utilizado para a implantação de nuvens híbridas. Com isso é possível aumentar a capacidade de uma nuvem privada, instanciando máquinas virtuais tanto localmente como em uma nuvem pública.
- Possui uma arquitetura completamente modular, facilitando sua integração com diferentes tecnologias [Cordeiro et al. 2010].
- A versão de código aberto do Eucalyptus não vem recebendo suporte adequado por parte dos desenvolvedores e não recebe atualizações há quase um ano.

Desde 2010, com o lançamento da versão 2.0 do OpenNebula, a cada dois meses em média, uma nova versão da ferramenta é lançada pelos desenvolvedores, com diversas melhorias e novas funcionalidades. Esse grande esforço por parte dos desenvolvedores, aliado a uma comunidade adepta e uma vasta documentação, tornam o OpenNebula um solução completa e adequada para pesquisas, estudos e desenvolvimentos em geral, na área de computação em nuvem.

Nas próximas seções são apresentadas as modificações realizadas nos módulos da ferramenta PCMONS. Todos esses módulos são descritos em detalhes na seção 3.1.

4.1. Implementação

O funcionamento básico do PCMONS ocorre da seguinte maneira: o sistema reúne diversas informações a respeito das máquinas virtuais em cada um dos nós da nuvem, como IPs, usuários que as instanciaram, e números de identificação das mesmas e os envia para a camada de visualização que tem como responsabilidade criar os arquivos necessários para o monitoramento destas máquinas virtuais.

A) Cluster Integrador de Dados - No arquivo de configuração desse módulo foram feitas algumas alterações para facilitar a integração com outras ferramentas de IaaS. Uma nova variável denominada INFRA foi criada e é responsável por guardar o nome

da ferramenta de infraestrutura sendo utilizada. Para que todos os nós que estão sendo utilizados para armazenamento de máquinas virtuais pelo OpenNebula sejam mapeados corretamente, esse módulo fica responsável por examinar a variável INFRA definida pelo usuário, e executar o trecho de código adequado, correspondente a ferramenta definida. Nesse mesmo módulo foi criado um método responsável pela atualização das informações das máquinas virtuais que estão sendo gerenciadas. As classes responsáveis pelo acesso ao banco de dados para armazenamento das informações das máquinas virtuais não sofreram modificações, já que essas informações e o acesso a base de dados para inserções e atualizações se mantiveram iguais para ambas as ferramentas de IaaS.

B) Coletor de Informações do Nó - Foram feitas algumas adaptações neste módulo, que é o responsável por coletar as informações locais em cada um dos nós da nuvem e reunir informações sobre as máquinas virtuais locais e enviar para o *Data Integrator Cluster* para adaptá-lo a ferramenta OpenNebula.

C) Interface Com Usuário - No módulo interface, responsável pela organização e apresentação das informações coletadas alguns trechos de código foram alterados para adaptar o PCMONS a última versão estável da ferramenta de monitoramento Nagios, a 3.2.0.

4.2. Estudo de Caso

Para testar a implementação da ferramenta PCMONS compatível com o OpenNebula, foi realizado um estudo de caso, onde foi implantado um ambiente de nuvem privada utilizando esta ferramenta. Para exemplificar uma situação prática para uso da estratégia de monitoramento e ferramenta desenvolvida, foram disponibilizadas na nuvem imagens de máquinas virtuais, para que um usuário possa instanciar simulando assim um ambiente de provedor de hospedagem.

A tabela 1 abaixo mostra a infraestrutura de software e hardware utilizadas nessa implantação. Uma das máquinas foi empregada como *front-end*, responsável por controlar a nuvem, enquanto a outra foi responsável pela execução do hipervisor e pela alocação das máquinas virtuais.

Tabela 1. Descrição do ambiente utilizado para o estudo de caso.

