UNIVERSIDADE FEDERAL DE SANTA CATARINA
PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIAS DA COMPUTAÇÃO

Shirlei Aparecida de Chaves

Arquitetura e Sistema de Monitoramento para Computação em Nuvem Privada

Dissertação apresentada ao Programa de Pós-Graduação em Ciências da Computação da Universidade Federal de Santa Catarina como parte dos requisitos para o grau de mestre em Ciência da Computação.
Orientador: Prof. Dr. Carlos Becker Westphall

Florianópolis – SC

2010

Catalogação na fonte pela Biblioteca Universitária
da
Universidade Federal de Santa Catarina

C512a Chaves, Shirlei Aparecida de
Arquitetura e sistema de monitoramento para computação em nuvem privada [dissertação] / Shirlei Aparecida de Chaves ; orientador, Carlos Becker Westphall. - Florianópolis, SC : 2010.
115 f.: il., tabs.

Dissertação (mestrado) - Universidade Federal de Santa Catarina, Centro Tecnológico. Programa de Pós-Graduação em Ciência da Computação.

Inclui referências

1. Ciência da computação. 2. Monitoramento. I. Westphall, Carlos Becker. II. Universidade Federal de Santa Catarina. Programa de Pós-Graduação em Ciência da Computação. III. Título.

CDU 681

Shirlei Aparecida de Chaves

ARQUITETURA E SISTEMA DE MONITORAMENTO PARA COMPUTAÇÃO EM NUVEM PRIVADA

Esta Dissertação foi julgada adequada para obtenção do Título de "Mestre em Ciência da Computação", área de concentração: "Sistemas de Computação" e aprovada em sua forma final pelo Programa de Pós-Graduação em Ciência da Computação.

Florianópolis, 10 de dezembro de 2010.

Prof. Dr. Mário Antônio Ribeiro Dantas - Coordenador

Banca Examinadora:

Prof. Dr. Carlos Becker Westphall - Orientador
Universidade Federal de Santa Catarina

Prof. Dra. Carla Merkle Westphall
Universidade Federal de Santa Catarina

Prof. Dr. Alfredo Goldman vel Lejbman
Universidade de São Paulo

Dr. Fernando Luis Koch
Universidade de Utrecht, Países Baixos

Agradecimentos

Prestar agradecimentos é uma tarefa tão árdua quanto definir os objetivos de trabalho. Ao longo da caminhada são tantas as pessoas que contribuem de alguma forma para que o trabalho chegue a uma finalização, que tentar montar uma lista com todos esses nomes poderia exigir mais espaço do que o próprio trabalho em si! É difícil também quantificar essa contribuição, pois há as contribuições técnicas, como aquela correção importante em uma linha de código, e há as contribuições emocionais, aquelas nas quais às vezes basta saber que a pessoa está disponível quando você precisar desabafar. Portanto, os meus agradecimentos vão a todas as pessoas que fazem parte da minha vida, pois a todas devo um pouco dessa conquista.

Porém, algumas pessoas acabam participando mais ativamente, nos dando conselhos, nos ouvindo e nos dando rumo, diariamente, em bons e (muitos) maus momentos! A essas, um agradecimento em especial:

Ao meu marido, Peter, por existir e estar sempre disposto a escutar muitas reclamações, sem nunca desistir das palavras de conforto e carinho!

Ao professor Carlos, pela companhia ao longo da caminhada e por trabalhar para que o LRG possa disponibilizar a infraestrutura necessária para o desenvolvimento das atividades.

Ao Rafael Uriarte, pelas muitas horas de configurações, programação e disponibilidade em trabalhar nas minhas idéias e escolhas.

E por último, mas não menos importante, ao PPGCC e à UFSC, por disponibilizarem a estrutura necessária e trabalharem para que o Mestrado seja possível e esteja sempre melhorando.

RESUMO

A computação em nuvem, como todo novo paradigma tecnológico, apresenta diversos desafios: consolidação de definição e bases conceituais; estabelecimento de consenso sobre aplicações e benefícios; gerenciamento, entre outros. Este trabalho visou a apresentação desse paradigma, focando-se na atividade gerencial de monitoramento. Apresenta-se uma revisão sobre o estado da arte do conceito computação em nuvem, discussão de seus principais modelos de implantação e entrega de serviços, esforços de padronização, questões de gerenciamento e principalmente, monitoramento. Discute-se uma proposta de arquitetura de monitoramento para computação em nuvem que contempla o modelo de implantação de nuvem privada e o modelo de entrega de serviços conhecido como IaaS (*Infrastructure as a Service* ou Infraestrutura como Serviço). Contempla-se o modelo de nuvem privada e IaaS porque se considera que essa é a combinação que fornece maior controle para a organização que adota o uso da computação em nuvem, assim como também fornece os benefícios de maior aproveitamento do parque tecnológico instalado. Considerou-se também que o modelo de serviço IaaS é um modelo importante a ser estudado e contextualizado, pois pode trazer grandes benefícios às pequenas e médias empresas, laboratórios de pesquisa e organizações governamentais, educacionais, sem fins lucrativos e outras, pois, além de melhorar o aproveitamento do parque tecnológico instalado, pode facilitar atividades acadêmicas, manutenção de software e até postergar aquisição de novos equipamentos. Para teste e validação da arquitetura proposta desenvolve-se um protótipo denominado PCMONS (*Private Cloud MONitoring Systems*), o qual é programado em módulos e utiliza as linguagens de programação Python, Perl e Linux Shell Scripting. O protótipo inicial foca no monitoramento de máquinas virtuais, dando suporte à plataforma de software para computação em nuvem Eucalyptus e a ferramenta de monitoramento Nagios. Para teste geral da arquitetura e protótipo, implanta-se um ambiente de computação em nuvem privada no Laboratório de Redes e Gerência (LRG) do Departamento de Informática e Estatística (INE) da Universidade Federal de Santa Catarina (UFSC). Apresentam-se os resultados obtidos através de um estudo de caso, o qual considera um possível cenário de uso para uma nuvem privada, com monitoramento realizado pelo PCMONS.

Palavras-chave: Computação em Nuvem. Nuvem Privada. Monitoramento.

ABSTRACT

Cloud computing, like any new technological paradigm, presents many challenges: consolidating the definition and conceptual basis, establishing consensus on applications and benefits, management, among others. This work introduces this paradigm, focusing on one of the management activities: monitoring. It presents a review on the state-of-the-art in cloud computing concepts, discuss their main deployment and service delivery models, standardization efforts, management issues and, mainly, monitoring. It also presents and discusses a proposal of an architecture for monitoring cloud computing, which encompasses the private cloud deployment model and the service delivery model known as IaaS (Infrastructure as a Service). Private clouds and IaaS are focused because we consider that this is the combination that provides most control to the organization that adopts the use of cloud computing, and also provides the benefits of increasing the use of installed technological infrastructure. It was also felt that the IaaS service model is an important model to be studied and contextualized, because it can bring great benefits to small and medium enterprises, research laboratories and governmental, educational and nonprofit organizations, and others, because, besides the potential better use of the installed technological infrastructure, it can facilitate academic activities, software maintenance and even to postpone purchase of new equipment. To test and validate the proposed architecture, it is developed a prototype called PCMONS (Private Cloud MONitoring Systems), which is programmed into modules and uses the Python, Perl and Linux Shell Scripting languages. The initial prototype focuses on the monitoring of virtual machines, supporting the Eucalyptus software platform for cloud computing and the monitoring tool Nagios. To test the general architecture and prototype, it is deployed a private cloud computing environment in the Network and Management Laboratory (LRG), Department of Informatics and Statistics (INE), Federal University of Santa Catarina (UFSC). The results obtained are presented through a case study, which considers a possible usage scenario for a private cloud, with monitoring conducted by PCMONS.
Keywords: Cloud computing. Private Cloud. Monitoring.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE QUADROS

LISTA DE ABREVIATURAS E SIGLAS

API – *Application Programming Interface* ou Interface de Programação de Aplicativos
CRM – *Customer Relationship Management* ou Gerenciamento de Relacionamento com o Cliente
IP – *Internet Protocol* ou Protocolo da Internet
MV – Máquina Virtual
QoS – *Quality of Service* ou Qualidade do Serviço
SLA – *Service Level Agreement* ou Acordo de Nível de Serviço
SMS – *Short Message Service* ou Serviço de Mensagem Curta
TI – Tecnologia da Informação
VM – *Virtual Machine* ou Máquina Virtual
SOAP - *Simple Object Access Protocol* ou Protocolo Simples de Acesso a Objetos
REST – *Representational State Transfer* ou Transferência de Estado Representacional
HTTP – *Hypertext Transfer Protocol ou* Protocolo de Transferência de Hierpertexto
SO – Sistema Operacional

SUMÁRIO

1 INTRODUÇÃO

O termo nuvem ou *cloud* é uma metáfora em relação à forma como a Internet é usualmente mostrada nos diagramas de rede - como uma nuvem. Nesses diagramas, o ícone da nuvem representa todas as tecnologias que fazem a Internet funcionar, abstraindo a infraestrutura e a complexidade que ela engloba (VELTE et al. 2009). Utilizar serviços dessa nuvem (armazenamento, banco de dados, processamento, entre outros), em uma breve definição, é fazer computação em nuvem.

Diversos foram os trabalhos de pesquisa e desenvolvimento que possibilitaram o desenvolvimento do que hoje se convencionou chamar computação em nuvem. Entre eles podemos citar a computação utilitária (do inglês *utility computing*), a computação em grade (*grid computing*), virtualização, computação em cluster e computação distribuída de modo geral. Essa constatação é compartilhada por diversos autores, entre eles podemos citar Foster et al. (2008), Vouk (2008), Lim et al. (2009). Além dessa combinação de tecnologias que possibilitam a viabilização do paradigma, outras 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.

1.1 MOTIVAÇÃO

Todas as mudanças pelas quais os paradigmas de sistemas computacionais passam trazem consigo novos desafios. Um deles é no gerenciamento, o qual nunca passa a ser desnecessário, apenas eventualmente mais ou menos automatizado, com uma gama maior ou menor de processos e dispositivos a serem controlados, com um esforço maior ou menor para implantação. Ou seja, as variáveis envolvidas são reagrupadas, mas o propósito geral e necessidade de se gerenciar permanecem constante. Dentre as atividades gerenciais, a de monitoramento exerce um papel fundamental no acompanhamento do sistema sendo gerenciado, possibilitando detectar falhas de funcionamento e também servindo como base para atividades de planejamento e melhorias. Por isso, acom-

panhar essas mudanças e desenvolver ou estender modelos e técnicas de monitoramento é uma atividade constante, desempenhando um papel importante não apenas técnico e gerencial, mas também de agente propagador de novas técnicas e impulsionador de novas tecnologias e paradigmas.

1.2 DEFINIÇÃO DO PROBLEMA E PROPOSTA

A computação em nuvem, pela sua característica de agrupar diversas tecnologias e não se tratar apenas de um paradigma computacional, mas também de um modelo de negócios, traz consigo um amplo espectro de atividades gerenciais de mais alto nível, como por exemplo, qual provedor de serviços escolher, entre outras, além de uma gama de aspectos técnicos mais específicos como que tecnologia de virtualização utilizar, como disponibilizar os dispositivos virtuais e como monitorar seu adequado funcionamento e conformidade com níveis de serviço que tenham sido definidos. Nesse sentido, estabelecer uma arquitetura de monitoramento, que englobe todos esses fatores e permita que seja possível exercer as atividades usuais de planejamento, provisionamento, escalonamento e outras, é uma agregação importante ao desenvolvimento e utilização do paradigma de computação em nuvem. No presente trabalho, pretende-se desenvolver essa arquitetura, focando-se no modelo de implantação de nuvem privada, o qual é discutido no capítulo 2, e levando em consideração padrões já estabelecidos ou que estejam sendo desenvolvidos, voltados a essa modalidade de computação em nuvem. Após o estabelecimento da mesma, se implanta um ambiente de testes, no qual essa arquitetura é colocada em prática para avaliação, utilizando-se as ferramentas disponíveis para esse modelo e estendendo-se suas funcionalidades quando necessário.

1.3 OBJETIVO GERAL

Desenvolver uma arquitetura e sistema de monitoramento para computação em nuvem privada, considerando as peculiaridades desse paradigma de computação e de negócios, que permita não só adicionar um novo sistema ao parque de software da organização, mas que também permita agregar as ferramentas e serviços já existentes.

1.4 OBJETIVOS ESPECÍFICOS

- Investigar as tecnologias envolvidas na computação em nuvem;
- Investigar as opções de software livre para desenvolvimento e implantação da computação em nuvem;
- Implantar uma nuvem privada com propósitos acadêmicos no Laboratório de Redes e Gerência (LRG);
- Desenvolver e implantar uma arquitetura e sistema de monitoramento para nuvem privada; e
- Testar o sistema desenvolvido através de um estudo de caso.

1.5 ORGANIZAÇÃO DO TRABALHO

Capítulo 1 – Introdução: Apresenta a introdução ao tema, contextualização e problema de pesquisa. São apresentados também os objetivos geral e específicos do trabalho.
Capítulo 2 – Computação em Nuvem: Nesse capítulo são apresentadas as diversas categorias de computação em nuvem, utilizações, problemas de gerenciamento e segurança, com a finalidade de, além de se apresentar o tema mais detalhadamente, contextualizar melhor o problema de pesquisa e o desenvolvimento da proposta.
Capítulo 3 – Tecnologia de Máquinas Virtuais ou Virtualização: Apresenta a tecnologia de virtualização, tecnologias relacionadas e sua ligação com computação em nuvem.
Capítulo 4 – Monitoramento em computação em Nuvem: Esse capítulo apresenta características de monitoramento para computação em nuvem, assim como arquiteturas de monitoramento utilizadas na computação em grade, uma vez que computação em nuvem pode fazer uso da mesma. Apresenta também a ferramenta para visualização dos dados de monitoramento utilizada no presente trabalho, o Nagios, além de trabalhos relacionados que utilizam o Nagios em suas estratégias de monitoramento.
Capítulo 5 – Desenvolvimento da Proposta: Nesse capítulo são apresentados e detalhados os passos adotados na construção de um ambiente de pré-testes, para familiarização prática com as tecnologias e softwares para computação em nuvem, além do estabelecimento e desenvolvimento de um protótipo de uma arquitetura de monitoramento para uma nu-

vem privada, a qual é testada em um ambiente de nuvem implantado no Laboratório de Redes e Gerência (LRG).
Capítulo 6 – Conclusão e Trabalhos Futuros: Nesse capítulo apresenta-se a conclusão do trabalho, suas principais contribuições, possibilidades de expansão do protótipo e trabalhos futuros.

2 COMPUTAÇÃO EM NUVEM

Cloud Computing ou computação em nuvem é um termo relativamente novo, para o qual diversas definições foram publicadas ao longo do amadurecimento do tema.

Para Mei, Chan e Tse (2008), computação em nuvem é um paradigma de computação emergente, cujo objetivo é compartilhar dados, computação e serviços de modo transparente entre os usuários de uma grade massiva. Para Vouk (2008), é o próximo passo natural na evolução dos serviços e produtos de tecnologia da informação sob demanda e que de modo geral será baseado em recursos virtualizados. O mesmo autor também cita que computação em nuvem adota a infraestrutura cibernética e é construída baseada em décadas de pesquisa em virtualização, computação distribuída, computação em grade, computação utilitária e mais recentemente, serviços de software, rede e web.

Segundo Foster et al.(2008), é um paradigma de computação distribuída de larga escala, que é impulsionado por economias de escala, na qual um conjunto gerenciado, dinamicamente escalado e virtualizado de serviços, plataformas, armazenamento e poder computacional é entregue sob demanda a clientes externos através da Internet. Para Lin, Dasmalchi e Zhu (2008), é um conceito de negócios e tecnologia emergente, com diferentes significados para diferentes pessoas: para usuários de TI (Tecnologia da Informação) e aplicações, é TI como serviço (ITaaS - IT as a Service); para desenvolvedores de aplicações para a Internet, é uma plataforma de desenvolvimento de software em escala para a Internet e um ambiente de execução; para provedores de infraestrutura e administradores, é uma infraestrutura massiva de *data centers* distribuídos conectados por redes IP (*Internet Protocol*).

Buyya, Yeo e Venugopal (2008), definem computação em nuvem como um tipo de sistema paralelo e distribuído, o qual consiste de uma coleção de computadores virtualizados e interconectados, os quais são dinamicamente disponibilizados e apresentados como um ou mais recursos computacionais unificados, baseados em acordos de nível de serviço estabelecidos através da negociação entre provedores do serviço e consumidores.

Além de diversos pesquisadores, devido ao interesse despertado pelo assunto, há organizações que vêm trabalhando na definição do tema, incluindo órgãos governamentais como o NIST (*National Institute of Standards and Technology* ou Instituto Nacional de Padrões e Tecno-

logia), do governo dos Estados Unidos. A definição formulada pelo grupo de trabalho em computação em nuvem desse órgão é apresentada abaixo:

> Computação em nuvem é um modelo que possibilita acesso conveniente e sob demanda através da rede a um conjunto compartilhado de recursos computacionais configuráveis (por exemplo, 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 (MELL; GRANCE, 2009, p. 1, tradução nossa).

Diversos autores têm adotado essa definição, entre eles pode-se citar Zhang, Cheng e Boutaba (2010), Cloud Security Alliance (2009).

Zhang, Cheng e Boutaba (2010), resumem que computação em nuvem alavanca a tecnologia de virtualização para obter o objetivo de fornecer recursos computacionais como utilidades, compartilhando alguns aspectos com computação em grade e computação autonômica, como o emprego de recursos distribuídos para alcançar os objetivos do nível de aplicação e provisionamento automática de recursos. Segundo ainda Zhang, Cheng e Boutaba (2010), a computação em nuvem difere das tecnologias de computação autonômica e em grade em alguns aspectos como, por exemplo, se concentrar mais na redução de custo dos recursos e não na redução de complexidade do sistema.

A partir dessas caracterizações, pode-se notar que alguns aspectos são geralmente repetidos. São os de ser um novo paradigma, não apenas um paradigma de computação distribuída, mas também um novo paradigma de negócios, com o qual se pretende prover sob demanda poder computacional, software, armazenamento e até mesmo uma infraestrutura de *data centers* distribuída.

Independentemente de qual ou quais das definições acima citadas se opte ou se prefira, a computação em nuvem ocupa atualmente uma posição de destaque, muito bem resumida por Armbrust et al. (2009, p. 91, tradução nossa): computação em nuvem emergiu como uma das mais influentes tecnologias na indústria de TI e está revolucionando o modo como os recursos de TI são gerenciados e utilizados.

2.1 CLASSIFICAÇÕES PARA COMPUTAÇÃO EM NUVEM

Dada a ampla abrangência tanto da definição quanto dos usos para computação em nuvem, diversas classificações são apresentadas, levando em consideração características de implantação e uso, as quais são apresentadas nessa seção. Antes dessa apresentação, no entanto, destacam-se algumas observações importantes com relação a essas classificações.

a) Dois elementos ou atores são comumente citados: provedor e consumidor, sendo o provedor aquele que oferece o serviço e o consumidor, aquele que se utiliza do serviço.
b) Segundo a DMTF (2010c), esse modelo provedor/consumidor é necessário para gerenciar os recursos da nuvem, já que eles podem ser de vários tipos. O modelo descreve o modo no qual um serviço é ofertado e consumido. A entrega efetiva do serviço é modelada através de entidades-modelo específicas do domínio (como no caso do IaaS, um servidor virtual, por exemplo).

2.1.1 Classificação conforme modelo de serviços

Na classificação de acordo com o modelo de serviços, são consideradas as funções ou serviços que o consumidor tem acesso. De modo geral, em todos eles o consumidor não gerencia a infraestrutura por trás da nuvem, a não ser em casos que ele próprio tenha implantado essa nuvem computacional.

• Plataforma como Serviço (*Platform as a Service* ou PaaS):

PaaS se refere ao fornecimento de recursos da camada de plataforma, incluindo suporte a sistemas operacionais e frameworks de desenvolvimento de software (ZHANG; CHENG; BOUTABA, 2010). Leavitt (2009) destaca ainda que esses recursos podem ser acessados e utilizados *online*, até mesmo em colaboração com outros usuários. Exemplos de provedores de PaaS incluem Google App Engine (GOOGLE, 2010), Microsoft Windows Azure (MICROSOFT, 2010) e SalesForce.com (SALESFORCE, 2010).

• Software como Serviço (*Software as a Service* ou SaaS):

Segundo Zhang, Cheng e Boutaba (2010), SaaS diz respeito ao provisionamento sob demanda de aplicações pela Internet. Exemplos de SaaS incluem SalesForce.com (SALESFORCE, 2010), Rackspace (RACKSPACE, 2010), e SAP Business ByDesign (SAP, 2010).

Esses programas são acessíveis a partir de vários dispositivos, através do uso de uma interface como, por exemplo, um navegador (MELL; GRANCE, 2009).

• Infraestrutura como Serviço (*Infrastructure as a Service* ou IaaS):

Para Zhang, Cheng e Boutaba (2010), IaaS se refere ao fornecimento sob demanda de recursos de infraestrutura, usualmente em termos de MVs. Uma explicação mais detalhada é dada na definição abaixo:

> O domínio IaaS é usualmente definido para incluir recursos dos seguintes tipos: servidores (virtuais ou físicos); armazenamento; conexões de rede e endereços IPs - públicos ou privados, expostos pelas interfaces de rede. Adicionalmente o modelo de recursos para a camada IaaS pode incluir recursos que representam características adicionais da infraestrutura, tais como características de rede (por exemplo, balanceamento de carga), templates (para implantação) e *snapshots* (DMTF, 2010a, p.18, tradução nossa).

Zhang, Cheng e Boutaba (2010) salientam ainda que o proprietário da nuvem que oferece IaaS é chamado de provedor de IaaS e que exemplos desses provedores incluem a Amazon EC2 (AMAZON, 2010), GoGrid (GOGRID, 2010) e Flexiscale (FLEXISCALE, 2010).

Na Figura 1, são ilustradas as principais categorias de modelo de serviço para computação em nuvem, exemplificando os principais usuários de cada modelo, além da visibilidade e valor dos serviços para cada tipo de usuário.