Hardware	Software	Papel no sistema
AMD Phenom(tm) 9650 Quad-Core, 2310 MHZ 4GB DDR2 667MHz HD SAMSUNG HD753LJ 750GB ATA-8-AC	Ubuntu 10.04.4 LTS OpenNebula 3.2 Nagios Core 3.2.0	OpenNebula <i>Front-end</i>
Intel(R) Core(TM) 2 Quad Q8200, 2.33 GHz 3GB DDR2 333MHz RAM HD SAMSUNG HD322HJ 320GB ATA	CentOS 5.8 (Final) Xen 3.1.2	Nó

Para esse estudo de caso foram criadas duas imagens com a instalação mínima dos sistemas operacionais baseados em Linux Ubuntu 10.04.03 LTS e CentOS 5.7. Para

permitir a personalização e instalação de aplicativos nas imagens, foi utilizado o mecanismo de contextualização implementado pelo próprio OpenNebula. Esse mecanismo permite passar arquivos e parâmetros de configuração para uma máquina virtual durante sua instanciação utilizando uma imagem ISO. No *template* da máquina virtual pode-se especificar o conteúdo do arquivo ISO (arquivos e diretórios), e informar em qual dispositivo a imagem ISO estará acessível e especificar os parâmetros de configuração que serão gravados em um arquivo para uso posterior dentro da máquina virtual. Uma única modificação nas máquinas virtuais foi feita para a utilização desse mecanismo. Foi adicionado um *script* na inicialização do sistema operacional da máquina virtual que tem a função de montar a partição ISO e executar o *script* de configuração. Esse uso de chamada indireta, onde um *script* chama o outro facilita bastante a instalação e adaptação das máquinas virtuais, já que a modificação do *script* de configuração pode ser feita localmente antes da instanciação da máquina virtual, ao invés da necessidade de alterar todas as máquinas virtuais.

As métricas de monitoramento utilizadas nesse estudo de caso provêm um conjunto mínimo de informações sobre os recursos monitorados e englobam dois conjuntos de monitoramento: um referente ao desempenho/disponibilidade da máquina virtual (carga, memória, ping e SSH) e outro em relação ao serviço que está sendo executado (número de conexões HTTP). A figura 3 mostra todos os serviços, ativos (PING e SSH) e passivos (HTTP_CONNECTIONS, LOAD, RAM) que estão sendo monitorados pelo PCMONS.

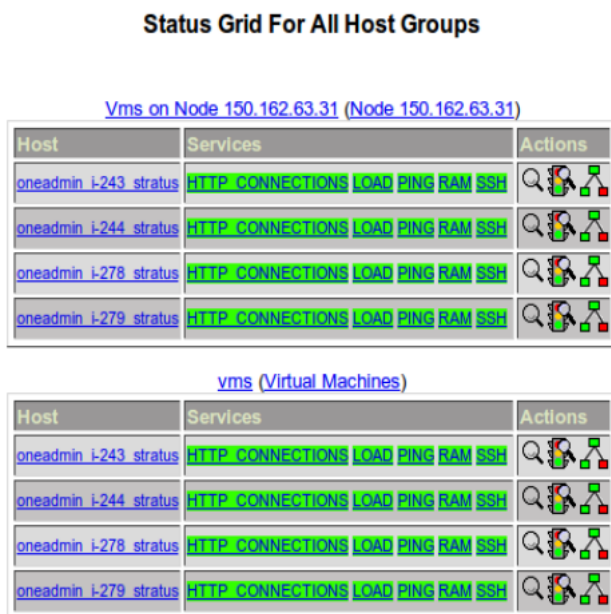


Figura 3. PCMONS: Serviços sendo monitorados no momento.

Na figura 4 é possível visualizar detalhadamente o *status* de todas as métricas que estão sendo monitoradas. A figura 5 mostra a organização das máquinas virtuais num *hostgroup* próprio e a organização das máquinas virtuais por nós onde estão sendo executadas. Nota-se, que, para esse estudo de caso, todas as máquinas virtuais estão sendo executadas no mesmo nó. Esse mapeamento que informa quais máquinas virtuais estão hospedadas em cada nó é uma característica importante que facilita a detecção rápida de

falhas e sua consequente correção.





