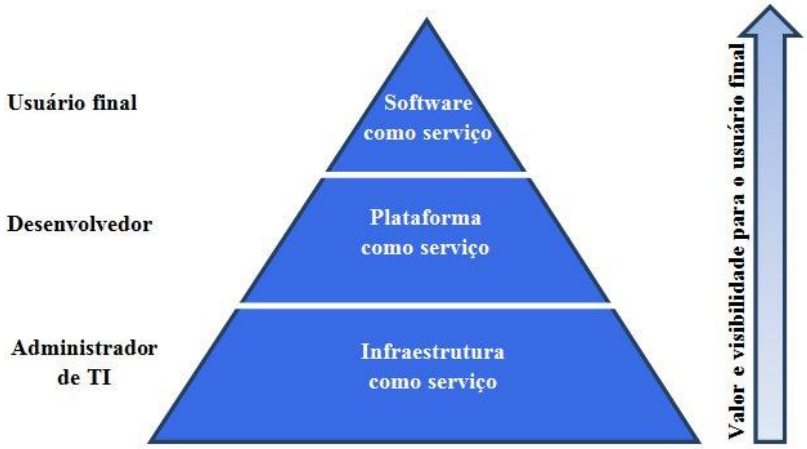


Figura 1 - Principais categorias de serviço para computação em nuvem (adaptado de INTEL, 2010).

Essa organização em forma de pirâmide exibida na Figura 1 está feita de modo a mostrar a visibilidade e valor para os usuários finais (INTEL, 2010), isto é, o usuário final tem mais visibilidade da aplicação que ele acessa pela Internet do que de uma MV, da qual ele pode nem ter conhecimento. Como computação em nuvem também se trata de um modelo de negócios, a autora desse trabalho visualiza uma analogia à tradicional pirâmide organizacional, na qual na base têm-se o operacional, representado pelos administradores de TI, no meio da pirâmide o tático, representado pelos desenvolvedores e, no topo, o estratégico, representado pelos usuários finais, interessados no serviço final, abstraindo os detalhes técnicos ou táticos.

Para finalizar a seção sobre os modelos de serviço, segundo a Cloud Security Alliance (2009), cada um desses modelos tem suas desvantagens com relação à extensibilidade e responsabilidade quanto à segurança, pois conforme o modelo elas são mais responsabilidade do cliente ou mais responsabilidade do provedor da nuvem.

2.1.2 Classificação quanto ao Modelo de Implantação

Além da categorização por modelos de serviço, existe também uma classificação por modelo de implantação (ou, eventualmente, se possa dizer por público alvo ou abrangência de público), como, por exemplo, se é uma nuvem aberta ao público em geral ou não. Essa classificação é a seguinte:

• **Nuvem pública:** segundo Armbrust et al. (2009), se a nuvem for disponibilizada para o público em geral no modo *pay-as-you-go* (algo como pague conforme o uso), ela é chamada nuvem pública. Em Mell e Grance (2009), essa definição é compartilhada, acrescentando ainda que a nuvem poderia estar disponível para o público em geral ou para um grande grupo industrial

• **Nuvem privada:** para Armbrust et al. (2009), o *data center* interno à uma organização, não disponível ao público em geral, é a nuvem privada. Para Mell e Grance (2009), uma nuvem privada é a infraestrutura operada apenas por uma organização, podendo ser gerenciada pela própria organização ou eventualmente por terceiros e pode tanto existir dentro como fora dos limites da organização.

Segundo Sotomayor et al. (2009), o objetivo principal de uma nuvem privada não é vender capacidade pela Internet, através de interfaces acessíveis ao público em geral, mas sim dar aos usuários locais uma infraestrutura ágil e flexível para atender cargas de trabalho de serviços dentro de seu próprio domínio administrativo.

• **Nuvem híbrida:** para Sotomayor et al. (2009), uma nuvem privada pode dar suporte a uma nuvem híbrida, através da complementação da capacidade da infraestrutura local com a capacidade computacional de uma nuvem pública, e essa nuvem privada/híbrida pode permitir acesso remoto através de interfaces remotas, como a de Web Services que a Amazon EC2 utiliza. Mell e Grance (2009) complementam ainda que no caso de uma nuvem híbrida, que é uma composição de uma ou mais nuvens (privada, comunitária ou pública), cada uma das nuvens permanece uma entidade única, a qual é conectada com as demais por tecnologias proprietárias ou padronizadas que permitem portabilidade de dados e aplicações.

Da última definição, a de nuvem híbrida, traz-se à tona um quarto tipo de nuvem que alguns autores citam: a de nuvem comunitária, a qual também se descreve brevemente abaixo.

• **Nuvem comunitária:** para Mell e Grance (2009), nesse caso 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 INFRAESTRUTURA DE SOFTWARE PARA COMPUTAÇÃO EM NUVEM

Diversos são os sistemas ou ferramentas que auxiliam na implantação e uso de uma nuvem, seja ela pública, privada ou híbrida. Esses sistemas podem dar mais atenção a um dos modelos de serviço, como IaaS, por exemplo, ou eventualmente abranger mais de um. Nessa seção, destacam-se as principais ferramentas de código aberto para implementação do modelo de IaaS. A lista não pretende ser exaustiva, apenas citar as mais comumente encontradas na literatura revisada para esse trabalho e descrever suas características e projetos de implantação de nuvem privada/pública/híbrida existentes que os utilizam. São eles: (a) *Eucalyptus:* Nimis, Lenk e Klems (2010), Sotomayor et al. (2009), Baun e Kunze (2009), entre outros; (b) *OpenNebula:* Nimis, Lenk e Klems (2010), Sotomayor et al. (2009), entre outros; (c) *Nimbus*: Marshall, Keahey e Freeman, (2010), Sotomayor et al. (2009), Moreno-Vozmediano, Montero,e Llorente (2009), entre outros autores.

2.2.1 Eucalyptus

O Eucalyptus, sigla para *Elastic Utility Computing Architecture Linking Your Programs To Useful Systems*, ou, numa tradução livre Arquitetura de Computação Elástica e Utilitária Para Ligar Seus Programas a Sistemas Úteis, é, de acordo com os desenvolvedores (EUCALYPTUS, 2010), uma infraestrutura de software código aberto para implementação de nuvem aproveitando a TI da empresa e a infraestrutura do provedor de serviços existente localmente. Essa característica é uma das grandes atratividades do Eucalyptus, pois acaba tornando a configuração e utilização da computação em nuvem algo relativamente rápido e viável dentro de uma organização. Os próprios desenvolvedores

destacam essa característica, tanto em documentos acadêmicos (o Eucalyptus é o resultado de um projeto de pesquisa da Universidade de Santa Bárbara, na Califórnia), como em documentos comerciais. Segundo eles,

> (...) o eucalyptus possibilita a implantação de nuvem híbrida e privada em *data centers* empresarias e não requer hardware de propósito especial ou reconfiguração. Aproveitando tecnologias de serviços web e Linux que comumente existem na infraestrutura atual de TI, o Eucalyptus permite que clientes fácil e rapidamente criem nuvens computacionais 'no local', adaptadas às suas necessidades específicas de aplicações (EUCALYPTUS, 2009 , tradução nossa).

Outro atrativo do Eucalyptus é o fato dele ser compatível com a interface da plataforma Amazon AWS (*Amazon Web Services*), o que permite que uma nuvem implantada localmente possa interagir com uma nuvem pública usando essa interface de programação. Permite, também, que imagens de máquinas virtuais utilizadas no Amazon EC2 (*Elastic Computing Cloud*) possam também ser utilizadas no Eucalyptus.

Maiores detalhes técnicos sobre o funcionamento do Eucalyptus serão discutidos no Capítulo 5, porém destacam-se nessa seção ainda alguns projetos internacionais que utilizam o Eucalyptus:

a) NASA Nebula[1]

O propósito desse projeto é oferecer uma solução de computação que possa de modo rápido atender os requisitos de um grande número de projetos. O NASA Nebula Services pretende fornecer SaaS, PaaS e IaaS e usa o Eucalyptus na parte de IaaS, como gerenciador das máquinas virtuais. Na descrição do projeto, na referência a máquinas virtuais, o Eucalyptus é citado como uma API (*Application Programming Interface* ou Interface de Programação de Aplicações) código aberto clone da plataforma de nuvem Amazon AWS.

b) Ubuntu Enterprise Cloud (UEC) [2]

Desde a distribuição 9.04 do Ubuntu houve uma parceria da Canonical, desenvolvedora do Ubuntu, com a equipe do Eucalyptus, de modo a torná-lo o mecanismo por trás de uma solução de computação em nuvem já empacotada no próprio sistema operacional. Naquela versão, ele ainda não fazia parte da distribuição em si, porém poderia ser

1 http://nebula.nasa.gov/services/

2 http://www.ubuntu.com/cloud

baixado e instalado através de repositórios usando ferramentas de gerenciamento de pacotes do Ubuntu como o *apt-get*, por exemplo.

Na distribuição Server Edition do Ubuntu 9.10 (eucalyptus 1.6.1) e 10.04 (eucalyptus 1.6.2), o eucalyptus passou a fazer parte integrante da distribuição, sendo possível durante o processo de instalação do sistema operacional escolher a opção "*Instalar Ubuntu Enterprise Cloud*" , a qual orienta na instalação e configuração dos componentes necessários do eucalyptus. O que se deve obter, após essa instalação, segundo informações da própria distribuição, é uma nuvem com um controlador *front-end* e um ou mais nodos para se rodar máquinas virtuais. A partir daí, com algumas configurações extras, é possível ter uma nuvem privada.

2.2.2 OpenNebula[3]

O OpenNebula, de acordo com os desenvolvedores, é um kit de ferramentas código aberto e baseado em padrões para computação em nuvem. Assim como no caso do Eucalyptus, o OpenNebula começou como um projeto de pesquisa e possui quando da realização dessa pesquisa tanto a versão código aberto como a versão comercial, através da empresa fundada para tratar da demanda comercial, a C12G Labs. Na versão código aberto, os hipervisores atendidos são o Xen, KVM e VMware. Ele também atende nuvens híbridas como o EC2 da Amazon e Elastic Hosts (provedor de IaaS).

Alguns projetos internacionais que utilizam o OpenNebula:

a) Reservoir[4]

O Reservoir é um projeto do FP7 (*Seventh Framework Programme***)** da União Européia, cujo objetivo é possibilitar escalonamento massivo de implantação e gerenciamento de serviços complexos de TI através de diferentes domínios administrativos, plataformas de TI e localização geográfica. Segundo a descrição disponível no site do projeto, a intenção é prover a fundação para uma economia de serviços baseados online, na qual, através do uso de tecnologias de virtualização, recursos e serviços seriam providos de modo transparente e gerenciados sob demanda, com preços competitivos e alta qualidade de serviço.

3 http://www.opennebula.org/

4 http://www.reservoir-fp7.eu/

b) StratusLab[5]

O StratusLab é um projeto iniciado em novembro de 2008 com a finalidade de explorar a integração de tecnologias de computação em nuvem e serviços, especialmente virtualização, com infraestruturas existentes de computação em grade.

2.2.3 Nimbus[6]

O Nimbus também é um kit de ferramentas de código aberto para computação em nuvem. Quando da realização dessa pesquisa, a versão 2.4 disponível atendia os hipervisores KVM e Xen e implementação parcial das APIs SOAP (*Simple Object Access Protocol* ou Protocolo Simples de Acesso a Objetos) e Query do Amazon EC2.

Segundo os desenvolvedores, a missão do projeto Nimbus é desenvolver a infraestrutura com ênfase nas necessidades científicas, mas muitos casos de uso não científicos também são atendidos.

Além das ferramentas acima citadas, há diversos outros projetos, como AbiCloud, Enomaly (a versão código aberto), oVirt, entre possivelmente muitos outros.

2.3 PADRONIZAÇÕES PARA COMPUTAÇÃO EM NUVEM

Conforme computação em nuvem ganha mais adeptos e maior atenção, diversos grupos se formaram e se formam para trabalhar no sentido de padronizá-la, visando especialmente interoperabilidade. Dentre essas iniciativas, destacam-se as seguintes.

2.3.1 Open Cloud Consortium (OCC) [7]

5 http://www.stratuslab.org

6 http://www.nimbusproject.org/

7 http://opencloudconsortium.org/

O OCC, de acordo com informações disponíveis em seu próprio site, está focado nas seguintes tarefas:

1 – Suporte ao desenvolvimento de padrões para computação em nuvem e de *frameworks* para interoperabilidade entre nuvens;
2 – Desenvolvimento de *benchmarks* para computação em nuvem;
3 – Suporte a implementações de referência para computação em nuvem, de preferência implementações de referência código aberto;
4 – Gerenciamento de *testbeds* para computação em nuvem, incluindo o *Open Cloud Testbed* e o *Intercloud Testbed*;
5 – 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 e no momento dessa pesquisa, havia os seguintes grupos:

- *Working Group on Standards and Interoperability For Large Data Clouds* (Grupo de trabalho em Padrões e Interoperabilidade para Grandes Nuvens de Dados).

O Foco desse grupo é o desenvolvimento de padrões para a interoperabilidade de grandes nuvens de dados, como padrões para interface para armazenamento e computação de dados. O grupo também está desenvolvendo *benchmarks* para grande quantidade de dados a serem computados em uma nuvem.

- *The Open Cloud Testbed Working Group* (Grupo de Trabalho para o *Testbed* de Nuvem Aberta).

No momento dessa pesquisa, o acesso a esse *Testbed* era limitado a membros do OCC que contribuem com recursos como rede e poder computacional.

- *The Open Science Data Cloud (OSDC) Working Group* (Grupo de Trabalho para a Nuvem Aberta de Dados Científicos).

Esse grupo de trabalho gerencia e opera uma grande nuvem de dados para dados científicos e está no momento dessa pesquisa fechado para membros do grupo.

- *Intercloud Testbed Working Group* (Grupo de Trabalho para o *Testbed* Inter Nuvem).

O *testbed* sendo desenvolvido por este grupo visa explorar serviços baseados no IF-MAP (*Interface for the Metadata Access Point*) pra mapear entidades relevantes às nuvens.

2.3.2 Open Cloud Computing Interface (OCCI) [8]

O Open Cloud Computing Interface (OCCI) ou Interface para Computação em Nuvem Aberta, numa tradução livre, é um grupo de trabalho do *Open Grid Forum* (OPEN GRID FORUM, 2010), o qual já tem um histórico de trabalhos em busca de padronização para computação em grade.

De acordo com informação disponibilizada no site do grupo de trabalho, a proposta do mesmo é produzir a especificação de uma API para o gerenciamento remoto da infraestrutura de computação em nuvem, possibilitando o desenvolvimento de ferramentas interoperáveis para tarefas comuns, incluindo implantação, escalabilidade autonômica e monitoramento. Ainda de acordo com o grupo, o escopo dessa especificação abrange todas as funcionalidades de alto nível necessárias ao gerenciamento do ciclo de vida de máquinas virtuais (ou *workloads*), sendo executadas em tecnologias virtualizadas (ou containeres) com suporte a serviços elásticos.

O OCCI trabalha na entrega de dois produtos: um documento com casos de uso e requisitos e o outro com a especificação de um protocolo. Ambas as entregas são detalhadas abaixo.

a) Casos de uso e requisitos para uma API de nuvem

O documento em questão é a primeira entrega para demonstrar e validar as características do Open *Cloud Computing Interface* ou Interface de Computação em Nuvem aberta, numa tradução livre. Ele é uma descrição informal de Casos de Uso e requisitos para a API de Nuvem OCCI e registra as necessidades de gerentes e administradores de computação em nuvem no modelo IaaS, na forma de Casos de Uso (quatorze ao total). Esses casos de uso servem como um guia primário para o desenvolvimento de requisitos da API.

Dentre os referidos casos de uso, alguns são mais relevantes para o escopo do trabalho em questão e são brevemente apresentados abaixo.

1) Infra-estrutura de nuvem SLA-aware usando SLA @ SOI

Esse caso de uso trata de interfaces padrão para provisionamento de infraestrutura e também que esse provisionamento use SLAs que possam ser lidos por máquinas. Alguns requisitos funcionais são destacados para o escopo desse trabalho:

8 http://www.occi-wg.org

- *Descrição da MV*: um modo de adicionar restrições não funcionais em atributos funcionais.

- *Gerenciamento de MV*: todos os parâmetros da requisição devem ser possíveis de se monitorar e verificar. Controle total de recursos (MV) alocados é necessário; no mínimo: iniciar, parar, suspender e resumir.

- *Monitoramento de MV*: Monitoramento de restrições não funcionais declaradas na requisição de provisionamento.

2) Gerenciador de serviços para controlar o ciclo de vida dos serviços

Com relação a monitoramento, esse caso de uso descreve que o status de qualquer elemento é dado na forma de uma lista de chaves e seus valores. Por exemplo, se o status de um elemento de memória for dado em quantidade de memória utilizada, essa informação seria fornecida na forma (usando uma pseudo-especificação):

Memória[usada] = 430 MB

Memória[cache] = 142 MB

Tanto a requisição quanto a resposta devem usar os mesmos identificadores para o elemento. Além disso, dois tipos de monitoramento devem ser atendidos:

- *Baseado em requisição (pull-based)*: O gerente de serviço pode requisitar o status de qualquer elemento que ele tenha registrado: MVs, redes, e outros. Pode também requisitar o status de componentes, como por exemplo, o status de certo disco de uma certa MV.

- *Baseado no modelo publish/subscribe*: O gerente de serviço pode se inscrever para ser notificado sobre eventos acerca de MVs e/ou redes. Alguns eventos que podem ser notificados são:

  - Erros em componentes da MV;
  - Mudanças no estado de uma MV (por exemplo de ATIVA para SUSPENSA); e
  - Notificações periódicas sobre estados de elementos. A freqüência dessas notificações pode ser configurada na mensagem de inscrição.

b) Protocolo OCCI

O protocolo OCCI é um protocolo código aberto para computação em nuvem, construído utilizando o protocolo HTTP (*Hypertext Transfer Protocol ou* Protocolo de Transferência de Hipertexto) e interface RESTful (*Representational State Transfer* ou Transferência de Estado Representacional). Nesse protocolo, por exemplo, as operações de criação, recuperação, atualização e remoção de um recurso são mape-

adas para as operações HTTP de POST, GET, POST/PUT e DELETE, respectivamente.

2.3.3 Open Cloud Standards Incubator

Esta é uma iniciativa da DMTF (*Distributed Management Task Force* ou Força Tarefa em Gerenciamento Distribuído), cujo foco é a padronização de interações entre os ambientes de nuvem através do desenvolvimento de especificações que entreguem semântica arquitetural e detalhes de implementação para obter gerenciamento de nuvem que seja interoperável entre provedores de serviço, seus consumidores e desenvolvedores (Cloud Management Standards). Até o momento da escrita desse trabalho, dois documentos principais (para o escopo desse trabalho) foram produzidos por essa iniciativa, os quais são apresentados adiante nesse tópico. Antes de entrar na descrição desses documentos, no entanto, apresentam-se algumas definições que aparecem nessas descrições, conforme caracterização da própria DMTF (DMTF, 2010b) e que também serão úteis no decorrer desse trabalho. São elas:

Serviço de Nuvem: um serviço disponível publicamente ou um serviço privado que é usado dentro de uma organização.

Interface de provedor: a interface através da qual os consumidores de serviços da nuvem acessam e monitoram os serviços contratados. Essa interface cobre negociação de SLA, acesso e monitoramento de serviços e cobrança.

Gerente de serviços: responsável por gerenciar a instância e topologia de um serviço. Ele fornece facilidades para o administrador criar instâncias de máquinas virtuais e serviços, além de prover mecanismos para criação, monitoramento, controle e relatórios de serviços.

Imagem Virtual: um elemento (normalmente parte de um pacote usando o OVF - *Open Virtualization Format*) que encapsula uma carga de trabalho que consiste de todo o código necessário para executar essa carga, com os metadados que são necessários para configurar o ambiente no qual executa. Maiores detalhes sobre esse elemento são trabalhados na Seção 3.4 Máquinas Virtuais e Imagens Virtuais.

Abaixo, um resumo dos referidos documentos produzidos pelo grupo de trabalho:

1) Casos de Uso e Interação para Gerenciamento de Nuvens:

Concentra-se em casos de uso, interações e formato de dados. O escopo do documento inclui interações e troca de artefatos de dados entre consumidores, desenvolvedores e provedores e provê uma lista de casos de uso (dezenove, distribuídos ao longo dos estados do ciclo de vida de um serviço de nuvem) de gerenciamento que englobam todo o ciclo de vida de um serviço de nuvem.

Alguns casos de uso, considerados mais relevantes para o escopo do presente trabalho, são apresentados resumidamente:

DMTF-005 Relatórios Contratuais (Contract Reporting)
Trata de relatórios recebidos pelo cliente, acerca de um serviço contratado.
Exemplos de tipos de relatórios:
• uso do serviço
– total
– por usuário
• consumo vs. limites contratados
• disponibilidade vs. níveis contratados
• conformidade com desempenho/SLA
• violações de SLA
• comprometido vs. disponível
• métricas por tarefas operacionais (por exemplo, quantidade; tempo de resposta)
• log de auditoria para atividades de tarefas operacionais

Note-se que são todas métricas de desempenho para serem comparadas com métricas pré-definidas. Relatórios com finalidade de cobrança são tratados à parte.

DMTF-008 Provisionamento de Recursos

Uma das indicações importantes nesse caso de uso, para o presente trabalho, é o passo 12, o qual envolve os passos necessários para o provisionamento de um serviço, como por exemplo:

- Conectar servidores em um rack
- Alocar sistemas computacionais
- Configurar zonas em um sistema
- *Implantar e ativar agentes de monitoramento*.

O caso de uso salienta ainda que esses passos podem ou não ser automatizados.

DMTF-011 Monitoramento de Recursos de Serviço

Esse caso de Uso descreve um passo importante no monitoramento de recursos:

- O Administrador de Serviços ao Consumidor (ASC) submete uma requisição de monitoramento para coletar medidas (disponíveis em um relatório) ou para monitorar limites (os quais poderiam resultar em um relatório de dados ou notificação). A requisição identifica um ou mais serviços ou recursos instanciados e um intervalo de tempo. Isto é, a requisição é feita acerca de um serviço, num determinado intervalo de tempo.

Em tempo: ASC, de acordo com a DMTF (2010b) é o ator do sistema que requisita instâncias de serviço e mudanças nas instâncias de serviço, cria usuários (incluindo políticas), aloca recursos, gera relatórios, etc.

DMTF-015 Notificação de Condição de Serviço ou Evento

Nesse caso, para um serviço configurado e em operação, determinadas condições ou eventos operacionais de tempo de execução foram identificados ou detectados, para as quais é necessária notificação imediata da condição ou evento ao consumidor do serviço. Um exemplo é a detecção de uma intrusão ou mudança inesperada de configuração.

2) Arquitetura para Gerenciamento de Nuvens

Este *white paper* descreve a arquitetura de referência no que se refere à interfaces entre um provedor de serviços de nuvem e um consumidor de serviços da nuvem. O documento também descreve requisitos de alto nível que um provedor de serviço deveria disponibilizar como parte da oferta de serviços para a nuvem.

Um desses requisitos, no tocante ao escopo desse trabalho, é o de *logs*, gerenciamento de eventos, resposta a incidentes e notificação. Esse requisito preconiza que, durante a operação das entidades de serviço (por exemplo, MVs ou outros serviços) na nuvem, o provedor de serviços da nuvem deve estar monitorando as entidades de serviço para registrar eventos relativos à operação, cobrança, conformidade e outros. O provedor de serviços da nuvem deve deixar essa informação disponível para o consumidor de serviços da nuvem de acordo com o SLA, em tempo real, em lote ou ambos.

Outro requisito não diretamente relacionado com o escopo dessa pesquisa, mas que aproxima esse grupo de trabalho do grupo apresentado anteriormente, o Open Cloud Consortium e sua API OCCI, é em

relação à pilha de protocolos atendidos. Para a DMTF (2010a) a interface deve dar suporte a todos os Padrões de Troca de Mensagem (*Message Exchange Patterns* - MEP) relevantes ao domínio, o que inclui o MEP *one-way* (*push* e *pull*) e o MEP *two-way* (síncrono e assíncrono), sob um protocolo requisição-resposta como o HTTP. Deve também prover acesso programável a vários provedores de serviços para a nuvem, usando mecanismos padrão da indústria como o REST e o SOAP.

2.4 FERRAMENTAS DE GERENCIAMENTO

Além das ferramentas básicas fornecidas pelos próprios softwares de computação em nuvem, como a ferramenta de linha de comando euca2ools do Eucalyptus, há outras ferramentas sendo desenvolvidas e testadas pela comunidade de software aberto, como é o caso do HybridFox ou com propósitos comerciais, como a RightScale. Nesse tópico apresentam-se algumas dessas ferramentas, com suas principais características de uso e limitações.

2.4.1 ElasticFox / Hybridfox

Ambas as ferramentas são *plug-ins* ou extensões para o navegador Firefox. No caso do ElasticFox, é possível gerenciar uma conta com a Amazon EC2, realizando atividades como iniciar novas instâncias de máquinas virtuais.

Como o código do ElasticFox é aberto, possibilitando customizações e novos desenvolvimentos a partir dele, surgiu um outro projeto chamado HybridFox, no qual é possível realizar as mesmas atividades que o ElasticFox numa conta da Amazon EC2, mas também num ambiente computacional com o Eucalyptus. As principais atividades são:

- Gerenciar imagens (ver Figura 2)
- Iniciar e parar instâncias de máquinas virtuais
- Gerenciar instâncias (ver Figura 3)
- Gerenciar IPs elásticos
- Gerenciar grupos de segurança
- Gerenciar pares de chaves
- Gerenciar blocos de armazenamento elástico

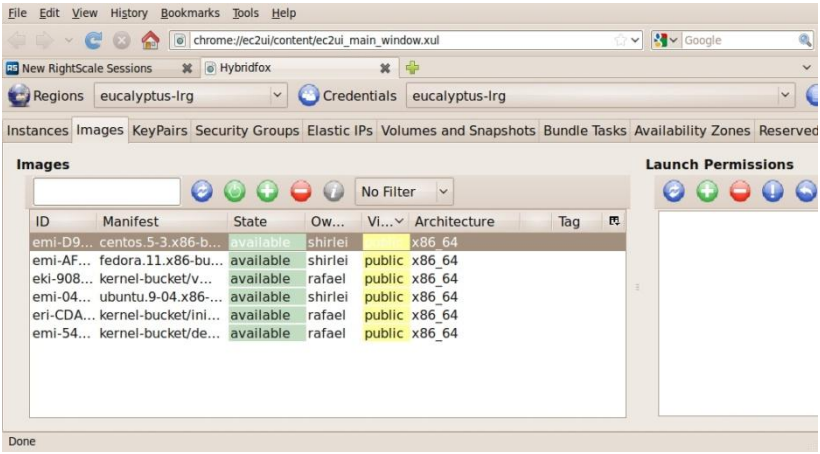


Figura 2 - Tela de gerenciamento de imagens no HybridFox

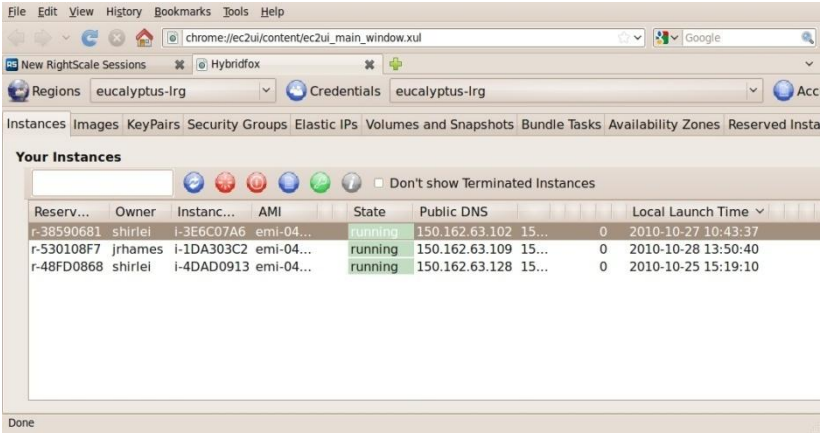


Figura 3 - Tela de gerenciamento de instâncias no HybridFox

O HybridFox é funcional e permite instalação e configuração rápidas, além de fácil uso para gerenciamento de atividades básicas como o instanciamento de máquinas virtuais. No entanto, ele é um *plug-in* específico para o navegador Firefox, o qual, apesar de amplamente utilizado (segundo estatísticas da W3Schools[9] 46,6% de uso em Junho de 2010), não é universal e atualizações do navegador podem acarretar na

9 http://www.w3schools.com/browsers/browsers_stats.asp

necessidade de atualização do HybridFox, como acontece com outros *plug-ins*, exigindo que os desenvolvedores lancem rapidamente novas versões.

2.4.2 RightScale

O RightScale é uma plataforma de gerenciamento de computação em nuvem baseada na web, que fornece possibilidade de gerenciamento de diversos provedores e serviços, como Amazon AWS, Eucalyptus, Rackspace, GoGrid, entre outros.

Além das versões comerciais, há uma versão *free* que o cliente pode assinar, porém com muitos recursos presentes nas edições pagas limitados ou ausentes, como monitoramento.

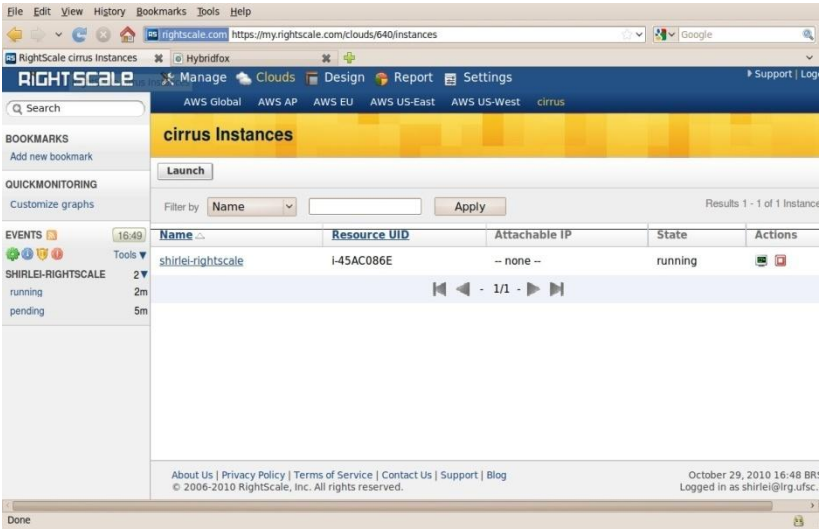


Figura 4 - Tela de gerenciamento de instâncias no RightScale

Na Figura 4 é mostrada a tela de gerenciamento de instâncias para a conta de testes criada na versão “Free Edition” do RightScale.

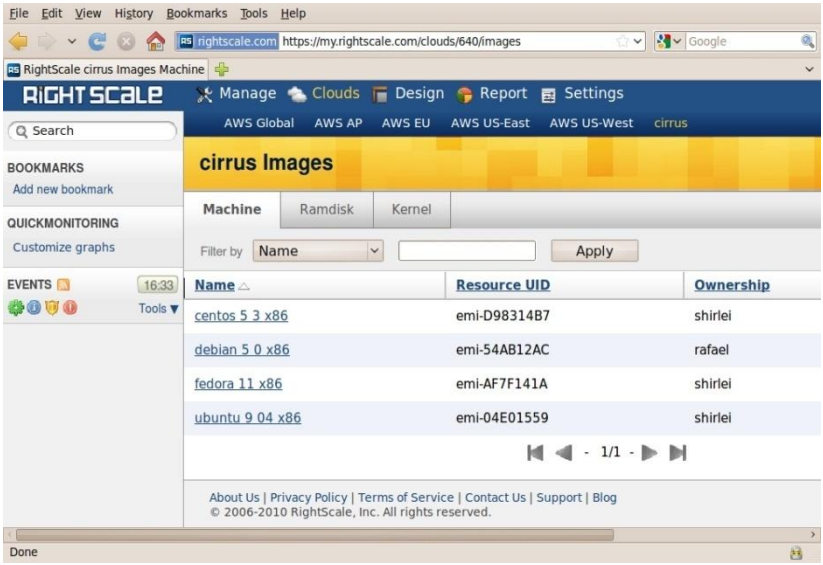


Figura 5 - Tela de visualização de imagens no RightScale

Na versão “Free Edition” do RightScale testada nesse trabalho, não é possível registrar imagens, apenas visualizar as imagens que já estão registradas na nuvem, conforme Figura 5.

2.4.3 Euca2ools

O Euca2ools é um conjunto de ferramentas de linha de comando, que servem para interagir com *web services* que exportem uma API baseada em REST/Query compatível com os serviços EC2 e S3 da Amazon. As ferramentas podem ser usadas tanto com os serviços da Amazon como com instalações do Eucalyptus. Segundo (EUCALYPTUS, 2010), essas ferramentas foram inspiradas nas ferramentas de linha de comando distribuídas pela Amazon (*api-tools* e *ami-tools*), e aceitam basicamente as mesmas opções e variáveis de ambiente. No entanto, o euca2ools foi desenvolvido do zero em Python, utilizando-se as bibliotecas Boto e o kit de ferramentas M2Crypto (EUCALYPTUS, 2010).

Diversas ações são possíveis com o euca2ools, como por exemplo, listar as imagens virtuais disponíveis, instanciar uma máquina vir-

tual e verificar as instâncias de máquina virtual que estão sendo executadas, conforme a Figura 6.

```
cirrus:/home/shirlei # euca-describe-images
IMAGE   emi-D98314B7    centos.5-3.x86-bucket/centos.5-3.x86.img.manifest.xml   shirlei available       public          x86_64  machine
IMAGE   emi-AF7F141A    fedora.11.x86-bucket/fedora.11.x86.img.manifest.xml     shirlei available       public          x86_64  machine
IMAGE   eki-90831384    kernel-bucket/vmlinuz-2.6.24-19-xen.manifest.xml        rafael  available       public          x86_64  kernel
IMAGE   emi-04E01559    ubuntu.9-04.x86-bucket/ubuntu.9-04.x86.img.manifest.xml shirlei available       public          x86_64  machine
IMAGE   eri-CDA21462    kernel-bucket/initrd.img-2.6.24-19-xen.manifest.xml     rafael  available       public          x86_64  ramdisk
IMAGE   emi-54AB12AC    kernel-bucket/debian.5-0.x86.img.manifest.xml   rafael  available       public          x86_64  machine
cirrus:/home/shirlei # euca-run-instances -k sckey emi-04E01559
RESERVATION     r-48FD0868      shirlei shirlei-default
INSTANCE        i-4DAD0913      emi-04E01559    0.0.0.0 0.0.0.0 pending sckey   2010-10-25T17:19:10.442Z        eki-90831384    eri-CDA21462
cirrus:/home/shirlei # euca-describe-instances
RESERVATION     r-582A0964      shirlei default
INSTANCE        i-409308B6      emi-04E01559    150.162.63.117  150.162.63.117  running         sckey   0       m1.small        2010-10-20T16:52:23.91Z
RESERVATION     r-405306E2      shirlei default
INSTANCE        i-34CA8702      emi-54AB12AC    150.162.63.127  150.162.63.127  running         sckey   0       m1.small        2010-10-14T00:02:17.536Z
RESERVATION     r-48FD0868      shirlei default
INSTANCE        i-4DAD0913      emi-04E01559    150.162.63.128  150.162.63.128  running         sckey   0       m1.small        2010-10-25T17:19:10.442Z
RESERVATION     r-426007CB      shirlei default
INSTANCE        i-523B096E      emi-04E01559    150.162.63.120  150.162.63.120  running         sckey   0       m1.small        2010-10-19T17:28:09.519Z
cirrus:/home/shirlei #
```

Figura 6 – Consulta por imagens, instanciação de máquina virtual e verificação de instâncias sendo executadas usando-se o euca2ools.

Até o momento da escrita desse trabalho, o euca2ools era uma ferramenta disponível para sistemas operacionais baseados em Linux.

Todas as ferramentas apresentadas aqui são configuradas para acessar os dados da nuvem através do fornecimento das seguintes informações:

- URL de acesso ao controlador da nuvem. Exemplo: http://cirrus.lrg.ufsc.br:8773/services/Eucalyptus
- Chave de acesso ao controlador
- Chave privada do usuário, cadastrado para acesso à nuvem, o qual vai fazer consulta aos serviços disponíveis.

Note-se, portanto, que informações sensíveis podem ser fornecidas a terceiros (no caso de uso do RightScale, por exemplo), gerando uma necessidade de confiança entre o usuário do serviço e o provedor da ferramenta, que pode ser estabelecida por contrato de serviço ou outra forma de negociação, a qual não faz parte do escopo desse trabalho.

2.5 CONCLUSÃO

Esse capítulo apresentou e resumiu as principais características e definições para computação em nuvem, dentre as quais se destacam o modelo de implantação (privada, pública, híbrida ou comunitária) e o modelo de serviços (IaaS, PaaS e SaaS). Além dessas definições, foram apresentadas algumas das principais plataformas de software para com-

putação de código aberto, o Eucalyptus e o OpenNebula, assim como os principais projetos que as utilizam. Foram também destacadas algumas das iniciativas em busca da padronização da computação em nuvem, as mais relevantes no escopo do presente trabalho. Para finalizar, apresentou-se algumas das ferramentas de software livre ou com uso livre limitado para gerenciamento de uma nuvem, notadamente características de gerenciamento de imagens e instanciamento de máquinas virtuais.

3 TECNOLOGIA DE MÁQUINAS VIRTUAIS OU VIRTUALIZAÇÃO

De acordo com Ray e Schultz (2009), virtualização se refere às tecnologias projetadas para prover uma camada de abstração entre o hardware de um sistema computacional e o software sendo executado sob esse hardware. Para Daniels (2009), é um ambiente ou camada de abstração entre os componentes de hardware e o usuário final.

O uso de máquinas virtuais tem diversas aplicações, incluindo, por exemplo, a de implantação de software, devido a sua habilidade de encapsular os recursos de armazenamento de uma máquina virtual em um único arquivo de imagem de disco virtual, o qual aparece, para o sistema operacional visitante, como um disco rígido (CHEN et al., 2009). Esse arquivo de imagem pode então ser distribuído para qualquer máquina, a qual pode ser inicializada (feito *boot*) a partir dessa imagem, pelo monitor de máquina virtual para reconstruir o mesmo ambiente de execução virtual sem o processo de reinstalação de software. Esse paradigma de implantação baseada em imagem de disco virtual é especialmente útil para ambientes de computação em larga escala, os quais podem ser heterogêneos e distribuídos e podem incluir um grande número de máquinas (CHEN et al., 2009).

Pode-se perceber com a descrição acima, que a tecnologia de máquinas virtuais ganha um destaque renovado dentro do universo de computação em nuvem, não só pela possibilidade de se obter melhor utilização de recursos pela disponibilidade de novas máquinas a partir de um mesmo hardware, como também por essa maior facilidade para implantação de software através do uso de imagens virtuais, essencial para se prover serviços sob demanda de modo rápido.

3.1 POR QUE VIRTUALIZAR?

Para Kirk (2009), a virtualização está transformando os *data centers*, devido a sua habilidade de consolidar recursos de hardware e reduzir custos com energia. Ainda de acordo com Kirk (2009), desenvolvimentos recentes na tecnologia de virtualização a posicionaram como uma tecnologia núcleo para computação em nuvem. Dentre esses desenvolvimentos, o autor cita dois principais:

- Habilidade de se mover uma máquina virtual em execução, juntamente com seu sistema operacional e aplicações, para um host físico sem grandes paralisações;

- Habilidade de se registrar (*logging*) em tempo real as ações de uma máquina virtual, com o propósito de se poder retroceder um sistema por completo até um determinado ponto e então avançá-lo novamente, com a finalidade de depurá-lo ou auditá-lo.

Para Krishna (2009), virtualização é a chave para se prover melhor utilização de servidores e uma melhor flexibilidade de uso e alocação de recursos.

3.2 TÉCNICAS UTILIZADAS NA VIRTUALIZAÇÃO

Primeiramente, é preciso apresentar dois novos conceitos, o de sistema operacional hospedeiro e o de sistema operacional visitante.

O sistema operacional hospedeiro é o sistema que é executado diretamente sobre o hardware físico, enquanto o sistema operacional visitante é o sistema que é executado na máquina virtual (hardware virtualizado).

Para que o sistema operacional visitante possa executar como se estivesse executando em uma máquina física (acessar dispositivos de hardware, por exemplo), é adicionada uma camada de software entre o hardware e o sistema operacional, chamada de Monitor de Máquina Virtual, sendo conhecido também pelo termo hipervisor. Esses monitores podem ser implementados de duas formas: paravirtualização ou virtualização total.

- **Paravirtualização**: técnica de virtualização que permite que o sistema operacional esteja ciente de que está sendo executado sob um hipervisor e não diretamente sob o hardware (Xen.org, 2010). As instruções a serem executadas sob o hardware são repassadas para o hipervisor. O núcleo do sistema operacional, no entanto, precisa ser modificado para se acomodar a essa situação.

- **Virtualização total:** Nesse caso, o hipervisor fornece uma máquina emulada, na qual o sistema operacional é executado. O núcleo do sistema operacional não precisa ser modificado, mas como o sistema operacional visitante não sabe da existência do hipervisor, as instruções executadas pelo sistema operacional são primeiramente testadas pelo hipervisor antes de executar no hardware.

Além das formas de implementação citadas acima, por software, nos últimos anos foram desenvolvidos processadores com extensão para terem virtualização no próprio hardware (ADAMS; AGESEN, 2006), facilitando a implementação da virtualização total. É o caso da tecnologia IVT (*Intel Virtualization Technology*) da Intel (INTEL, 2010b) e do AMD-V (AMD *Virtualization*), da AMD (AMD, 2011).

Diversas ferramentas de software implementam as técnicas de paravirtualização ou virtualização total e a seguir apresentam-se algumas das principais implementações do mercado para a plataforma x86.

3.2.1 Xen

O primeiro lançamento público do hipervisor Xen foi feito em 2003, como um projeto de pesquisa da Universidade de Cambridge. Essa apresentação pode ser conferida em detalhes no artigo intitulado Xen e a Arte da Virtualização (BARHAM, 2003). Ele é um software de código aberto para virtualização ou, mais detalhadamente, conforme descrição do próprio fabricante, "uma camada de software que roda diretamente no hardware do computador, substituindo o sistema operacional e assim permitindo que o mesmo hardware rode diferentes sistemas operacionais convidados, concorrentemente" (Xen.org, 2010, pág. 1, tradução nossa).

De acordo com (Xen.org, 2010), um computador rodando o hipervisor Xen contém três componentes:

- O hipervisor Xen, que roda diretamente sobre o hardware e é a interface para todas as requisições de hardware (I/O, CPU,...) feitas pelos sistemas operacionais convidados.

- Um domínio privilegiado, chamado *Domain 0* (Dom0), que é um convidado especial sendo executado no hipervisor com acesso direto ao hardware e com responsabilidades de gerenciamento de convidados. Esse domínio é executado pelo hipervisor Xen durante a inicialização do sistema e pode rodar qualquer sistema operacional, à exceção do Windows. O Dom0 tem privilégios especiais no acesso ao hipervisor Xen e pode executar tarefas administrativas, relativas aos domínios não privilegiados (DomU), como gerenciamento de I/O. Também pode ser usado por administradores de sistema, pois é possível se conectar nesse domínio e gerenciar todo o sistema computacional.

- Múltiplos domínios convidados sem privilégios (*Unprivileged Domain Guests*), conhecidos como DomU. Esses domínios rodam no hipervisor, mas não tem acesso direto ao hardware. Esses domínios são iniciados e controlados pelo Dom0, e agem de modo independente no sistema. Esses convidados podem ser executados com um sistema operacional modificado, processo conhecido como paravirtualização, ou um sistema sem modificação, exigindo hardware com suporte à virtualização, conhecido como *Hardware Virtual Machine* ou HVM.

3.2.2 KVM

O *Kernel-based Virtual Machine* (Máquina Virtual baseada em Kernel) ou KVM é um hipervisor de código aberto mais novo que o Xen, foi lançado em 2007. Ao contrário do Xen, o KVM não utiliza paravirtualização, mas sim a virtualização total e necessita de um suporte à virtualização por parte da CPU. Outra diferença do KVM em relação ao Xen é a forma pela qual interage com o núcleo (*kernel*) do sistema operacional. No caso do KVM, ele é um módulo carregável do núcleo.

De acordo com Habib (2008), uma instalação típica do KVM é composta pelos seguintes componentes:

- Um *driver* de dispositivo para gerenciamento do hardware de virtualização. As funcionalidades desse *driver* estão disponíveis através do dispositivo de caracteres /dev/kvm;

- Um componente de espaço de usuário para emular o hardware do PC; e

- Um modelo de Entrada/Saída derivado diretamente do QEMU (software de código aberto para emulação de máquinas e virtualizador), com suporte a características do QEMU, como a *copy-on-write* de imagens de disco.

Outra questão relevante em relação ao KVM é que, como não requer mudança no núcleo do sistema operacional, pode executar qualquer sistema operacional que tenha sido implantado para um processador compatível.

3.2.3 VMware

A VMware foi pioneira na virtualização da plataforma x86, estando no mercado desde 1998.