oneadmin_i-243_stratus	HTTP CONNECTIONS	 OK	2012-05-12 22:19:31	0d 0h 1m 32s	1/4	0
	LOAD	 OK	2012-05-12 22:20:13	0d 0h 0m 50s	1/4	OK - load average: 0.00, 0.00, 0.00
	PING	 OK	2012-05-12 22:17:07	0d 0h 53m 56s	1/4	PING OK - Packet loss = 0%, RTA = 0.17 ms
	RAM	 OK	2012-05-12 22:20:41	0d 0h 0m 22s	1/4	64 81/83
	SSH	 OK	2012-05-12 22:19:44	0d 0h 51m 19s	1/4	SSH OK - OpenSSH_5.3p1 Debian-3ubuntu7 (protocol 2.0)
oneadmin_i-244_stratus	HTTP CONNECTIONS	 OK	2012-05-12 22:19:43	0d 0h 1m 20s	1/4	0
	LOAD	 OK	2012-05-12 22:20:07	0d 0h 0m 56s	1/4	OK - load average: 0.00, 0.00, 0.00
	PING	 OK	2012-05-12 22:20:47	0d 0h 50m 16s	1/4	PING OK - Packet loss = 0%, RTA = 2.16 ms
	RAM	 OK	2012-05-12 22:20:50	0d 0h 1m 4s	1/4	94 70/83
	SSH	 OK	2012-05-12 22:20:51	0d 0h 50m 12s	1/4	SSH OK - OpenSSH_5.3p1 Debian-3ubuntu7 (protocol 2.0)
oneadmin_i-278_stratus	HTTP CONNECTIONS	 OK	2012-05-12 22:19:41	0d 1h 17m 51s	1/4	0
	LOAD	 OK	2012-05-12 22:19:41	0d 1h 17m 51s	1/4	OK - load average: 0.00, 0.00, 0.00
	PING	 OK	2012-05-12 22:19:46	0d 0h 4m 39s	1/4	PING OK - Packet loss = 0%, RTA = 0.20 ms
	RAM	 OK	2012-05-12 22:19:41	0d 1h 17m 51s	1/4	97 81/83
	SSH	 OK	2012-05-12 22:16:21	0d 0h 4m 42s	1/4	SSH OK - OpenSSH_5.3p1 Debian-3ubuntu7 (protocol 2.0)
oneadmin_i-279_stratus	HTTP CONNECTIONS	 OK	2012-05-12 22:19:52	0d 1h 16m 45s	1/4	0
	LOAD	 OK	2012-05-12 22:19:52	0d 1h 16m 45s	1/4	OK - load average: 0.01, 0.06, 0.02
	PING	 OK	2012-05-12 22:18:43	0d 0h 2m 20s	1/4	PING OK - Packet loss = 0%, RTA = 0.19 ms
	RAM	 OK	2012-05-12 22:19:52	0d 1h 16m 45s	1/4	69 44/64
	SSH	 OK	2012-05-12 22:20:01	0d 0h 6m 2s	1/4	SSH OK - OpenSSH_4.3 (protocol 2.0)

Figura 4. PCMONS: Status dos serviços monitorados em cada uma das máquinas virtuais.

5. Considerações Finais e Trabalhos Futuros

A viabilidade de otimização de recursos computacionais faz com que sistemas baseados em computação em nuvem se tornem mais comuns a cada dia. Muitas empresas estão migrando, totalmente ou parcialmente, seus recursos físicos de computação e seus serviços e aplicações para infraestruturas virtuais. Com isso, administradores e gerentes de rede possuem sob sua responsabilidade quantidade cada vez maior de máquinas e serviços. Neste cenário, ferramentas que auxiliem e automatizem tarefas de monitoramento e gerenciamento de máquinas virtuais e seus serviços são bastante convenientes e necessárias.

Na área de monitoramento em nuvens privadas, pouco foi publicado até o momento. Como a abordagem de computação em nuvem é recente, a maioria dos esforços estão voltados para o desenvolvimento de soluções para o fornecimento, padronização e aceitação geral do paradigma. Atualmente, existem diversas soluções para a implantação de ambientes de computação em nuvem. O OpenNebula, por ser uma ferramenta de código aberto com uma vasta gama de recursos, aliados a um grande esforço por parte dos desenvolvedores e uma comunidade ativa, se mostrou uma das soluções mais completas e ideal para a construção desse ambiente de forma simples e sem maiores investimentos. Com esse trabalho foi possível provar a facilidade de extensão do PCMONS e torná-lo compatível com a ferramenta que provém o modelo de IaaS em computação em nuvem OpenNebula. Grande parte dessa facilidade devido à sua arquitetura genérica e modular,

Host	Status	Services	Actions
oneadmin I-243_stratus	UP	5 OK	[Icons]
oneadmin I-244_stratus	UP	5 OK	[Icons]
oneadmin I-278_stratus	UP	5 OK	[Icons]
oneadmin I-279_stratus	UP	5 OK	[Icons]