O VMware (produto de mesmo nome da empresa), é um framework para virtualização da plataforma x86, o qual possui diversos produtos, tanto para servidores quanto para desktops. Isto é, é possível ter tanto uma versão para servidor, como o VMware Server, gerenciando diversas máquinas virtuais, como uma versão para rodar uma imagem virtual em um *desktop*. A técnica utilizada é a de virtualização total, embora haja suporte para virtualização através da Interface de Máquina Virtual disponível em algumas implementações do produto. Esse suporte, no entanto, já teve sua aposentadoria em novos produtos a partir de 2010-2011 anunciado pela empresa. Os produtos VMware apresentam versões *free*, como o VMware Server e versões pagas, como o VMware vSphere.

3.4 MÁQUINAS VIRTUAIS E IMAGENS VIRTUAIS

Máquinas virtuais executam sistemas operacionais, os quais podem ser normalmente disponibilizados de duas maneiras. Uma delas é instalar diretamente na maquina virtual, a partir do drive de CD-ROM. Outra forma é carregá-lo a partir de um sistema operacional préconfigurado, em um arquivo comumente chamado de imagem virtual ou apenas imagem, conforme descrito no início do presente capítulo. Os aplicativos que sejam necessários posteriormente podem ser instalados mais tarde, com a máquina virtual em execução, ou sendo adicionados à imagem.

Ou seja, é essa imagem que determina o estado inicial de uma máquina virtual e seus aplicativos, isto é, quais serão suas características logo após o boot. Wei, Zang e Ammons (2009), expõem muito bem essa questão quando afirmam que o estado inicial de cada máquina virtual na nuvem é determinado por alguma imagem, e, portanto, essa imagem precisa ter sua integridade altamente garantida. Os mesmos autores também citam que, além da integridade, há a necessidade se compartilhar essas imagens de modo seguro, pois algumas das vantagens da computação em nuvem dependem do uso de imagens fornecidas por terceiros.

Na Figura 7, Wei, Zhang e Ammons (2009), exemplificam, em alto nível, uma arquitetura típica de uma nuvem computacional, focando especialmente na questão das imagens. Nesse cenário típico, com os nomes de personagens comumente usados para cenários relacionados à segurança – Alice e Bob, os autores demonstram as formas básicas pelas quais um usuário pode acessar um gerenciador da nuvem: através de um portal, via navegador ou através de uma interface de linha de comando, similar à oferecida pelo serviço EC2 da Amazon e pela ferramenta euca2ools do Eucalyptus, apresentada na Seção 2.4.3.

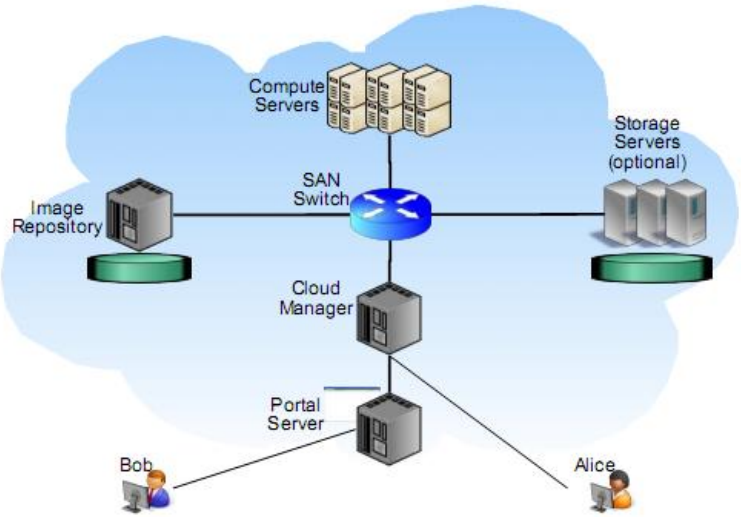


Figura 7 - Arquitetura típica, em alto nível, de uma nuvem (WEI; ZHANG; AMMONS, 2009).

No cenário da Figura 7, são demonstrados três tipos de recursos fornecidos por uma nuvem: um repositório de imagens para máquinas virtuais, um conjunto de servidores nos quais essas imagens poderiam ser executadas e também um dispositivo de armazenamento, para a persistência de dados que o usuário deseje armazenar. Na Figura 7, fica também evidente o papel importante desempenhado pelas imagens de máquinas virtuais no cenário de computação em nuvem. Essas imagens, além de desempenharem um papel único no ambiente em questão, apre-

sentam riscos de segurança e necessidades de gerenciamento individuais, conforme listados abaixo:

1) Imagens de máquina virtual, por definição, são inativas ou dormentes (WEI, ZHANG e AMMONS, 2009, pg. 92, tradução nossa).
2) Imagens de máquina virtual são projetadas para serem utilizadas e compartilhadas por diferentes usuários, normalmente sem nenhuma relação entre si (WEI, ZHANG e AMMONS, 2009, pg. 92, tradução nossa).
3) Imagens de máquina virtual não são simplesmente uma porção estática de informação do usuário, mas uma entidade que inclui toda a pilha de software que vai inicializar uma máquina virtual em seu estado inicial, ou, no caso de um *snaptshot*, num estado (checkpoint) previamente armazenado (WEI, ZHANG e AMMONS, 2009, pg. 92 , tradução nossa).

3.5 CONCLUSÃO

A computação em nuvem faz uso de diversas tecnologias já consolidadas. Entre elas, uma das principais é a virtualização, a qual foi apresentada nesse capítulo com especial destaque a fatores como porque virtualizar, técnicas de virtualização disponíveis. Dentro desse cenário, apresentaram-se também os principais hipervisores ou Monitores de Máquina Virtual, considerando-se os que são livres, de código aberto ou com licença de uso parcialmente livre, como o Xen, KVM e VMware, os quais são a camada de software que fornece ao sistema operacional visitante a abstração da máquina virtual. Para finalizar, apresentaram-se também as imagens virtuais e seu papel no funcionamento do sistema. O objetivo do capítulo não foi esgotar o tema, cuja literatura é extensa e complexa, mas prover uma visão geral, de modo a contextualizar sua importância para a computação em nuvem.

4 MONITORAMENTO EM COMPUTAÇÃO EM NUVEM

A computação em nuvem faz uso de tecnologias como computação utilitária, computação em grade e outras. A área de computação em grade, por exemplo, possui consideráveis esforços de pesquisa e implementação. O Open Grid Forum especificou uma arquitetura de monitoramento, a GMA (*Grid Monitoring Architecture* ou Arquitetura de Monitoramento de Grade), a qual usa uma abordagem produtor/consumidor, levando em consideração que uma grade pode ter milhares de recursos e usuários simultâneos (OPEN GRID FORUM, 2002). A arquitetura referida apresenta três tipos de componentes:

1) *Serviço de diretório*: para publicação de eventos e descoberta. Nesse serviço, os produtores e os consumidores que aceitam mensagens de controle se registram para que os interessados tomem conhecimento de sua existência e possam se comunicar.

2) *Produtores*: tornam os dados de desempenho disponíveis

3) *Consumidores*: recebem os dados de desempenho disponibilizados pelos produtores.

Note-se a questão de que os consumidores recebem os dados de desempenho, isto é, a comunicação é entre produtor e consumidor, o serviço de diretório tem a finalidade exclusiva de iniciar o contato entre as partes (que uma tome conhecimento da outra). A comunicação entre produtores e consumidores na arquitetura GMA pode se dar por três tipos de interação: *publish/subscribe* (interessados se inscrevem para receber determinadas publicações), *query/response* (interessado requisita a informação do produtor no esquema pergunta/resposta) e notificação (comunicação em um estágio, em que o produtor envia os dados de desempenho para o consumidor).

Em Chung e Chang (2009), é apresentada a arquitetura GRIM (*Grid Resource Information Monitoring* ou Monitoramento de Informação de Recursos da Grade). Essa arquitetura é dividida em três camadas, a saber:

1) *Query layer* (camada de consulta): o projeto dessa camada é similar ao da camada de visualização do PCMONS (apresentado na próxima seção), no sentido de que usuários com diferentes expectativas estão envolvidos. Na camada de consulta os usuários podem requisitar informações de interesse acerca de algum recurso, enquanto na camada de visualização do PCMONS a intenção é habilitar diferentes interfaces para visualização dos dados de monitoramento.

2) *Mediation layer* (camada de mediação): essa camada da arquitetura GRIM intermedeia a consulta dos usuários e a atualização das informações dos *hosts* da camada *information provider layer*. No PCMONS, a camada de integração tem objetivo similar, pois faz a integração entre a camada de visualização e a de infraestrutura, traduzindo requisições de uma camada para a outra, quando necessário, assim como agregando e sumarizando as informações coletadas.

3) *Information provider layer* (Camada de informações do provedor): consiste dos hosts sendo monitorados. Um host na arquitetura GRIM é qualquer recurso que provê serviços de grade e os hosts têm sensores para detectar seu estado (CHUNG; CHANG, 2009). Na arquitetura PCMONS, a camada de infraestrutura tem propósito e comportamento similares, uma vez que é composta pelos recursos computacionais da nuvem, além de serem adicionados sensores às máquinas virtuais durante o processo de boot.

Há ainda, no entanto, muito debate sobre o relacionamento/comparação entre computação em nuvem e computação em grade. Algumas diferenças são mencionadas, especialmente em termos de provisionamento e interface de serviços, como em Wang et al. (2008):

- *Interfaces centradas no usuário*: os usuários não precisam mudar seus hábitos de trabalho, como linguagens de programação. Para acessar os serviços e recursos de grade, os usuários precisam aprender comandos e APIs específicas da computação em grade. Brock e Goscinski (2010) mencionam que o nível de expertise para usar uma nuvem é significativamente menor se comparada com uma grade.

- *Provisionamento de serviços sob demanda*: fornecidos pelos provedores de nuvem; usuários podem instalar softwares para personalizar seu próprio ambiente computacional.

- *Oferta de QoS garantida*: não há espaço para melhor esforço e em geral um SLA é negociado com o cliente, considerando níveis de desempenho, disponibilidade e outros.

Outra característica importante é que as nuvens são gerenciadas por entidades únicas (BROCK; GOSCINSKI, 2010), enquanto que grades não têm uma entidade central de gerenciamento.

Diversos são os motivos pelos quais é necessário o monitoramento em um ambiente de computação em nuvem. Segundo Elmroth e Larsson (2009), os principais são:

a) Garantir que as máquinas virtuais obtenham a capacidade estipulada em SLAs; e

b) Coletar dados para contabilidade dos recursos sendo utilizados, necessários para cobrança, por exemplo.

Além dos motivos acima citados, Elmroth e Larsson (2009) indicam ainda dois tipos complementares de monitoramento que precisam ser feitos pelo sistema de monitoramento:

a) Medições da infraestrutura, as quais incluem medições de baixo nível dos recursos sendo usados pela MV, como quantidade de memória RAM ou largura de banda; e
b) Indicadores-chave de desempenho específicos da aplicação, por exemplo, o número de usuários conectados num servidor.

As razões citadas para o monitoramento podem ser definidas como o conjunto de *por que* monitorar, porém há outros dois lados que requerem atenção: *o que* monitorar e *como* monitorar.

O que monitorar está intimamente ligado ao serviço contratado e aos níveis de serviço acordados em um SLA.

Como monitorar sofre influência do que está sendo monitorado, como por exemplo, intervalo de tempo. De acordo com Elmroth e Larsson (2009), conceitualmente, o monitoramento de recursos pode ser efetuado em intervalos de tempo contínuos ou discretos, sendo que a medição contínua é um tipo especial de medição discreta, já que o intervalo de tempo entre uma medição e outra é muito pequeno, fazendo com que a coleta seja na forma de um fluxo de dados. Elmroth e Larsson (2009) citam ainda que os dados coletados possam ser entregues processados ou não, e de acordo com três esquemas: logo que possível, em intervalos regulares ou quando determinada situação é verificada.

Outros princípios e técnicas de monitoramento também podem ser levados em consideração ao se projetar um sistema de monitoramento para computação em nuvem, como por exemplo, os relacionados à computação em grade. A computação em grade pode formar a base da infraestrutura de computação em nuvem e por isso técnicas de monitoramento em computação em grade também devem ser levadas em consideração quando se analisa formas de monitoramento para computação em nuvem. Para Imamagic e Dobrenic (2007), monitorar sistemas distribuídos, tais como os de grade (assim como os de nuvem), é crucial para a operação normal de todos os subsistemas. Atividades como auditoria eficiente de segurança, detecção de falhas, manutenção, escalonamento de trabalho, desempenho dos recursos, entre outras, são possíveis através da coleta constante de informações do sistema.

Em computação em nuvem, as atividades de definição de o que e como monitorar nem sempre são tarefas triviais, pois além de depende-

rem do modelo sendo usado (IaaS, PaaS, SaaS), envolvem também aspectos legais e a própria dinamicidade do paradigma (por exemplo, uma máquina virtual que em um momento estava instanciada para atender determinada demanda, em outro momento já não está mais). É necessário também se considerar que monitoramento inclui coleta de dados, os quais podem ser de natureza sensível, necessitando de mecanismos de segurança.

O presente trabalho, considerando essas peculiaridades, procura desenvolver uma arquitetura de monitoramento que leve em consideração os aspectos discutidos na presente seção. Além do desenvolvimento dessa arquitetura, apresentada no Capítulo 5, é também desenvolvido um protótipo de testes, utilizando quando possível ferramentas já bem estabelecidas, como é o caso do Nagios, apresentado a seguir.

4.1 NAGIOS

O Nagios é um framework código aberto para monitoramento de hosts e serviços, cujo principal propósito é a detecção de falhas.

Segundo a documentação disponível no site[10] oficial do software, o sistema de monitoramento do Nagios provê as seguintes funcionalidades:

- **Monitoramento compreensível**: todos os componentes de infraestrutura de missão crítica (aplicações, serviços, sistemas operacionais, protocolos de rede, métricas de sistema e infraestrutura de rede) são monitorados.
- **Visibilidade:** fornece uma visão central de todos os processos de negócio e rede de operações de TI.
- **Conscientização:** a equipe de TI pode receber alertas via email ou SMS (*Short Message Service* ou Serviço de Mensagem Curta). Possui também capacidade de notificação multiusuário, garantindo que os alertas cheguem à atenção das pessoas necessárias.
- **Correção de Problemas:** Tratadores de problemas permitem que serviços, aplicações, servidores e dispositivos sejam reiniciados automaticamente quando algum problema é detectado.

10 http://www.nagios.org/about

- **Planejamento de Capacidade e Tendência**: permite que a organização planeje upgrades de infraestrutura com antecedência.
- **Relatórios**: Verifica se SLAs estão sendo cumpridos, provê dados históricos de falhas, notificações e resposta a alertas para análises futuras.
- **Arquitetura extensível**: Provê integração fácil com aplicações internas e de terceiros. Isso é possível através de centenas de *plug-ins* ou extensões desenvolvidas pela comunidade de código aberto.

Como o Nagios é uma ferramenta popular e de código aberto, que vem sendo desenvolvida a mais de 10 anos, contando com ampla documentação e suporte pela comunidade, além de ter flexibilidade de uso através da criação de módulos específicos (através de *plug-ins*, por exemplo), diversos trabalhos na área de monitoramento o utilizam como ferramenta de suporte.

Imamagic e Dobrenic (2007) utilizam o Nagios num ambiente de grade, projetando e implementando diversas extensões e abordagens para problemas específicos. No trabalho citado (IMMAGIG; DOBRENIC, 2007), exploram uma característica importante da ferramenta, a qual eles chamam de sensores. Esses sensores ou *plug-ins* são componentes externos que são invocados pelo Nagios para fazer alguma verificação. A distribuição básica do Nagios vem com um conjunto básico de *plug-ins*, como verificação de carga (*check_load*) do sistema, de conexões diversas (*check_http, check_tcp, ...*), entre outros. No entanto, sensores personalizados podem ser desenvolvidos em qualquer linguagem, fazendo com que o Nagios possa monitorar praticamente qualquer recurso desde que possa ser desenvolvido um sensor ou *plug-in* para a finalidade desejada.

Stelte e Hochstatter (2009) também utilizam a característica de extensibilidade do Nagios, desenvolvendo um *plug-in* chamado iNagMon (*Network Monitoring on the iPhone*), para permitir a fácil interação com o Nagios através da tela de dispositivos móveis como o iPhone. O interesse de tal estudo e desenvolvimento se deve ao fato de dispositivos móveis serem amplamente utilizados e também ao fato mencionado pelos autores de ser o Nagios amplamente reconhecido como uma ferramenta importante de monitoramento.

Em Oliveira et. al (2010), foi desenvolvido um sistema de gerenciamento de PLC (*Power Line Communication*), a tecnologia que utiliza a rede de energia elétrica para envio de dados de telecomunicações. No sistema desenvolvido pelos autores citados, diversos *plug-ins* foram

construídos para o Nagios, entre eles os de gerenciamento de fluxo de rede, controle de tráfego, de ruído, entre outros.

No presente trabalho, essa característica de escalabilidade do Nagios é aproveitada também para viabilizar o protótipo de testes da arquitetura de monitoramento de nuvem privada proposta.

4.2 CONCLUSÃO

O presente capítulo, considerando que computação em grade pode fazer parte da computação em nuvem, apresentou algumas arquiteturas propostas para monitoramento de grades, assim como destacou aspectos de monitoramento a serem considerados particularmente para computação em nuvem. Foi possível observar também que o Nagios é bastante utilizado em outros trabalhos, como ferramenta de monitoramento e visualização dos dados monitorados.

5 EXPLORAÇÃO INICIAL DO PARADIGMA E FERRAMENTAS

Para uma interação inicial com o tema, especialmente ferramentas e funcionalidades, um ambiente de pré-testes foi configurado, conforme apresentado na Figura 8.

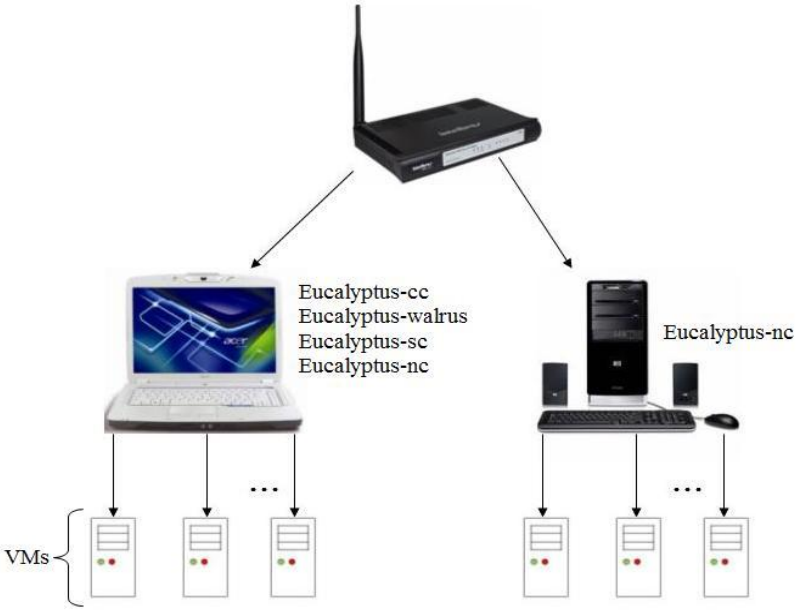


Figura 8 - Ambiente de pré-testes.

O hardware apresentado na Figura 8 foi utilizado para fins exploratórios, isto é, uma maior familiaridade com o tema e ferramentas, conforme citado anteriormente. A especificação técnica desse hardware é dada na Tabela 1, assim como são citados os softwares utilizados, os quais são apresentados em maiores detalhes no decorrer da presente seção.

Tabela 1 - Descrição das ferramentas de software e hardware utilizadas no ambiente de pré-testes

Hardware	**Software**	**Papel no ambiente**
Notebook Acer Aspire 5920 Processador: Intel® Core™ 2 Duo T5750 Cache do Processador: 2MB L2 Memória RAM: 4GB DDR2 667MHz Disco Rígido (HD): 250GB SATA 5400rpm	OpenSUSE 11.1 64 bits Eucalyptus release 1.6.1 Xen versão 3.3.1 Nagios versão 3.0.6	Controlador de cluster Controlador de nuvem Controlador de armazenamento Nodo
CPU HP Pavilion 6205BR Pentium Processador: 4 3.6GHz Memória RAM: 1 GB DDR2 333 MHz Disco Rígido (HD): 250GB	OpenSUSE 11.1 32 bits Eucalyptus release 1.6.1 Xen versão 3.3.1	Nodo
Roteador Wireless Intelbras WRG 240 E 54Mbps		Atribuir IPs e acesso à Internet para máquinas virtuais

O sistema operacional openSUSE foi escolhido por ter suporte ao Eucalyptus e também suporte nativo ao Xen, sendo que esse último tem sua instalação e configuração bastante facilitadas pela ferramenta de instalação e configuração fornecida pelo openSUSE, o YaST.

O Eucalyptus foi escolhido por ser uma plataforma para computação em nuvem de código aberto, bastante documentada, oferecer suporte para diversas distribuições Linux, como o Ubuntu (distribuição que inclusive incorporou o Eucalyptus a partir do Ubuntu 9.04), CentOS 5, Debian “Squeeze” e OpenSUSE 11. Além disso, por oferecer uma interface similar a EC2 da Amazon, amplamente utilizada e considerada por alguns um padrão de facto quando se trata se serviços para nuvem. Por fim, o seu design hierarquizado visa e facilita (mas não limita) o seu uso em laboratórios e em pequenas e médias empresas (Nurmi et al. 2009). Além desses fatores, salienta-se também que no momento da definição do andamento dos trabalhos no Laboratório de Redes e Gerência, do qual a autora dessa dissertação faz parte, o assunto computação

em nuvem e suas ferramentas eram ainda muito recentes e grupos de trabalho foram organizados para explorar as tecnologias existentes, ficando a presente autora responsável pelo Eucalyptus, bastante popular à época. Posteriormente o grupo de pesquisa da Universidade de Santa Bárbara, nos Estados Unidos, arrecadou um capital disponível para empresas iniciantes e se desmembrou da Universidade, mantendo a fundação do Eucalyptus código aberto e adicionando outras características em uma versão comercial. A versão de código aberto continua sendo desenvolvida e mantida, com constantes atualizações. No início dos trabalhos a versão disponível era a 1.6.1 e atualmente o software já se encontra na versão 2.0.1, lançada em novembro de 2010. Ou seja, o Eucalyptus continua se enquadrando no foco desse trabalho, de se utilizar apenas softwares de código aberto.

A escolha do hipervisor de modo geral está atrelada ao software para computação em nuvem escolhido. No caso do Eucalyptus, na versão de código aberto, há suporte para dois hipervisores também de código aberto: o Xen e o KVM. O hipervisor ou monitor de máquina virtual é quem permite que seja executado o sistema operacional hóspede, construindo as interfaces virtuais para as interfaces reais do sistema hospedeiro. Há diversas maneiras de se implementar o mecanismo de virtualização, conforme apresentado no Capítulo 3 – Tecnologias de Máquinas Virtuais ou Virtualização, e cada hipervisor pode estar apto a trabalhar com uma dessas técnicas e eventualmente mais de uma. Para o caso dos dois hipervisores citados, 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 (SANTOS; CHARÃO, 2008), não precisando de suporte nativo à virtualização pelo hardware. Para acomodar o uso dessa técnica, o núcleo do sistema operacional precisa ser adaptado. No caso do hipervisor KVM (*Kernel-based Virtual Machine*), o hardware utilizado deve possuir suporte à virtualização pelo processador, característica que era encontrada apenas em computadores de grande porte, como mainframes, até o lançamento das extensões IVT (Intel *Virtualization Technology*) pela Intel e da AMD-V (AMD *Virtualization*), pela AMD (SANTOS; CHARÃO, 2008). O próprio KVM é recente, tendo sido lançado oficialmente em 2007. Essa virtualização por suporte nativo no hardware é chamada de Máquina Virtual de Hardware ou *Hardware Virtual Machine* (HVM). 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. No

hardware utilizado para testes, ambos os processadores eram Intel, mas nenhum deles com suporte à virtualização, portanto o hipervisor escolhido foi o Xen. É importante salientar, no entanto, que há diversos estudos comparando o desempenho de diversos hipervisores e cada um tem suas vantagens em um ou outro aspecto estudado ou testado. A escolha entre um e outro, em geral, está relacionada à flexibilidade de uso, desempenho e considerações estratégicas/estudo e planejamento do ambiente onde vai ser utilizado (CERBELAUD; GARG; HUYLEBROECK, 2009)

O Nagios foi escolhido pelas características já citadas na Seção 4.1 Nagios. No presente trabalho, sua escolha foi também fortalecida pelo fato de a plataforma de software para computação em nuvem escolhida, o Eucalyptus, estar usando o Nagios para um monitoramento inicial de seus componentes, através do fornecimento de um script para monitoramento dos seus componentes (ping e carga): nodos, controlador de cluster, controlador da nuvem.

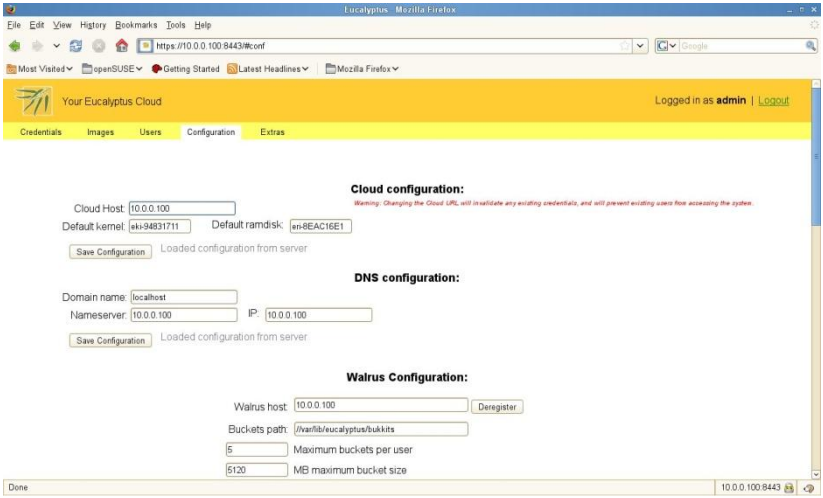


Figura 9 - Interface web para configuração da nuvem, a partir da qual são geradas as credenciais

O Eucalyptus é composto por quatro componentes de alto nível, cada um com sua própria interface de *web service*. São eles (Nurmi et. al, 2009):

1) Controlador de nodo (*node controller* – nc) : controla a execução, inspeção e término das instâncias de MVs, no host onde está sendo executado.

2) Controlador de *cluster* (*cluster controller* – cc): faz o agendamento e coleta informações da execução das MVs num controlador de nodo específico e gerência instâncias de rede virtual.

3) Controlador de armazenamento (*storage controller* – walrus): 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 MVs e dados de usuários.

4) Controlador de nuvem (*cloud controller* - clc): ponto de entrada na nuvem para usuários e administradores. Ele indaga os gerenciadores de nodos sobre informações acerca de recursos, faz decisões de agendamento de alto nível e as implementa através de requisições aos controladores de cluster.

Através das informações disponibilizadas na configuração da parte administrativa do Eucalyptus, conforme Figura 9, são geradas as credenciais que serão utilizadas para autenticação do usuário. Note-se que algumas informações, quando alteradas, ocasionam a necessidade de se gerar novamente tais credenciais e atualizá-las nos nodos.

Com relação à configuração de rede, na versão Eucalyptus 1.6.1, há quatro modos de configuração de rede, conforme listado abaixo (EUCALYPTUS, 2009):

1) *System*: modo mais simples, com menos recursos de rede. Nesse caso, não é o Eucalyptus que fica responsável pela atribuição de IPs aos nodos, mas um servidor DHCP externo ao Eucalyptus. O Eucalyptus designa um MAC que ele mesmo gera para a MV (normalmente FF:FF:FF:FF:FF:FF), antes do boot, e associa a interface ethernet da máquina virtual à interface física através da ponte (*bridge*) Xen que foi configurada para o nodo local. O endereço IP então é atribuído à essa máquina virtual pelo servidor DHCP da mesma forma que é atribuído a qualquer outra máquina física dessa rede.

2) *Static*: nesse modo, o administrador configura uma lista de pares MAC/IP, que é utilizada pelo Eucalyptus para atribuir às MVs. A parte de associar a interface de rede da máquina virtual à interface física é similar ao modo anterior. A vantagem desse modo seria um maior controle sobre os IPs que se quer associar às MVs.

c) *Managed*: esse é o modo mais completo, oferecendo possibilidade de se isolar as máquinas virtuais em uma rede de máquinas virtuais, de se criar grupos seguros (*security groups*), com regras de ingresso,

assim como definir uma lista de IPs públicos, que os usuários podem utilizar para atribuir às suas MVs durante o boot ou em tempo de execução.

d) *Managed NOV-LAN*: mesmas características do modo gerenciado, mas sem a possibilidade de se isolar a rede de MVs.

Cada um dos modos citados possui características particulares de configuração, configuráveis no arquivo eucalyptus.conf

Após a instalação e configuração inicial, de modo geral, seguindo-se a documentação do Eucalyptus, a instalação fica funcional. A instalação do controlador de nuvem, do controlador de cluster e do controlador de nodo são triviais. A configuração do hipervisor e especialmente da rede é que podem requerer maior atenção, conforme o modo de rede utilizado e o ambiente em que se está configurando. Quando da instalação e configuração, para a versão 1.6.1 utilizada no pré-teste, estava disponível documentação passo a passo para as distribuições OpenSUSE 11, Debian Lenny (5.0.1), Ubuntu Karmic (9.10), Debian "squeeze", Ubuntu Jaunty (9.04) e CentOS 5.

O Eucalyptus permite, através de configuração no arquivo eucalyptus.conf, definir o número de CPUs ou núcleos (core) a serem utilizados no nodo. Na configuração padrão, ele utiliza o máximo de CPUs ou núcleos que encontrar para instanciar as máquinas (no caso do ambiente de testes, como os dois nodos utilizados eram Core 2 duo, seria por padrão possível instanciar duas máquinas virtuais em cada um). Para testes, essa variável foi configurada para MAX_CORES = 4 (número escolhido ao acaso, apenas para permitir instanciar o dobro de máquinas virtuais durante os testes). Ao contrário do que acontece com CPUs ou núcleos, não é possível compartilhar memória entre instâncias, isto é, não é possível utilizar mais memória do que a memória física disponível, mesmo alterando a configuração da variável MAX_MEM, na versão utilizada do Eucalyptus (versão 1.6.1).

Na Tabela 2, são apresentadas as configurações principais utilizadas no eucalyptus.conf, em ambos os nodos, no ambiente de pré-testes apresentado nessa seção.

Tabela 2 - Principais configurações utilizadas no eucalyputs.conf

Variável	**Utilização**
CC_PORT="8774"	Porta na qual o controlador do cluster estará 'ouvindo'.
SCHEDPOLICY= "ROUNDROBIN"	Política de escalonamento de nodos usada pelo controlador do cluster. No *roundrobin*, os nodos são selecionados um após o outro até que seja encontrado um que possa rodar a MV.
NC_PORT="8775"	Porta na qual o controlador do nodo estará “ouvindo”.
HYPERVISOR="xen"	Hipervisor com o qual o controlador do nodo irá interagir para gerenciar as MVs.
# MAX_MEM=2048	Máximo de memória que o Eucalyptus poderá usar do nodo. Se essa opção ficar desabilitada, será utilizada toda a memória disponível.
MAX_CORES="4"	Número máximo de CPUs / núcleos que o Eucalyptus poderá utilizar no nodo. Se ficar desabilitado, serão utilizados todos os disponíveis.
INSTANCE_PATH= "/usr/local/eucalyptus/"	Diretório a ser utilizado pelo controlador do nodo para armazenar as instâncias sendo executadas, assim como suas cópias em *cache*. As imagens sendo executadas serão apagadas depois que a instância é terminada, mas as cópias em cache serão mantidas (e removidas pelo algoritmo de substituição de cache LRU – *Least Recent Used*).
VNET_PUBINTERFACE ="eth0" VNET_PRIVINTERFACE ="eth0"	No *front-end*, a interface privada poderia indicar o dispositivo que está conectado à mesma rede que os nodos e a pública, o dispositivo que está conectado à rede pública. Se for o mesmo dispositivo, indica-se a mesma interface para as duas variáveis. No nodo, ambas são configuradas para a interface física anexada à ponte configurada pelo Xen.
VNET_BRIDGE="br0"	Deve ser configurada para o nome da ponte que o Xen configurou.
VNET_MODE= "SYSTEM"	Modo pelo qual o Eucalyptus gerenciará a rede virtual.

Para se utilizar o Eucalyptus dentro de uma arquitetura com diversos usuários e integrando-se com outras ferramentas, de modo que as configurações e ferramentas envolvidas fiquem transparentes para o usuário, as seguintes situações precisam ser tratadas:

- Gerenciamento da configuração de rede: a configuração é feita no eucalyptus.conf e requer conhecimento do ambiente de rede em que vai ser utilizada a infraestrutura;

- Gerenciamento de chaves: no modo de operação padrão, ao se instanciar uma máquina virtual, para que ela esteja acessível via ssh posteriormente, é fornecida uma chave no momento do instanciamento. Com essa chave é possível fazer ssh para a máquina sem ter que prover usuário e senha, evitando assim a necessidade de se configurar usuários na imagem de máquina virtual utilizada. A priori, qualquer elemento de posse dessa chave pode acessar a máquina como *root*.

- Configuração de um ambiente mínimo para listagem de imagens disponíveis e instanciação de máquinas virtuais: o modo mais básico é o da linha de comando, através do uso da ferramenta euca2ools. Para que seja possível se executar comandos de listagem de imagens disponíveis para instanciação, instâncias sendo executadas e se iniciar uma MV é preciso ter a ferramenta instalada, além de variáveis de ambiente, conforme mostrado no Quadro 1.

```
EUCA_KEY_DIR=$(dirname $(readlink –f ${BASH_SOURCE}))
export S3_URL=http://150.162.63.25:8773/services/Walrus
export EC2_URL=http://150.162.63.25:8773/services/Eucalyptus
export EC2_PRIVATE_KEY=${EUCA_KEY_DIR}/euca2-shirlei-de3342b5-
pk.pem
export EC2_CERT=${EUCA_KEY_DIR}/euca2-shirlei-de3342b5-cert.pem
export EC2_JVM_ARGS=-
Djavax.net.ssl.trustStore=${EUCA_KEY_DIR}/jssecacerts
export EUCALYPTUS_CERT=${EUCA_KEY_DIR}/cloud-cert.pem
export EC2_ACCESS_KEY='
export EC2_SECRET_KEY='
# This is a bogus value; Eucalyptus does not need this but client tools do.
export EC2_USER_ID='xxxxxxx'
alias ec2-bundle-image="ec2-bundle-image --cert ${EC2_CERT} --privatekey
${EC2_PRIVATE_KEY} --user xxxxxxx --ec2cert ${EUCALYPTUS_CERT}"
alias ec2-upload-bundle="ec2-upload-bundle -a ${EC2_ACCESS_KEY} -s
${EC2_SECRET_KEY} --url ${S3_URL} --ec2cert ${EUCALYPTUS_CERT}"
```

Quadro 1 - Variáveis de ambiente para o uso do euca2ools

A configuração básica apresentada no Quadro 1 é a configuração feita em sistemas operacionais baseados no Linux, para usar o euca2ools como ferramenta de gerenciamento. Isto é, até a escrita desse trabalho, o euca2ools estava disponível apenas para Linux. Para o Windows e eventualmente outro sistema operacional se poderia configurar alguma das ferramentas apresentadas na Seção 2.4, como por exemplo, o *plug-in* Hybridfox para o navegador Firefox.

6 ARQUITETURA PROPOSTA

Toda a análise das ferramentas de gerenciamento que foram abordadas nesse trabalho, além do ambiente de pré-testes, foi feita observando as características que poderiam causar impacto no monitoramento dos recursos, nas características faltantes e especialmente no que precisaria ser monitorado em uma nuvem privada. Essa análise, além da realização do pré-teste, possibilitou visualizar melhor as tecnologias, ferramentas e processos envolvidos na implantação de uma nuvem privada, generalizando-se uma arquitetura de monitoramento que poderia ser dividida em três camadas, conforme ilustrado na Figura 10.

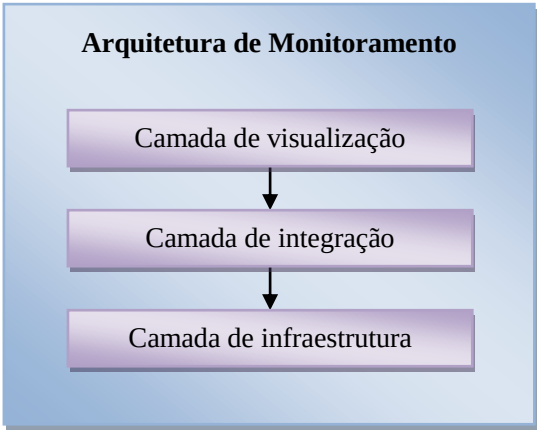


Figura 10 - Camadas da arquitetura de monitoramento para computação em nuvem privada

A seta na arquitetura indica de onde a camada obtém a informação necessária para suas operações, não significando necessariamente que uma não tenha conhecimento da outra.

Outro fator considerado na análise que levou à arquitetura proposta foi o fato que soluções de computação em nuvem geralmente dão mais atenção a um modelo de serviço e às vezes até mesmo em tecnologias ou produtos específicos. No modelo de IaaS há uma forte utilização de tecnologias relacionadas à virtualização, a qual dispõe de diversas ferramentas, como, por exemplo, diferentes hipervisores (Xen, KVM, VMware e outros). Isso significa que soluções de propósito geral são

difíceis de desenvolver, devido à própria natureza da computação em nuvem, e então a orquestração de tecnologias acaba tendo mais atenção do que áreas como monitoramento. Em computação em nuvem privada, por exemplo, as empresas normalmente usam seu próprio *data center* para aproveitar as vantagens da computação em nuvem e ao mesmo tempo se manter dentro de seu próprio *firewall*, por questões de segurança. A arquitetura em três camadas mostrada na Figura 10 foi projetada para atender às necessidades de monitoramento de uma nuvem privada. Entre essas necessidades podem-se citar a integração aos sistemas de monitoramento já utilizados na empresa, como o Nagios, além de ser extensível para atender a novos recursos e ainda ser simples de operar.

Na seqüência cada camada é apresentada em maiores detalhes, observando-se as suas características gerais e também as atividades que serão responsabilidade de cada camada.

A) Camada de visualização/consolidação

Essa camada é a de maior nível hierárquico. Ela serve de interface de gerenciamento de alto nível, através da qual é possível verificar informações como o cumprimento de políticas organizacionais e SLAs. Isso significa que os usuários dessa camada geralmente vão estar interessados, entre outras, na verificação das seguintes informações:

- Imagens de máquinas virtuais disponíveis para instanciamento;
- Níveis de serviço disponíveis e/ou negociáveis para as MVs instanciadas; e
- Dados disponíveis de monitoramento das MVs em execução.

B) Camada de Integração

Num ambiente de computação em nuvem, a camada de Infraestrutura pode ser, e provavelmente será composta por hardware e software heterogêneos, os quais precisarão de uma interface comum de comunicação ou de acesso. O usuário, ao fazer uma solicitação qualquer para a nuvem, geralmente não conhece detalhes da implementação dessa infraestrutura e nem deve ter que se preocupar com as medidas necessárias na hora da utilização do paradigma para que se tenha informações de monitoramento disponíveis posteriormente.

Um cenário típico para exemplificar o exposto acima é de instanciação de uma máquina virtual. A partir do momento em que o usuário solicita a instanciação dessa MV, a instanciação pode ser feita através do hipervisor Xen, do KVM ou outro, dependendo de quais hipervisores estiverem disponíveis na camada de infraestrutura. A decisão de qual

hipervisor utilizar, portanto, se daria de acordo com algum critério ou política de uso previamente definidos. Após essa decisão ser tomada, a solicitação de instanciamento da MV seria passada ao hipervisor escolhido, na camada de infraestrutura. Nesse caso, fica clara a necessidade de se ter uma camada que faça essa ponte entre o que o usuário solicitou eventuais políticas de uso e a camada de infraestrutura, responsável efetiva por colocar em execução o que foi solicitado pelo usuário. Essa política de uso não precisa ser necessariamente algo organizacional e de alto nível, como 'utilizar primeiro o hipervisor cuja licença é paga', podendo ser algo prático do tipo 'passar a solicitação para o primeiro hipervisor disponível'.

C) Camada de Infraestrutura

Essa camada é a última da arquitetura, mas nem por isso menos importante. É nela que se encontram o software e o hardware disponíveis na nuvem (em utilização ou que possam vir a ser utilizados):

- hardware: dispositivos de armazenamento, processamento, de rede, entre outros.

- software: sistema operacional (tanto das máquinas físicas que compõem o hardware citado acima, com as imagens para MVs), aplicativos diversos, licenças, hipervisores, entre outros.

Uma questão importante nessa camada é a granularidade da informação. Isto é, quanto mais detalhes se têm sobre essa camada, maior gama de organização do monitoramento é possível. Por exemplo, na camada de visualização, poderia se organizar a visualização das informações de monitoramento, por equipamento, por cluster, entre outros.

6.1 IMPLEMENTAÇÃO DA ARQUITETURA PROPOSTA

Levando em consideração a falta de soluções genéricas e de código aberto para gerenciamento e monitoramento de uma nuvem privada, além do objetivo de validar a arquitetura proposta, uma solução própria foi desenvolvida, chamada PCMONS (*Private Cloud MOnitoring System* ou Sistema de Monitoramento para Nuvem Privada). Essa solução é um sistema modular e extensível para o monitoramento de nuvens privadas.

Na versão atual, o PCMONS é compatível com o Eucalyptus na camada de infraestrutura e o Nagios na camada de visualização. No

entanto, seu desenvolvimento considera a facilidade de integração com outras ferramentas para computação em nuvem, através do desenvolvimento de extensões. Isto é, não há forte acoplamento ao Eucalyptus ou ao Nagios, os quais são um módulo do sistema, utilizados apenas quando configurados. Além disso, também na versão atual, o software age principalmente na camada de integração, em atividades como coleta e preparo das informações relevantes para a camada de visualização. A-baixo são apresentados os módulos desenvolvidos e as atividades pelas quais são responsáveis dentro do sistema de monitoramento.

A) *Booting VM* (boot da máquina virtual)

Esse módulo é responsável pelos ajustes necessários à máquina virtual que está sendo instanciada, como ajustes no sistema de arquivos, por exemplo, para atender às políticas de monitoramento. A Figura 11 mostra os passos envolvidos durante a instanciação de uma MV, os quais seriam de responsabilidade do módulo *booting VM.*

É importante ressaltar que as políticas de monitoramento aqui visualizadas, poderiam ter as seguintes características:

1) Um conjunto inicial de SLAs pré acordados (e especificados numa linguagem padronizada, como o *WS-Agreement* (OPEN GRID FORUM, 2007) ou outra).

Esse conjunto inicial não tem a intenção de impossibilitar a negociação de SLAs, mas permitir que a organização possa estabelecer um conjunto mínimo de métricas que deva ser monitorado ao se instanciar uma máquina virtual.

Exemplo de uma política que possa ser traduzida para uma métrica constante nesse conjunto inicial:

"Toda instanciação de máquina virtual deve ser feita com injeção de chave e o número de acessos com a chave injetada deve ser monitorado".

2) Um conjunto inicial de SLAs (*templates*) para monitoramento pré-definido.

Esse conjunto inicial não tem a intenção de tornar a negociação de SLAs estática, mas permitir que a organização possa estabelecer um conjunto mínimo de métricas que, apesar de não obrigatórias, poderiam ser monitoradas ao se instanciar uma máquina virtual.