Host	Status	Services	Actions
oneadmin I-243_stratus	UP	5 OK	[Icons]
oneadmin I-244_stratus	UP	5 OK	[Icons]
oneadmin I-278_stratus	UP	5 OK	[Icons]
oneadmin I-279_stratus	UP	5 OK	[Icons]

Figura 5. PCMONS: *Hostgroup* próprio e mapeamento entre máquinas virtuais e suas respectivas máquinas físicas.

desenvolvida considerando sua fácil adaptação. O código fonte do PCMONS está disponível sob a licença GPL e pode ser encontrado em <http://code.google.com/p/pcmons/>.

Como trabalhos futuros pretende-se ampliar o suporte da ferramenta PCMONS a outras ferramentas de IaaS, como OpenStack ou Nimbus, e ferramentas de monitoramento como Zabbix e Cacti, além de facilitar sua adaptação por terceiros, através de documentação mais consistente e pelo fornecimento de um conjunto de rotinas e padrões para a utilização das suas funcionalidades, através de uma API. Pretende-se também utilizar e testar a ferramenta em ambientes maiores, com um amplo número de *hosts* e uma grande quantidade de máquinas virtuais, sem limitações de recursos computacionais. Planeja-se também o suporte para a especificação e a verificação do cumprimento de Acordos de Níveis de Serviços, gerando relatórios e possivelmente respostas autônomas para determinados eventos, como a migração de máquinas virtuais em caso de falhas em determinadas máquinas físicas.

Referências

- Chaves, S. (2010). Arquitetura e sistema de monitoramento para computação em nuvem privada. Master's thesis, Universidade Federal De Santa Catarina, Instituto de Informática e Estatística, Programa de Pós-Graduação em Ciências da Computação, Florianópolis.
- Cordeiro, T., Damalio, D., Pereira, N., Endo, P., Palhares, A., Gonç andalves, G., Sadok, D., Kelner, J., Melander, B., Souza, V., and Mã andngs, J.-E. (2010). Open source cloud computing platforms. pages 366–371.
- De Chaves, S., Uriarte, R., and Westphall, C. (2011). Toward an architecture for monitoring private clouds. *Communications Magazine, IEEE*, 49(12):130–137.
- Huang, H., Ruan, Y., Shaikh, A., Routray, R., Tan, C.-h., and Gopisetty, S. (2009). Building end-to-end management analytics for enterprise data centers. In *Proceedings of the 11th IFIP/IEEE international conference on Symposium on Integrated Network Management, IM'09*, pages 661–675, Piscataway, NJ, USA. IEEE Press.
- Hurwitz, J., Bloor, R., Kaufman, M., and Halper, D. F. (2010). *Cloud Computing for Dummies*. Wiley Publishing, EUA, 1 edition.
- Milojić andić and, D., Llorente, I. M., and Montero, R. S. (2011). Opennebula: A cloud management tool. *Internet Computing, IEEE*, 15(2):11–14.

- NIST SP 800-145 (2011). The nist definition of cloud computing. <http://csrc.nist.gov/publications/nistpubs/800-145/SP800-145.pdf>.
- Shao, Z., Jin, H., and Lu, X. (2009). Pmonitor: A lightweight performance monitor for virtual machines. *Education Technology and Computer Science, International Workshop on*, 3:689–693.
- Smoot, S. R. and Tan, N. K. (2012). *Private Cloud Computing*. Morgan Kauffman, EUA, 1 edition.
- Sotomayor, B., Montero, R. S., Llorente, I. M., and Foster, I. (2009). Virtual infrastructure management in private and hybrid clouds. *IEEE Internet Computing*, 13:14–22.
- Vecchiolla, C., Chu, X., and Buyya, R. (2009). Aneka: A Software Platform for .NET-based Cloud Computing. pages 267–295. *High Speed and Large Scale Scientific Computing*, IOS Press, Amsterdam, Netherlands.
- Velte, T., Velte, A., and Elsenpeter, R. C. (2009). *Cloud Computing, A Practical Approach*. Osborne/McGraw-Hill, EUA, 1 edition.
- Verma, D. C. (2009). *Principles of Computer Systems and Network Management*. Springer Publishing Company, Incorporated, 1st edition.