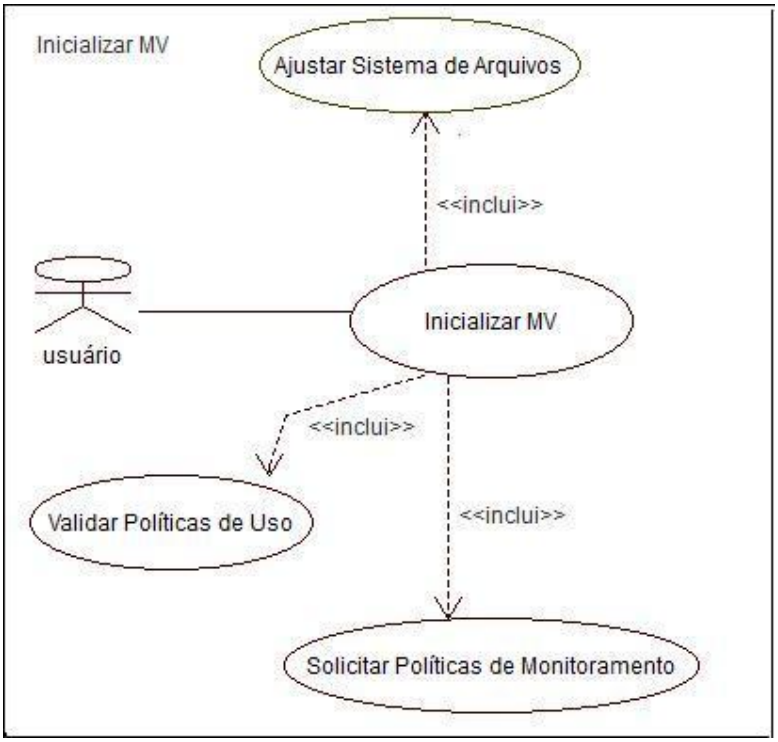


Figura 11 - Passos envolvidos na instanciação de uma MV

B) *Running VMs* (Máquinas Virtuais em Execução)

No módulo anterior, as ações tomadas situavam-se num contexto em que a máquina virtual ainda não estava executando, mas sendo preparada para execução. No módulo *running vms*, são executadas as operações de monitoramento propriamente ditas. Para prover um monitoramento mínimo, o protótipo monitora as seguintes métricas:

a) Ping
b) SSH
c) Memória
d) Carga do sistema
e) Conexões HTTP

A métrica conexões HTTP especificamente é utilizada para o estudo de caso utilizado, apresentado na Seção 5.4. O sensor ou *plug-in* adicionado na MV apresenta um comportamento bastante simples, como pode ser visto no trecho de código apresentado no Quadro 2. São coleta-

das as informações de monitoramento do *plug-in* responsável por cada métrica, agrupadas e enviadas para um servidor.

```
def send_monitoring_data(self,data):
        '''
        connect to a server to send monitoring data
        '''
    try:
        logging.debug('Connecting to Monitoring Server...')
        server = xmlrpclib.ServerProxy('%s:%s'%(settings.SERVER, set-
tings.SERVER_PORT))
        sent = server.get_vm_notification(data)
    except (socket_error, xmlrpclib.Fault, xmlrpclib.ProtocolError,
xmlrpclib.ResponseError), error_code:
        print error_code
       logging.error('Err: (%s)'%error_code)
```

Quadro 2 - Trecho de código responsável por enviar dados de monitoramento a um servidor

Um *plug-in* de monitoramento de uma métrica específica, isto é, a representação de qualquer elemento monitorado, retorna uma lista de chaves e valores, conforme recomendação do Open Grid Forum (2010).

```
  def get_meminfo(self):
    lines = self.parseSplitFile('/proc/meminfo')
    for line in lines:
      if re.match('MemFree',line[0]):
        self.memory['free'] = int(line[1])
      elif re.match('MemTotal', line[0]):
        self.memory['total'] = int(line[1])
      elif re.match('Buffer|Cached', line[0]):
        self.memory['cache'] = int(self.memory['cache']) + int(line[1])
      else:
        pass
    self.memory['used'] = self.memory['total'] - self.memory['free']
    return self.memory

  def parseSplitFile(self,filename):
    f = open(filename, "rb")
    lines = f.readlines()
    lines = map(lambda x: x.strip().split(), lines)
    return lines
```

Quadro 3 - Trecho de código do *plug-in* de monitoramento de memória

No caso do *plug-in* de monitoramento do elemento memória, cujo trecho de código é mostrado no Quadro 3, um exemplo de retorno da informação do estado desse elemento em um dado momento do tempo seria (em notação da linguagem Python):

memory = {'total': 131284, 'cache': 95588, 'used': 127352, 'free': 3932}

A implantação efetiva dos *plug-ins* de monitoramento nas MVs é feita durante o *boot* do sistema operacional, através de um script de inicialização. Esse script pode ser verificado no APÊNDICE BCE E.

C) Interface

Na versão 0.1 do PCMONS, para a interface, foi implementado um módulo para a versão 3.0.6 do Nagios.

O Nagios tem características próprias de monitoramento e apresentação dos dados resultantes e, portanto uma solução híbrida, que combina monitoramento passivo e ativo foi orquestrada.

Para o monitoramento passivo, sensores ou *plug-ins* são adicionados na MV pelo módulo *booting_vms* e o envio da informação do estado dos elementos monitorados é feito conforme descrito no módulo *running_vms* da seção anterior. Quando o servidor de notificações recebe as informações de monitoramento desses sensores, como no exemplo mostrado no Quadro 1, ele toma as ações cabíveis, que podem ser armazenar essas informações em uma base de dados e enviá-las para o aplicativo responsável por mostrar as informações na interface adequada, no caso, o Nagios. Para o monitoramento ativo, são utilizados *plug-ins* do próprio Nagios.

Para que cada nova MV que entra ou sai da nuvem seja mostrada na interface do Nagios, um script de geração de configuração para o Nagios é rodado na *cron* (ver Quadro 4). Se o arquivo de configuração das MVs tiver sofrido alguma alteração, é feito *reload* do serviço do Nagios. Um exemplo de um arquivo de configuração válido em um determinado momento é o do APÊNDICE C – Arquivo de Configuração de MVs para o Nagios.

```
def create_nagios_conf_running_vms(self):
   vms = self.db.get_info_for_monitoring_running_Vms()
      if len(vms) > 0:
         try:
            fh                                                    =
open(running_vms.cluster.cluster_config.NAGIOS_TEMP_CONF,'w')
            try:
               tree =
ET.parse('/opt/pcmons/interface/nagios/cron/basic_monitoring.xml')
            except IOError as (errno, strerror):
               print 'Problems while reading basic_monitoring.xml'
               print "I/O error ({0}): {1}".format(errno,strerror)
            for vm in vms:
               #Check if the machine has a valide ip address
               if vm['public_dns_name'] != '0.0.0.0':
                  hostname =
"%s_%s_%s"%(vm['user'],vm['instance_id'],vm['node_hostname'])
                  self.hostnames.append(hostname)
                  alias                    =                    "%s/%s(%s-
%s)"%(vm['user'],vm['instance_id'],vm['node_hostname'],vm['node_ip'])
                  if vm['node_ip'] != '':
                     vm_host = [vm['node_ip'],hostname]
                     self.nodes_group.append(vm_host)
                  host_def       =       self.define_host(hostname,       alias,
vm['public_dns_name'])
                  fh.write(host_def)
                  service = tree.findall('services/service')
                  for s in service:
                     child = s.getchildren()
                     services = self.define_passive_service(hostname, child)
                     fh.write(services)
            hostgroup = self.define_hostgroup()
            fh.write(hostgroup)
            fh.close()
         except IOError as (errno, strerror):
            print "I/O error ({0}): {1}".format(errno,strerror)
```

Quadro 4 - Trecho de código do script da cron responsável por verificar se MVs entraram ou saíram da nuvem e gerar novo arquivo de configuração para o Nagios

O arquivo basic_monitoring.xml referenciado no Quadro 4 é o arquivo que contém as métricas monitoradas a serem exibidas na interface do Nagios. Ele pode ser visto em maiores detalhes no APÊNDICE D.

A Tabela 3 contém um apanhado geral de cada um dos módulos desenvolvidos e seus principais artefatos.

Tabela 3 - Módulos desenvolvidos do PCMONS e principais componentes

Nome do Módulo	***Camada***	**Descrição**
Booting_VM	*Integração*	Faz os ajustes necessários no sistema de arquivos das MVs, para instalação e configuração básica e especializada de monitoramento, antes do *boot* da MV.
Principais Componentes		
	VM_Monitoring_Node_Plugin.py	Obtém informação sobre as MVs gerenciadas no nodo.
	VM_Monitoring_Cluster_Plugin.py	Controla o processo de obtenção de informação da plataforma de software de IaaS e cada um de seus nodos.
Running_VM	*Integração*	Executa o monitoramento durante a execução da MV.
Principais Componentes		
	eucalyptus_custom.pl	É uma extensão do Eucalyptus. Verifica qual sistema operacional está contido em uma dada imagem e executa as configurações necessárias no nível de execução (*run level*) apropriado (principalmente cópia de arquivos de inicia-

		lização).
	startup.sh	Instala e configure arquivos de monitoramento durante o processo de boot da MV.
	Notification_Server.py	Recebe as notificações de monitoramento das MVs em execução.
Nagios	*Interface*	
Principais Componentes		
	Nagios_Passive_Check.py	Processa as informações de monitoramento, de acordo com o padrão usado pelo Nagios.
	send_passive_check_nagios.sh	Usado pelo Nagios_Passive_Check para escrever mensagens de comando para o *pipe* de comandos do Nagios.
	Generate_VMs_Nagios_Conf.py	Gera o arquivo de configuração das MVs para o Nagios.

6.2 CONSIDERAÇÕES DE DESENVOLVIMENTO E IMPLEMENTAÇÃO

Abaixo são apresentadas diversas questões de projeto que podem ser consideradas durante a construção de um sistema de monitoramento de grade (Mei Yiduo, 2007) e que também podem ser levadas em consideração no desenvolvimento de um sistema para monitoramento de uma nuvem privada.

- Granularidade adequada para o processamento de dados

Tanto monitoramento em tempo real como processamento de registros ou *logs*, com finalidade de histórico, são necessários. O primeiro

é necessário para se saber o estado de um recurso no presente e o último, para questões de planejamento, melhoria de desempenho, contabilidade e outros. O PCMONS leva em consideração essa questão, já que os dados podem ser armazenados num esquema de banco de dados (atualmente trabalha com MySQL), tanto para propósitos históricos como para visualização em tempo real.

- Baixo overhead (sobrecarga)

O processo de monitoramento como um todo deve ter baixa interferência na desempenho do sistema. No caso do PCMONS, são utilizadas técnicas de monitoramento ativo e passivo e a própria arquitetura da nuvem pode auxiliar na distribuição de carga, uma vez que ela pode ser composta por clusters e o PCMONS pode ser executado em cada um dos clusters.

- Alta disponibilidade

Em um ambiente de grade (assim como em um ambiente de nuvem), a heterogeneidade é uma característica forte, requerendo uma visualização lógica unificada dos recursos, assim como de seus estados.

Essa é uma das características que levaram a solução apresentada nesse trabalho, o PCMONS, a ser dividido em camadas e em módulos. A heterogeneidade presente na camada de infraestrutura é tratada na camada de integração. Usuários na camada de visualização vêm apenas a visualização lógica dos recursos e seus estados, como mostrado na Figura 14.

- Escalabilidade

A grade (assim como a nuvem) pode ter seu tamanho incrementado para milhares de nodos e o sistema de monitoramento precisa estar apto a atender essa característica. No caso do PCMONS, quando um novo recurso entra na rede (na versão atual, uma nova MV), os sensores necessários para monitoramento já são adicionados durante o processo de inserção do elemento na nuvem.

- Flexibilidade

É necessária uma fácil integração e interoperabilidade com plataformas de grades existentes (assim como plataformas de nuvens existentes). O PCMONS é modular, permitindo a integração e interoperabilidade com uma determinada plataforma através do desenvolvimento de *plug-ins* para a mesma.

- Implantação transparente

Nodos em uma grade entram e saem rapidamente, assim como a implantação de monitoramento em cada nodo consome tempo e é susce-

tível a erros, o que gera a necessidade de uma configuração de monitoramento transparente. Recursos em uma nuvem têm comportamento similar. O PCMONS é projetado para instalar de modo transparente os sensores de monitoramento enquanto o elemento está se juntando à nuvem.

Além das considerações acima, ao se implementar uma solução de software, uma das atividades necessárias é escolher uma linguagem de programação e essa escolha depende de fatores como a familiaridade da equipe de desenvolvimento com a linguagem, assim como qual linguagem poderia atender melhor os requisitos do projeto. Primeiramente, decidiu-se por usar Python como linguagem de programação, mas como o foco foi principalmente no desenvolvimento de módulos para a camada de integração, essa decisão não pôde ser mantida. A camada de integração do PCMONS interage com diferentes softwares (Eucalyptus, Xen, Linux Shell, e outros) e, portanto outras linguagens precisaram ser utilizadas também: Perl (para suporte ao Eucalyptus) e Bash script (especialmente no módulo *booting_vm*, que trabalha com os níveis de execução do sistema operacional. Assim, o sistema como um todo é construído usando-se Python, Perl e Bash script, mas outras linguagens ainda podem ser necessárias durante o processo de expansão para atender a novas ferramentas.

Outra decisão importante ocorre em relação às quais plataformas dar suporte. Para viabilizar a implementação e testes de um protótipo, assim como a liberação de uma primeira versão do sistema, optou-se por dar suporte ao Eucalyptus, plataforma que vinha sendo estudada e utilizada no ambiente de testes, para se aproveitar a experiência adquirida e também pelas características do Eucalyptus já citadas na seção 2.2.1 Eucalyptus. O Nagios na camada de visualização foi escolhido pelos motivos já citados na seção 4.1 Nagios.

Para finalizar, a abordagem produtor/consumidor, assim como na arquitetura GMA apresentada no Capítulo 4, foi utilizada.

6.3 ESTUDO DE CASO E RESULTADOS OBTIDOS

Para realização do estudo de caso, foi implantado um ambiente de nuvem com a infraestrutura de software e hardware apresentada na Tabela 4, no Laboratório de Redes e Gerência (LRG) do Departamento de Informática e Estatística da Universidade Federal de Santa Catarina.

Para exemplificar uma situação prática para uso da estratégia de monitoramento e ferramenta desenvolvida, foram disponibilizadas na nuvem implantada imagens de máquinas virtuais, para que um usuário possa instanciar um servidor web, simulando assim um ambiente de provedor de hospedagem.

Tabela 4 - Descrição do ambiente implantado para o estudo de caso

Hardware	Software	Papel no Sistema
Processador: AMD Phenom(TM) 9650 Quad-Core @ 2.3 GHz Cache do Processador: 2MB L2 Cache Memória RAM: 4GB DDR2 667MHz Disco Rígido (HD) : 750GB Interface Serial ATA II .	OpenSUSE 11.2 Eucalyptus 1.6.1 Xen versão 3.4.1 Nagios versão 3.0.6	Controlador de Cluster (CC) Controlador da Nuvem (CLC) Controlador de Armazenamento (SC) Walrus Controlador de Nó (NC)
Processador: Intel(R) Core (TM)2 Quad Q8200 @ 2.33GHz Cache do Processador: 2MB L2 Cache Memória RAM: 3 GB DDR2 333 MHz Disco Rígido (HD): 320 GB Serial ATA 3.0Gbps	OpenSUSE 11.2 Eucalyptus 1.6.2 Xen versão 3.4.1	Controlador de Nó(NC)

Para verificar as imagens disponíveis, o usuário pode utilizar uma das ferramentas apresentadas na Seção 2.4 – Ferramentas de Gerenciamento, ou então acessar a interface web do controlador de nuvem do Eucalyptus (ver Figura 12). As imagens utilizadas foram baixadas do site do Eucalyptus. Elas fornecem uma instalação mínima de sistema operacional, baseado em Linux, podendo ser então utilizadas e personalizadas para diferentes propósitos, já que se pode instalar a princípio qualquer aplicação disponível para a distribuição Linux que está sendo utilizada.

Para permitir a personalização e instalação de software nas imagens, foi feita uma extensão para o Eucalyptus. Essa extensão foi construída tirando-se vantagem de uma característica do próprio Eucalyptus: a injeção de chave feita (para propósitos posteriores como conexão SSH na MV) durante a preparação da MV para boot. Foi adicionado nesse script que faz a injeção da chave (*add_key.pl*), uma chamada para um

script customizado próprio, conforme pode ser visto no APÊNDICE A (script add_key.pl) e APÊNDICE B (script customizado próprio). Essencialmente, esse script customizado executa as adições necessárias nos níveis de execução (*run levels*) do sistema operacional da imagem, para executar as operações de configuração durante o boot da máquina virtual. Essas configurações são relativas a monitoramento e também à adequação da imagem para o estudo de caso (hospedagem de serviços web).

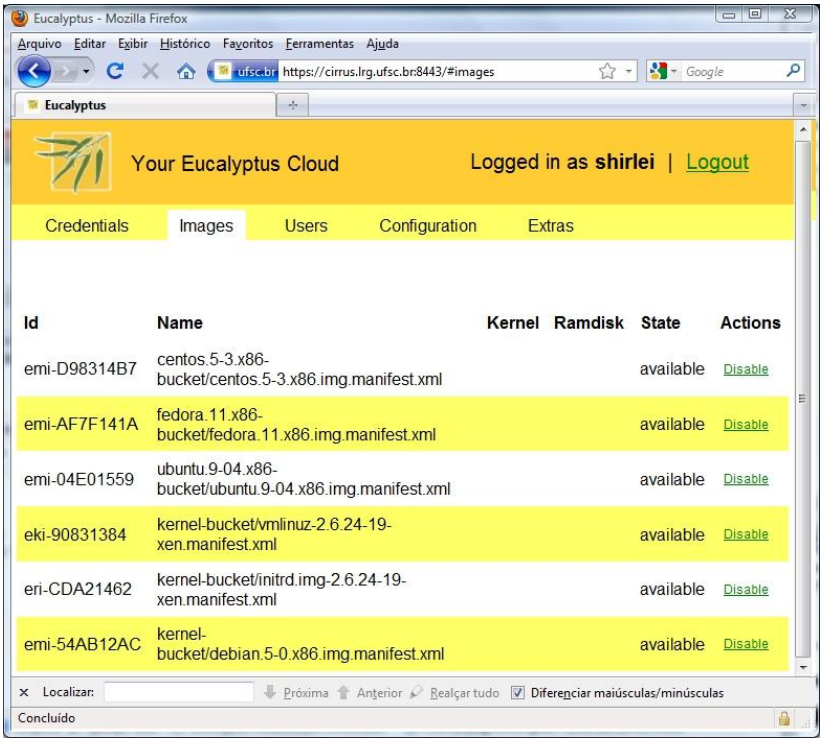


Figura 12 - Imagens virtuais registradas para o estudo de caso

Para prover máquinas virtuais com serviços de hospedagem habilitados, o conjunto de ferramentas Apache, PHP e SQLite é injetado durante o processo de preparação da MV para boot, pelo módulo booting_vm. O script *startup.sh* (ver APÊNDICE E) do módulo *running_vm* está programado para instalar e configurar os pacotes injetados durante esse processo de preparação.

As imagens preparadas para servirem como servidor web foram as do Ubuntu 9.04 e a do CentOS 5. Embora as distribuições Linux compartilhem diversas similaridades, também possuem diversas particularidades, as quais dificultam uma preparação genérica. Uma particularidade, por exemplo, é o tipo de empacotamento de software utilizado. As distribuições do Ubuntu utilizam o formato deb (DEBIAN, 2011), enquanto o CentOS utiliza o rpm (RPM, 2011). Essa diferença influi no modo como os pacotes de softwares a serem instalados são distribuídos e instalados. Outra questão importante é com relação à versão de software e momento de disparar o processo de instalação. Sem um controle da versão de software a ser instalado, a automatização do processo de ajustes da instalação (como porta na qual o servidor web vai responder as requisições) pode se tornar inviável. Se os softwares forem baixados pela rede, a conexão precisa estar disponível, do contrário, o processo de preparo falhará. Para minimizar esses problemas, o módulo que faz o preparo das imagens faz a cópia do software necessário (para evitar problemas de *download*), além de copiar os scripts necessários para o ajuste dos *softwares* instalados, previamente testados.

Além da preparação do software necessário para o servidor web, *plug-ins* de monitoramento para checar utilização de memória e carga de CPU são instalados na MV (injetados pelo módulo booting_vm e instalados pelo running_vm), para enviar dados ao sistema de monitoramento. Outras métricas como tempo de resposta de ping, conexões SSH e HTTP também são monitoradas. O ping e a conexão SSH são verificados de modo ativo, enquanto que memória, CPU, carga e conexões HTTP de modo passivo.

As métricas coletadas provêm um conjunto mínimo de informação 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 na mesma (número de conexões HTTP), que no estudo de caso em questão é o serviço Web. Por se tratar, no entanto, de um estudo de caso no qual se pretendia testar a possibilidade de monitoramento do ambiente configurado e a possibilidade de aplicação prática da arquitetura de monitoramento desenvolvida, esse conjunto inicial de métricas foi escolhido por serem métricas típicas de ambiente Linux, disponíveis em praticamente todas as versões mais novas de seu núcleo.

Abaixo são apresentadas as métricas monitoradas em maiores detalhes e, na Tabela 5, um resumo dessas métricas.

1) Memória

Essa métrica reflete a utilização da memória física e é medida através da leitura do arquivo /proc/meminfo (GAVIN, 1998), disponibilizado pelo núcleo do Linux. No sensor desenvolvido, são coletadas informações sobre a memória disponível, livre e em cache.

2) LoadAvg (Carga Média)

A carga média é utilizada para saber se as CPUs físicas estão sendo usadas além da capacidade ou subutilizadas. No sensor desenvolvido, os valores são lidos de /proc/loadavg, disponibilizado pelo núcleo do Linux. Os valores constantes no /proc/loadavg são produzidos em três intervalos de tempo: 1, 5 e 15 minutos. Segundo Walker (2006), o ponto de utilização perfeita, significando que as CPUs estão sempre ocupadas e, mesmo assim, nenhum processo nunca espera pela CPU, é a média batendo com o número de CPUs. Por exemplo, se há 4 CPUs em uma máquina e a carga média para um minuto é 4.0, a máquina tem utilizado seus processadores perfeitamente nos últimos 60 segundos.

3) SSH

O protocolo SSH ou *Secure Shell* provê um modo de se conectar de modo seguro a um sistema. No estudo de caso desse trabalho, o acesso as MVs é feito por SSH. Se por algum motivo a conexão SSH não for possível, o acesso à máquina fica impossibilitado, não permitindo, por exemplo, a instalação de novos aplicativos na MV.

4) Ping

Ping na verdade é o nome do programa, disponível em diversos sistemas operacionais (normalmente com o pacote de ferramentas de rede), utilizado para testar a conectividade da MV. Para essa métrica, a cada intervalo de tempo predeterminado, o programa Ping é executado. Um maior detalhamento sobre esse programa pode ser visto em Eyler (2005).

5) Número de conexões HTTP

Monitorar o número de conexões HTTP em um servidor pode ser útil por diversos motivos, entre eles planejamento de escalabilidade como também detecção de ataques do tipo DoS (*Denial of Service* ou Negação de Serviço ou DDoS (*Distributed Denial of Service* ou Negação de Serviço Distribuída).

Tabela 5 - Resumo das métricas coletadas.

Métrica	**Motivo**
Memória	Verificar utilização da memória
Carga do Sistema	Verificar carga do sistema
SSH	Verificar disponibilidade de acesso à MV
Ping	Verificar alcançabilidade da MV
Número de conexões HTTP	Verificar possibilidade de usuários não conseguirem acesso (servidor ocupado devido a número de conexões muito alto).

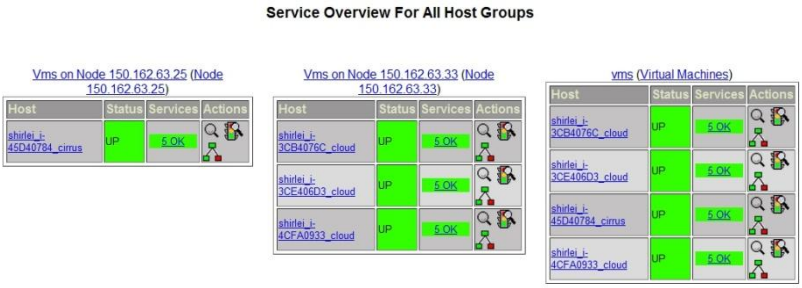
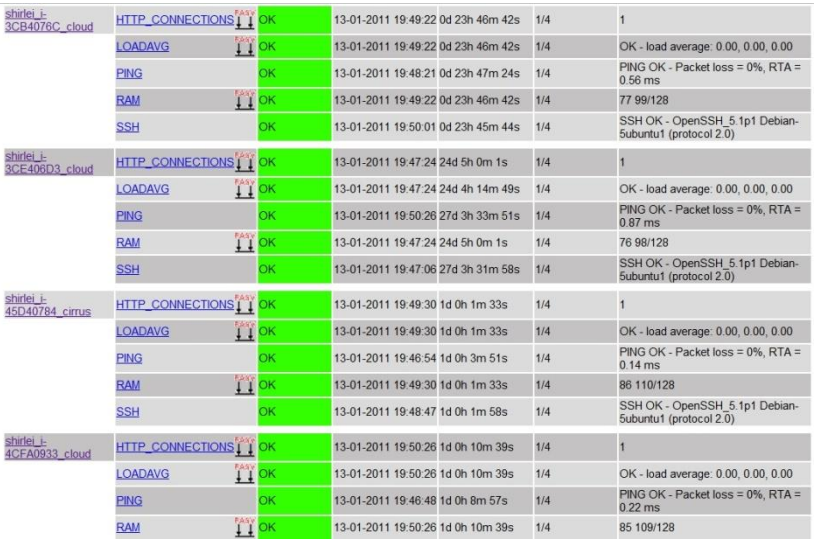


Figura 13 - MVs organizadas por *hostgroups* na interface do Nagios



shirlei_i-3CB4076C_cloud	HTTP_CONNECTIONS	OK	13-01-2011 19:49:22	0d 23h 46m 42s	1/4	1
	LOADAVG	OK	13-01-2011 19:49:22	0d 23h 46m 42s	1/4	OK - load average: 0.00, 0.00, 0.00
	PING	OK	13-01-2011 19:48:21	0d 23h 47m 24s	1/4	PING OK - Packet loss = 0%, RTA = 0.56 ms
	RAM	OK	13-01-2011 19:49:22	0d 23h 46m 42s	1/4	77 99/128
	SSH	OK	13-01-2011 19:50:01	0d 23h 45m 44s	1/4	SSH OK - OpenSSH_5.1p1 Debian-5ubuntu1 (protocol 2.0)
shirlei_i-3CE406D3_cloud	HTTP_CONNECTIONS	OK	13-01-2011 19:47:24	24d 5h 0m 1s	1/4	1
	LOADAVG	OK	13-01-2011 19:47:24	24d 4h 14m 49s	1/4	OK - load average: 0.00, 0.00, 0.00
	PING	OK	13-01-2011 19:50:26	27d 3h 33m 51s	1/4	PING OK - Packet loss = 0%, RTA = 0.87 ms
	RAM	OK	13-01-2011 19:47:24	24d 5h 0m 1s	1/4	76 98/128
	SSH	OK	13-01-2011 19:47:06	27d 3h 31m 58s	1/4	SSH OK - OpenSSH_5.1p1 Debian-5ubuntu1 (protocol 2.0)
shirlei_i-45D40784_cirrus	HTTP_CONNECTIONS	OK	13-01-2011 19:49:30	1d 0h 1m 33s	1/4	1
	LOADAVG	OK	13-01-2011 19:49:30	1d 0h 1m 33s	1/4	OK - load average: 0.00, 0.00, 0.00
	PING	OK	13-01-2011 19:46:54	1d 0h 3m 51s	1/4	PING OK - Packet loss = 0%, RTA = 0.14 ms
	RAM	OK	13-01-2011 19:49:30	1d 0h 1m 33s	1/4	86 110/128
	SSH	OK	13-01-2011 19:48:47	1d 0h 1m 58s	1/4	SSH OK - OpenSSH_5.1p1 Debian-5ubuntu1 (protocol 2.0)
shirlei_i-4CFA0933_cloud	HTTP_CONNECTIONS	OK	13-01-2011 19:50:26	1d 0h 10m 39s	1/4	1
	LOADAVG	OK	13-01-2011 19:50:26	1d 0h 10m 39s	1/4	OK - load average: 0.00, 0.00, 0.00
	PING	OK	13-01-2011 19:46:48	1d 0h 8m 57s	1/4	PING OK - Packet loss = 0%, RTA = 0.22 ms
	RAM	OK	13-01-2011 19:50:26	1d 0h 10m 39s	1/4	85 109/128

Figura 14 - Interface do Nagios para os serviços monitorados

Na Figura 13 e na Figura 14 são mostradas capturas de tela de um momento no tempo em que máquinas virtuais estão sendo executadas de acordo com o estudo de caso descrito. Nessas figuras é possível visualizar:

1) As métricas monitoradas (carga da CPU, número de conexões HTTP e outras), exibidas no padrão de visualização do Nagios.
2) A nomenclatura especial utilizada para o *hostname* das máquinas virtuais, com o intuito de ser mais informativo. Esse nome consiste na junção do nome do usuário que instanciou a máquina, com o id da máquina virtual fornecido pelo sistema, mais o nome da máquina física onde essa MV está rodando. Logicamente, se essas máquinas fossem utilizadas num ambiente com uma configuração DNS disponível e com a necessidade de uma nomenclatura diversa, essa configuração poderia ser ajustada.
3) É possível ver também duas outras personalizações efetuadas durante a criação do arquivo de configuração do Nagios, pelo módulo *interface* do PCMONS:

a. A organização das máquinas virtuais num *hostgroup* próprio (o *Virtual Machines*)
b. A organização das máquinas virtuais por nodos (máquinas físicas). Esse mapeamento de em quais máquinas físicas estão localizadas as máquinas virtuais é uma característica importante do PCMONS, pois essa informação é de extrema importância para a correta detecção e correção de falhas nessas máquinas virtuais.

Para testar o monitoramento da métrica número de conexões HTTP, referente ao estudo de caso (instanciação de um servidor web), foi elaborado um procedimento de testes usando o JMeter. O Apache JMeter é uma aplicativo de desktop 100% Java, de código aberto, projetado para executar testes de comportamento funcional e medir performance (APACHE SOFTWARE FOUNDATION, 2010).

Para realizar os testes, é necessário criar um Plano de Testes no formato do JMeter, incluindo os elementos de teste. No caso do presente estudo, foi criado um plano de testes englobando todas as máquinas virtuais sendo executadas no momento dos testes, conforme pode ser visto na Figura 15.

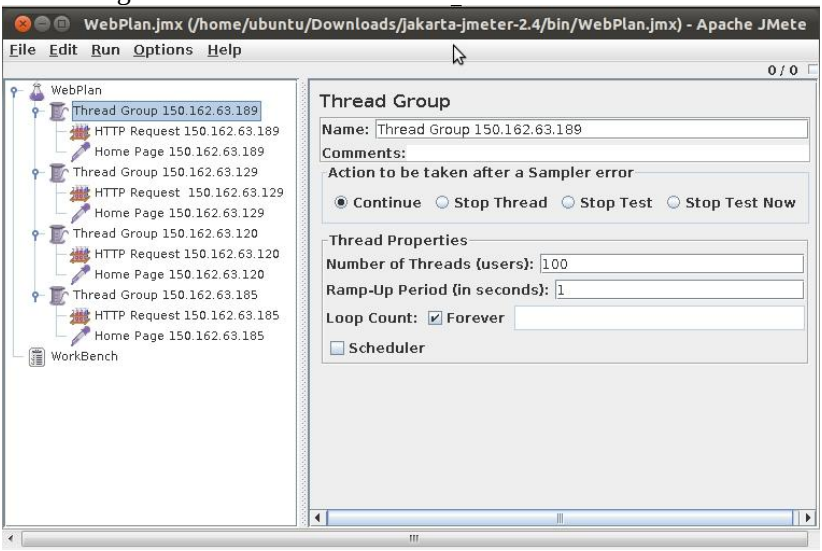


Figura 15 – Plano de testes para conexões HTTP nas máquinas virtuais em execução.

Os elementos presentes na Figura 15 são detalhados a seguir, salientando que a nomenclatura que aparece na coluna da esquerda da figura foi feita manualmente durante a criação de cada tipo de elemento do teste, isto é, ela é editável.

WebPlan – Esse é o elemento principal, que representa o plano de testes.
Thread Group – Este elemento controla as threads que serão executadas pelo teste e os demais elementos do plano de testes devem estar sob esse elemento.
HTTP Request – Esse elemento faz parte do grupo elementos de configuração. Ele não faz as requisições diretamente, mas pode modificá-las.
Home Page – Trata-se de um elemento de amostragem, o qual faz requisições para a página inicial do host em questão.

Na primeira execução do plano de testes foram configuradas 100, 500 e 1000 *threads* simultâneas, com um intervalo de um segundo entre cada thread. Foi configurado também para que executasse em um loop eterno, porém o teste foi executado durante 20 minutos e então finalizado. Os resultados do monitoramento exibidos no Nagios são mostrados na Figura 16, Figura 17, Figura 18.

shirlei_i-3CB4076C_cloud	HTTP_CONNECTIONS	OK	12-01-2011 20:49:04	0d 0h 48m 24s	1/4	191
	LOADAVG	OK	12-01-2011 20:49:04	0d 0h 48m 24s	1/4	OK - load average: 0.01, 0.02, 0.00
	PING	OK	12-01-2011 20:48:21	0d 0h 49m 6s	1/4	PING OK - Packet loss = 0%, RTA = 0.19 ms
	RAM	OK	12-01-2011 20:49:04	0d 0h 48m 24s	1/4	88 113/128
	SSH	OK	12-01-2011 20:50:01	0d 0h 47m 26s	1/4	SSH OK - OpenSSH_5.1p1 Debian-5ubuntu1 (protocol 2.0)
shirlei_i-3CE406D3_cloud	HTTP_CONNECTIONS	OK	12-01-2011 20:52:04	23d 6h 1m 43s	1/4	168
	LOADAVG	OK	12-01-2011 20:52:04	23d 5h 16m 31s	1/4	OK - load average: 0.10, 0.05, 0.01
	PING	OK	12-01-2011 20:50:26	26d 4h 35m 33s	1/4	PING OK - Packet loss = 0%, RTA = 0.90 ms
	RAM	OK	12-01-2011 20:52:04	23d 6h 1m 43s	1/4	95 122/128
	SSH	OK	12-01-2011 20:52:06	26d 4h 33m 40s	1/4	SSH OK - OpenSSH_5.1p1 Debian-5ubuntu1 (protocol 2.0)
shirlei_i-45D40784_cirrus	HTTP_CONNECTIONS	OK	12-01-2011 20:49:13	0d 1h 3m 15s	1/4	185
	LOADAVG	OK	12-01-2011 20:49:13	0d 1h 3m 15s	1/4	OK - load average: 0.01, 0.02, 0.00
	PING	OK	12-01-2011 20:51:54	0d 1h 5m 33s	1/4	PING OK - Packet loss = 0%, RTA = 0.11 ms
	RAM	OK	12-01-2011 20:49:13	0d 1h 3m 15s	1/4	95 122/128
	SSH	OK	12-01-2011 20:48:47	0d 1h 3m 40s	1/4	SSH OK - OpenSSH_5.1p1 Debian-5ubuntu1 (protocol 2.0)
shirlei_i-4CFA0933_cloud	HTTP_CONNECTIONS	OK	12-01-2011 20:50:07	0d 1h 12m 21s	1/4	192
	LOADAVG	OK	12-01-2011 20:50:07	0d 1h 12m 21s	1/4	OK - load average: 0.00, 0.00, 0.00
	PING	OK	12-01-2011 20:51:48	0d 1h 10m 39s	1/4	PING OK - Packet loss = 0%, RTA = 0.29 ms
	RAM	OK	12-01-2011 20:50:07	0d 1h 12m 21s	1/4	95 123/128
	SSH	OK	12-01-2011 20:48:35	0d 1h 8m 52s	1/4	SSH OK - OpenSSH_5.1p1 Debian-5ubuntu1 (protocol 2.0)

Figura 16 – Resultados do Plano de Testes para 100 requisições HTTP simultâneas para cada MV.

shirlei_i-3CB4076C_cloud	HTTP_CONNECTIONS	OK	13-01-2011 21:24:24	1d 1h 25m 25s	1/4	388
	LOADAVG	OK	13-01-2011 21:24:24	1d 1h 25m 25s	1/4	OK - load average: 0.00, 0.00, 0.00
	PING	OK	13-01-2011 21:28:21	1d 1h 26m 7s	1/4	PING OK - Packet loss = 0%, RTA = 0.43 ms
	RAM	OK	13-01-2011 21:24:24	1d 1h 25m 25s	1/4	96 124/128
	SSH	OK	13-01-2011 21:25:01	1d 1h 24m 27s	1/4	SSH OK - OpenSSH_5.1p1 Debian-5ubuntu1 (protocol 2.0)
shirlei_i-3CE406D3_cloud	HTTP_CONNECTIONS	OK	13-01-2011 21:27:25	24d 6h 38m 44s	1/4	471
	LOADAVG	OK	13-01-2011 21:27:25	24d 5h 53m 32s	1/4	OK - load average: 0.00, 0.00, 0.00
	PING	OK	13-01-2011 21:25:26	27d 5h 12m 34s	1/4	PING OK - Packet loss = 0%, RTA = 0.20 ms
	RAM	OK	13-01-2011 21:27:25	24d 6h 38m 44s	1/4	97 125/128
	SSH	OK	13-01-2011 21:27:06	27d 5h 10m 41s	1/4	SSH OK - OpenSSH_5.1p1 Debian-5ubuntu1 (protocol 2.0)
shirlei_i-45D40784_cirrus	HTTP_CONNECTIONS	OK	13-01-2011 21:24:31	1d 1h 40m 16s	1/4	361
	LOADAVG	OK	13-01-2011 21:24:31	1d 1h 40m 16s	1/4	OK - load average: 0.03, 0.01, 0.00
	PING	OK	13-01-2011 21:26:54	1d 1h 42m 34s	1/4	PING OK - Packet loss = 0%, RTA = 0.11 ms
	RAM	OK	13-01-2011 21:24:31	1d 1h 40m 16s	1/4	96 123/128
	SSH	OK	13-01-2011 21:28:47	1d 1h 40m 41s	1/4	SSH OK - OpenSSH_5.1p1 Debian-5ubuntu1 (protocol 2.0)
shirlei_i-4CFA0933_cloud	HTTP_CONNECTIONS	OK	13-01-2011 21:25:27	1d 1h 49m 22s	1/4	353
	LOADAVG	OK	13-01-2011 21:25:27	1d 1h 49m 22s	1/4	OK - load average: 0.00, 0.00, 0.00
	PING	OK	13-01-2011 21:26:48	1d 1h 47m 40s	1/4	PING OK - Packet loss = 0%, RTA = 0.18 ms
	RAM	OK	13-01-2011 21:25:27	1d 1h 49m 22s	1/4	96 124/128
	SSH	OK	13-01-2011 21:28:35	1d 1h 45m 53s	1/4	SSH OK - OpenSSH_5.1p1 Debian-5ubuntu1 (protocol 2.0)

Figura 17 – Resultados do Plano de Testes para 500 requisições HTTP simultâneas para cada MV.

shirlei_i-3CB4076C_cloud	HTTP_CONNECTIONS	OK	13-01-2011 21:59:24	1d 1h 57m 18s	1/4	159
	LOADAVG	OK	13-01-2011 21:59:24	1d 1h 57m 18s	1/4	OK - load average: 0.00, 0.00, 0.00
	PING	OK	13-01-2011 21:58:21	1d 1h 58m 0s	1/4	PING OK - Packet loss = 0%, RTA = 0.45 ms
	RAM	OK	13-01-2011 21:59:24	1d 1h 57m 18s	1/4	83 107/128
	SSH	OK	13-01-2011 22:00:01	1d 1h 56m 20s	1/4	SSH OK - OpenSSH_5.1p1 Debian-5ubuntu1 (protocol 2.0)
shirlei_i-3CE406D3_cloud	HTTP_CONNECTIONS	OK	13-01-2011 21:57:26	24d 7h 10m 37s	1/4	377
	LOADAVG	OK	13-01-2011 21:57:26	24d 6h 25m 25s	1/4	OK - load average: 0.00, 0.00, 0.00
	PING	OK	13-01-2011 22:00:26	27d 5h 44m 27s	1/4	PING OK - Packet loss = 0%, RTA = 0.88 ms
	RAM	OK	13-01-2011 21:57:26	24d 7h 10m 37s	1/4	93 120/128
	SSH	OK	13-01-2011 21:57:06	27d 5h 42m 34s	1/4	SSH OK - OpenSSH_5.1p1 Debian-5ubuntu1 (protocol 2.0)
shirlei_i-45D40784_cirrus	HTTP_CONNECTIONS	OK	13-01-2011 21:59:32	1d 2h 12m 9s	1/4	238
	LOADAVG	OK	13-01-2011 21:59:32	1d 2h 12m 9s	1/4	OK - load average: 0.00, 0.00, 0.00
	PING	OK	13-01-2011 21:56:54	1d 2h 14m 27s	1/4	PING OK - Packet loss = 0%, RTA = 0.12 ms
	RAM	OK	13-01-2011 21:59:32	1d 2h 12m 9s	1/4	92 118/128
	SSH	OK	13-01-2011 21:58:47	1d 2h 12m 34s	1/4	SSH OK - OpenSSH_5.1p1 Debian-5ubuntu1 (protocol 2.0)
shirlei_i-4CFA0933_cloud	HTTP_CONNECTIONS	OK	13-01-2011 22:00:28	1d 2h 21m 15s	1/4	278
	LOADAVG	OK	13-01-2011 22:00:28	1d 2h 21m 15s	1/4	OK - load average: 0.01, 0.01, 0.00
	PING	OK	13-01-2011 21:56:48	1d 2h 19m 33s	1/4	PING OK - Packet loss = 0%, RTA = 0.21 ms
	RAM	OK	13-01-2011 22:00:28	1d 2h 21m 15s	1/4	96 123/128
	SSH	OK	13-01-2011 21:58:35	1d 2h 17m 46s	1/4	SSH OK - OpenSSH_5.1p1 Debian-5ubuntu1 (protocol 2.0)

Figura 18 – Resultados do Plano de Testes para 1000 requisições HTTP simultâneas para cada MV.

Para o plano de testes com 1000 requisições, o JMeter apresentou alguns erros de execução e portanto os resultados foram parecidos com

os apresentados para 500 requisições, conforme pode ser visto na Figura 17 e Figura 18.

6.4 CONCLUSÃO

Este capítulo apresentou o ambiente de testes, o estudo de caso e as principais configurações e possibilidades de uso do PCMONS. Descreveu também os módulos desenvolvidos, suas interações com o ambiente de monitoramento como um todo, além de apresentar as principais idéias e tecnologias por trás de cada módulo. Para que o monitoramento das MVs, assim como a configuração do sistema como um todo fosse possível, se explorou conceitos e tecnologias importantes que habilitam a execução de uma máquina virtual, como sistemas de arquivos, distribuições Linux, linguagens de programação que possibilitam a automatização de rotinas de boot do SO, como a Shell Bash Scripting. Para finalizar, foi executado um plano de testes utilizando a ferramenta JMeter, para testar o comportamento e monitoramento do sistema com diversas requisições HTTP.

7 CONCLUSÃO

O trabalho apresentou e resumiu importantes conceitos de computação em nuvem, além da experiência pessoal da autora com esse novo paradigma. Ele também mostrou que a computação em nuvem não é apenas uma nova moda na área de tecnologia da informação, mas um modo viável de otimizar os recursos computacionais disponíveis em um *data center*. Usando as ferramentas e conceitos disponíveis, foi possível identificar um espaço em aberto para projeto e desenvolvimento de uma arquitetura genérica para o monitoramento de ambientes de computação em nuvem privada que usam o modelo de serviço IaaS, assim como o projeto e desenvolvimento de ferramentas de código aberto para efetivar esse monitoramento, considerando especialmente questões como interoperabilidade e flexibilidade.

Um estudo de caso, buscando reproduzir uma situação do dia a dia de usuários de uma nuvem privada, foi colocado em prática para testar a arquitetura proposta e a ferramenta desenvolvida. A proposta de monitoramento utilizada dá bastante enfoque ao papel das imagens de sistema operacional utilizadas para a instanciação de máquinas virtuais. Isso se deve ao fato já mencionado de elas representarem um papel fundamental na tecnologia de máquinas virtuais e também por representarem toda a pilha de software na qual um sistema será executado. Elas podem, enquanto um sistema dormente representado pelo arquivo da imagem, ser tratadas como um arquivo qualquer. Porém, a partir do momento em que passam a representar o seu papel num ambiente virtualizado e, em última instância, numa nuvem, passam a se comportar como uma espécie de agente, levando consigo informações sobre o seu sistema ‘recém-acordado’ e, possivelmente, sobre as ações que deve tomar no ambiente em que está inserida.

Ao longo do processo, foi possível notar também que a computação em nuvem pode se beneficiar de ferramentas e conceitos já bem estabelecidos para o gerenciamento de computação distribuída. Além do mais, a orquestração de soluções de monitoramento em infraestruturas já instaladas é viável, incluindo o uso de ferramentas bem conhecidas e documentadas, como o Nagios. Por outro lado, mesmo em nuvens privadas, a heterogeneidade dos recursos computacionais pode requerer um esforço extra durante a implementação de uma solução de monitoramento.

7.1 PRINCIPAIS CONTRIBUIÇÕES

Além das contribuições citadas acima, citam-se outras, considerando-se especialmente a conjuntura tecnológica e a contribuição social desse trabalho:

1) Divulgação e contribuição ao desenvolvimento do tema computação em nuvem.

A computação em nuvem despertou inicialmente diversos questionamentos envolvendo principalmente a possibilidade de ser apenas mais um termo que causa alarido (*buzzword*), mas sem representar nada de novo. Aos poucos, foi despertando o interesse acadêmico e nos anos de 2009 e 2010 (períodos em que o presente trabalho foi desenvolvido) surgiram os primeiros workshops e conferências internacionais sobre o tema, fazendo com que o LRG e a UFSC fossem pioneiros no desenvolvimento científico do tema no âmbito brasileiro.

2) Alinhamento com a política pública brasileira de "serviços livres para a sociedade do conhecimento".

Ao explorar e demonstrar a possibilidade de se configurar uma nuvem utilizando-se apenas softwares livres e de código aberto, aproveitando-se de recursos de hardware existentes, não necessariamente recursos de ponta ou desenvolvidos especificamente para computação em nuvem ou as tecnologias que ela engloba (como a virtualização), oportuniza que laboratórios acadêmicos ou outras entidades públicas ou sem fins lucrativos possam oferecer serviços de computação em nuvem sem a necessidade de novos investimentos, reutilizando hardware existente na própria instituição.

3) Implantação no LRG de uma nuvem privada, na qual os alunos interessados podem utilizar principalmente o serviço de servidores web executando em máquinas virtuais.

4) Contribuição acadêmica e científica.

Do presente trabalho, além da dissertação em si, resultaram as seguintes atividades acadêmicas e científicas:

- Um trabalho de conclusão de curso de graduação (TCC);
- Três artigos publicados (um evento nacional e dois eventos internacionais);
- Um artigo sendo revisado em revista internacional de design e implementação;

- Contribuição de conteúdo em tutoriais e palestras:
- Grid and Cloud Computing Management and Security (I2TS 2009)
- Grid and Cloud Computing Management and Security (IEEE/IFIP NOMS 2010)
- Management, Security and Green Cloud Computing (Semana de Cursos e Palestras da Computação 2010).

7.2 TRABALHOS FUTUROS

Para monitorar métricas específicas, especialmente de um modo independente de interface, um conjunto prévio de plug-ins de monitoramento precisa ser desenvolvido. Portanto, como trabalho futuro, pretende-se implementar no PCMONS um conjunto prévio maior de métricas e também o suporte a outras ferramentas de plataforma de computação em nuvem e visualização de dados de monitoramento, como o OpenNebula e o Zabbix. Além dessa extensão para suporte a outras ferramentas, facilitar ainda mais a adaptação do PCMONS por terceiros, através do desenvolvimento de documentação consistente e pelo fornecimento de APIs. Esse processo já foi iniciado com a hospedagem do projeto na plataforma Google Code.[11]

Finalmente, baseando-se nos resultados obtidos com essa pesquisa, acredita-se que a computação em nuvem, além de ser um importante paradigma atual, tem um futuro promissor, assim como espaço para muita pesquisa e desenvolvimento.

11 http://code.google.com/p/pcmons/

REFERÊNCIAS

ADAMS, Keith; AGESEN, Ole. A comparison of software and hardware techniques for x86 virtualization. In: INTERNATIONAL CONFERENCE ON ARCHITECTURAL SUPPORT FOR PROGRAMMING LANGUAGES AND OPERATING SYSTEMS, 12., 2006, San Jose, California, EUA. **Proceedings... .** Nova York, Ny, EUA: ACM, 2006. p. 2 - 13.

AMAZON (Org.). **Close Amazon Elastic Compute Cloud (Amazon EC2)**. Disponível em: <http://aws.amazon.com/ec2/>. Acesso em: 20 dez. 2010.

AMD (Org.). **AMD Virtualization**. Disponível em: <http://www.amd.com/br/products/technologies/virtualization/Pages/virtualization.aspx>. Acesso em: 05 janeiro 2011.

APACHE SOFTWARE FOUNDATION (Org.). **Apache JMeter**. Disponível em: <http://jakarta.apache.org/jmeter/>. Acesso em: 10 jan. 2010.

ARMBRUST, Michael et al. A View of Cloud Computing. **Communications Of The ACM,** Nova York, Ny, EUA, v. 53, n. 04, p.50-58, abr. 2010.

BARHAM, Paul et al. Xen and the art of virtualization. In: ACM SYMPOSIUM ON OPERATING SYSTEMS PRINCIPLES, 19., 2003, Bolton Landing, Ny, EUA. **Proceedings... .** Nova York, Ny, EUA: ACM, 2003. p. 164 - 177.

BAUN, Christian; KUNZE, Marcel. Building a private cloud with eucalyptus. In: IEEE INTERNATIONAL CONFERENCE ON E-SCIENCE WORKSHOPS, 5., 2009, Oxford, UK. **Proceedings... .** Washington, DC, EUA: IEEE Computer Society, 2009. p. 33 - 38.

BROCK, Michael; GOSCINSKI, Andrzej. Grids vs. clouds. In: INTERNATIONAL CONFERENCE ON FUTURE INFORMATION TECHNOLOGY, 5., 2010, Busan, Coréia. **Proceedings...** . Washington, DC, EUA: IEEE Computer Society, 2010. p. 1 - 6.

BRODKIN, Jon. **Gartner: seven cloud-computing security risks.** Disponível em: <http://www.infoworld.com/d/security-central/gartner-seven-cloud-computing-security-risks-853?PAGE=0%2C0>. Acesso em: 03 abr. 2010.

BUYYA, Rajkumar; YEO, Chee S.; VENUGOPAL, Srikumar. Market-Oriented Cloud Computing: Vision, Hype, and Reality for Delivering IT Services as Computing Utilities. In: IEEE INTERNATIONAL CONFERENCE ON HIGH PERFORMANCE COMPUTING AND COMMUNICATIONS, 10., 2008, Dalian,China. **Proceedings... .** Los Alamitos, CA, EUA: IEEE, 2008. p. 5 - 13.

CERBELAUD, Damien; GARG, Shishir; HUYLEBROECK, Jeremy. Opening the clouds: qualitative overview of the state-of-the-art open source VM-based cloud management platforms. In: ACM/IFIP/USENIX INTERNATIONAL CONFERENCE ON MIDDLEWARE, 10., 2009, Urbana Champaign, Illinois, EUA. **Proceedings...** . Nova York, NY, EUA: ACM, 2009. p. 1 - 8.

CHEN, Bin et al. Fast, On-Demand Software Deployment with Lightweight, Independent Virtual Disk Image. In: INTERNATIONAL CONFERENCE ON GRID AND COOPERATIVE COMPUTING, 8., 2009, Lanzhou, Gansu, China. **Proceedings...** . Washington, Dc, EUA: Ieee Computer Society, 2009. p. 16 - 23.

CHUNG, Wu-chun; CHANG, Ruay-shiung. A new mechanism for resource monitoring in Grid computing. **Future Generation Computer Systems,** Nova York, NY, EUA, p. 1-7. jan. 2009.

CLOUD SECURITY ALLIANCE (Org.). **Guidance for Critical Areas of Focus in Cloud Computing.** Disponível em: <http://www.cloudsecurityalliance.org/guidance/csaguide.v2.1.pdf.>. Acesso em: 01 nov. 2010.

DANIELS, Jeff. Server virtualization architecture and implementation. **Crossroads**, Nova York, NY, EUA, v. 16, n. 1, p.18-20, set. 2009.

DEBIAN (Org.). **The Debian GNU/Linux FAQ.** Disponível em: <http://www.debian.org/doc/FAQ/ch-pkg_basics.en.html>. Acesso em:

14 jan. 2011.

DMTF (Org.). Cloud Management Standards. 2010a. Disponível em: <http://dmtf.org/standards/cloud>. Acesso em: 08 ago. 2010.

_____. **Architecture for Managing Clouds.** v.1 2010b. Disponível em: <http://www.dmtf.org/sites/default/files/standards/documents/DSP-IS0102_1.0.0.pdf>. Acesso em: 21 out. 2010.

_____. **Use Cases and Interactions for Managing Clouds**. v. 1.0.0, 2010c. Disponível em: <http://www.dmtf.org/sites/default/files/standards/documents/DSP-IS0103_1.0.0.pdf>. Acesso em: 21 out. 2010.

ELMROTH, Erik; LARSSON, Lars. Interfaces for Placement, Migration, and Monitoring of Virtual Machines in Federated Clouds. In: INTERNATIONAL CONFERENCE ON GRID AND COOPERATIVE COMPUTING, 8., 2009, Lanzhou, Gansu, China. Proceedings... . Washington, DC, EUA: IEEE Computer Society, 2009. p. 253 - 260.
EUCALYPTUS (Org.). **The Open Source Cloud Platform.** v. 1.6.1, 2009. Disponível em: <http://open.eucalyptus.com>. Acesso em: 16 nov. 2009.

_____. **Eucalyptus Open-Source Cloud Computing Infrastructure - An Overview.** Disponível em: <http://www.eucalyptus.com/pdf/whitepapers/Cloud_Builder_Guide.pdf >. Acesso em: 16 maio 2010.

EYLER, Pat. **An Overview of ping**. 12 Out. 2005. Disponível em: < http://www.linuxjournal.com/article/8605 >. Acesso em: 04 janeiro 2011.

FLEXISCALE (Org.). **Flexiscale Public Cloud**. Disponível em: <http://www.flexiant.com/products/flexiscale/>. Acesso em: 20 dez. 2010.

FOSTER, Ian et al. Cloud Computing and Grid Computing 360-Degree Compared. In: WORKSHOP IN GRID COMPUTING ENVIRONMENTS, 1., 2008, Austin, Tx , EUA. **Proceedings... .** Los Alamitos, CA, EUA: IEEE, 2008. p. 1 - 10.

GAVIN, David. **Performance Monitoring Tools for Linux**. 30 Nov. 1998. Disponível em: <http://www.linuxjournal.com/article/2396?page=0,1>. Acesso em: 04 janeiro 2011.

GOGRID (Org.). **Cloud Hosting, Cloud Servers, Hybrid Hosting, Cloud Infrastructure from GoGrid**. Disponível em: <http://www.gogrid.com>. Acesso em: 20 dez. 2010.

GOOGLE (Org.). **Google App Engine**. Disponível em: <http://code.google.com/appengine>. Acesso em: 20 dez. 2010.

HABIB, Irfan. Virtualization with KVM. **Linux Journal,** Seattle, WA, EUA, v. 166, fev. 2008.

IMAMAGIC, Emir; RADIC, Branimir; DOBRENIC, Dobrisa. CRO-GRID grid monitoring architecture. In: INTERNATIONAL CONFERENCE ON INFORMATION TECHNOLOGY INTERFACES, 27., 2005, Cavtat, Croácia. **Proceedings... .** Washington, DC, EUA: IEEE Computer Society, 2005. p. 65 - 72.

IMAMAGIC, Emir; DOBRENIC, Dobrisa. Grid infrastructure monitoring system based on Nagios. In: WORKSHOP ON GRID MONITORING, 1., 2007, Monterey, California, EUA. **Proceedings... .** Nova York, NY, EUA: ACM, 2007. p. 23 - 28.

INTEL (Org.). **Intel Cloud Computing Taxonomy and Ecosystem Analysis.** fev. 2010. Disponível em: <download.intel.com/it/pdf/Cloud_Compute_Taxonomy.pdf>. Acesso em: 24 out. 2010.

_____. **Intel® Virtualization Technology (Intel® VT)**. Disponível em: <http://www.intel.com/technology/virtualization/technology.htm>. Acesso em: 05 janeiro 2011.

KANT, Krishna. Data center evolution: A tutorial on state of the art, issues, and challenges. **Computer Networks**, Nova York, NY, EUA, v. 53, n. 17, p.2939-2965, 03 dez. 2009.

KROEKER, Kirk L.. The evolution of virtualization. **Communications Of The ACM**: Being Human in the Digital Age, Nova York, NY, EUA, v. 52, n. 3, p.18-20, mar. 2009.

LEAVITT, Neal. Is Cloud Computing Really Ready for Prime Time? **Computer**, Los Alamitos, CA, EUA, v. 42, n. 1, p.15-20, jan. 2009.

LIM, Harold C. et al. Automated control in cloud computing: challenges and opportunities. In: WORKSHOP ON AUTOMATED CONTROL FOR DATACENTERS AND CLOUDS, 1., 2009, Barcelona, Espanha. **Proceedings... .** Nova York, NY, EUA: ACM, 2009. p. 13 - 18.

LIN, Geng; DASMALCHI, Glenn; ZHU, Jinzy. Cloud Computing and IT as a Service: Opportunities and Challenges. In: IEEE INTERNATIONAL CONFERENCE ON WEB SERVICES, 6., 2008, Beijing, China. **Proceedings... .** Washington, DC, EUA: IEEE Computer Society, 2008. p. 5 - 5.

MARSHALL, Paul; KEAHEY, Kate; FREEMAN, Tim. Elastic Site: Using Clouds to Elastically Extend Site Resources. In: IEEE/ACM INTERNATIONAL CONFERENCE ON CLUSTER, CLOUD AND GRID COMPUTING, 10., 2010, Melbourne, Vic, Australia. **Proceedings... .** Washington, DC, EUA: IEEE Computer Society, 2010. p. 43 - 52.

MEI, Lijun; CHAN, Wing Kwong; TSE, T.h. A Tale of Clouds: Paradigm Comparisons and Some Thoughts on Research Issues. In: 2008 IEEE ASIA-PACIFIC SERVICES COMPUTING CONFERENCE, 3., 2008, Yilan, Taiwan. **Proceedings... .** Los Alamitos, CA, EUA: IEEE, 2008. p. 464 - 469.

MELL, Peter; GRANCE, Tim. **The NIST Definition of cloud computing.** v. 15, 10 jul. 2009. Disponível em: <http://csrc.nist.gov/groups/SNS/cloud-computing/cloud-def-v15.doc>. Acesso em: 16 maio 2010.

MICROSOFT (Org.). **Windows Azure Platform**. Disponível em: <http://www.microsoft.com/windowsazure/>. Acesso em: 20 dez. 2010.

MORENO-VOZMEDIANO, Rafael; MONTERO, Ruben S.; LLO-

RENTE, Ignacio M.. Elastic management of cluster-based services in the cloud. In: WORKSHOP ON AUTOMATED CONTROL FOR DATACENTERS AND CLOUDS, 1., 2009, Barcelona, Espanha. **Proceedings... .** Nova York, NY, EUA: ACM, 2009. p. 19 - 24.

NURMI, Daniel et al. The Eucalyptus Open-source Cloud-computing system. In: IEEE/ACM INTERNATIONAL SYMPOSIUM ON CLUSTER COMPUTING AND THE GRID, 9., 2009, Shanghai, China. **Proceedings... .** Washington, DC, EUA: IEEE Computer Society, 2009. p. 124 - 131.

OLIVEIRA, Diogo N. de et al. A management system for PLC networks using SNMP Protocol. In: IEEE INTERNATIONAL SYMPOSIUM ON POWER LINE COMMUNICATIONS AND ITS APPLICATIONS, 14., 2010, Rio de Janeiro,rj, Brasil. **Proceedings... .** Washington, DC, EUA: IEEE Computer Society, 2010. p. 78 - 83.

OPEN GRID FORUM (Org.). **OGF Document Series.** jan. 2002. Disponível em: <http://www.ggf.org/documents/GFD.7.pdf>. Acesso em: 08 ago. 2010.

_____. **Web Services Agreement Specification.** 14 de Março de 2007. Disponível em: <www.ggf.org/documents/GFD.107.pdf>. Acesso em: 19 dez. 2010.

_____. **Open Cloud Computing Interface - Use cases and requirements for a Cloud API**. v. 12. Disponível em: <http://forge.ogf.org/sf/go/doc15732?nav=1>. Acesso em: 21 out. 2010.

RACKSPACE (Org.). **Dedicated Server, Managed Hosting, Web Hosting by Rackspace Hosting**. Disponível em: <http://www.rackspace.com>. Acesso em: 20 dez. 2010.

RAY, Edward; SCHULTZ, Eugene. Virtualization Security. In: ANNUAL WORKSHOP ON CYBER SECURITY AND INFORMATION INTELLIGENCE RESEARCH, 5., 2009, Knoxville, EUA. **Proceedings... .** Nova York, NY, EUA: ACM, 2009. p. 1 - 5.

RPM (Org.). **RPM.** Disponível em: <http://www.rpm.org/>. Acesso em: 14 jan. 2011.

SALESFORCE (Org.). **Cloud Computing Platform - salesforce.com**. Disponível em: <http://www.salesforce.com/platform/>. Acesso em: 20 dez. 2010.

SANTOS, Ronaldo C. M. Dos; CHARÃO, Andrea S.. Análise Comparativa de Desempenho do Hipervisor Xen: Paravirtualização versus Virtualização Total. In: SIMPÓSIO DE SISTEMAS COMPUTACIONAIS DE ALTO DESEMPENHO, 9., 2008, Campo Grande, Ms. **Anais... .** Porto Alegre, RS: SBC, 2008. p. 169 – 176.

SAP (Org.). **SAP - SAP Business ByDesign: the Best of SAP, On Demand**. Disponível em: <http://www.sap.com/sme/solutions/businessmanagement/businessbydesign/index.epx>. Acesso em: 20 dez. 2010.

STELTE, Björn; HOCHSTATTER, Iris. INagMon Network Monitoring on the iPhone. In: INTERNATIONAL CONFERENCE ON NEXT GENERATION MOBILE APPLICATIONS, SERVICES AND TECHNOLOGIES, 3., 2009, Cardiff, Wales, Uk. **Proceedings... .** Washington, DC, EUA: IEEE Computer Society, 2009. p. 534 - 538.

TAI, Stefan et al. Cloud service engineering. In: ACM/IEEE INTERNATIONAL CONFERENCE ON SOFTWARE ENGINEERING, 32., 2010, Cape Town, África do Sul. **Proceedings... .** Nova York, NY, EUA: ACM, 2010. v. 2, p. 475 - 476.

VELTE, Toby; VELTE, Anthony; ELSENPETER, Robert C.. **Cloud Computing, A Practical Approach.** San Francisco, Ca: Mcgraw-hill Osborne Media, 2009. 352 p.

VOUK, Mladen A.. Cloud computing — Issues, research and implementations. In: INTERNATIONAL CONFERENCE ON INFORMATION TECHNOLOGY INTERFACES, 30., 2008, Croácia. **Proceedings... .** Los Alamitos, CA, EUA: IEEE, 2008. p. 31 - 40.

WALKER, Ray. **Examining Load Average**. 01 Dez. 2006. Disponível em: <http://www.linuxjournal.com/article/9001?page=0,0>. Acesso em: 20 dez. 2010.

WANG, Lizhe et al. Scientific Cloud Computing: Early Definition and Experience. In: IEEE INTERNATIONAL CONFERENCE ON HIGH PERFORMANCE COMPUTING AND COMMUNICATIONS, 10., 2008, Dalian, China. **Proceedings... .** Washington, DC, EUA: IEEE Computer Society, 2008. p. 825 – 830.

WEI, Jinpeng et al. Managing security of virtual machine images in a cloud environment. In: 2009 ACM WORKSHOP ON CLOUD COMPUTING SECURITY, 1., 2009, Chicago, Il, EUA. **Proceedings... .** Nova York, NY, EUA: ACM, 2009. p. 91 - 96.

XEN (Org.). **What is Xen?** Disponível em: <http://www.xen.org/files/Marketing/WhatisXen.pdf>. Acesso em: 07 mar. 2010.

YIDUO, Mei et al. Rapid and Automated Deployment of Monitoring Services in Grid Environments. In: ASIA-PACIFIC CONFERENCE ON SERVICES COMPUTING, 2., 2006, Guangzhou, Guangdong, China. **Proceedings...** . Washington, DC, EUA: IEEE Computer Society, 2006. p. 328 - 335.

ZHANG, Qi; CHENG, Lu; BOUTABA, Raouf. Cloud computing: state-of-the-art and research challenges. **Journal of Internet Services and Applications**, Nova York, EUA, v. 1, no. 1, p. 7-18, maio 2010.

APÊNDICE A - Script de injeção de chave do Eucalyptus, modificado para integração com o PCMONS

```
delete @ENV{qw(IFS CDPATH ENV BASH_ENV)};
$ENV{'PATH'}='/bin:/usr/bin:/sbin:/usr/sbin/';
$MKDIR=untaint(`which mkdir`);
$RMDIR=untaint(`which rmdir`);
$CHOWN=untaint(`which chown`);
$CHMOD=untaint(`which chmod`);
$MKTEMP=untaint(`which mktemp`);
$TUNE2FS=untaint(`which tune2fs`);
$LOSETUP=untaint(`which losetup`);
# check binaries
if (!-x $MKDIR || !-x $RMDIR || !-x $CHOWN || !-x $CHMOD || !-x $MKTEMP ||
!-x $LOSETUP) {
   print STDERR "add_key cannot find all required binaries\n";
   do_exit(1);
}
# check input params
$mounter = untaint(shift @ARGV);
$offset = untaint(shift @ARGV);
$img = untaint(shift @ARGV);
$key = shift @ARGV; # untaint later
$tmpfile = "";
$loopdev = "";
$mounted = 0;
$attached = 0;
if (!-f "$img" || !-x "$mounter") {
   print STDERR "add_key cannot verify inputs: mounter=$mounter img=$img\n";
   do_exit(1);
}
if ($offset eq "") {
   $offset = 0;
}
chomp($tmpfile = untaint(`$MKTEMP -d`));
if (! -d "$tmpfile") {
   print STDERR "no dir: $tmpfile";
   do_exit(1);
}

# find loop dev and attach image to it
for ($i=0; $i<10 && !$attached; $i++) {
   $loopdev=untaint(`$LOSETUP -f`);
   if ($loopdev ne "") {
         if ($offset == 0) {
            $rc = system("$LOSETUP $loopdev $img");
```

```
            } else {
                $rc = system("$LOSETUP -o $offset $loopdev $img");
            }
            if (!$rc) {
                $attached = 1;
            } else {
                system("$LOSETUP -d $loopdev");
            }
    }
}
if (!$attached) {
    print STDERR "cannot attach a loop device\n";
    do_exit(1);
}
if (system("$TUNE2FS -c 0 -i 0 $loopdev >/dev/null 2>&1")) {
    print STDERR "cmd: $TUNE2FS -c 0 -i 0 $loopdev\n";
#    do_exit(1);
}

# without a key, add_key.pl just runs tune2fs
if (not defined($key)) {
    do_exit(0);
}
$key = untaint($key);
if (!-f "$key") {
    print STDERR "add_key cannot verify inputs: key=$key\n";
    do_exit(1);
}
if (system("$mounter mount $loopdev $tmpfile")) {
    print STDERR "cannot mount: $mounter mount $loopdev $tmpfile\n";
    do_exit(1);
}
$mounted = 1;
if ( !-d "$tmpfile/root/.ssh" ) {
    if (system("$MKDIR $tmpfile/root/.ssh")) {
            print STDERR "cmd: $MKDIR $tmpfile/root/.ssh\n";
            do_exit(1);
    }else{
    system("$CHOWN root $tmpfile/root/.ssh");
    }
    system("$CHOWN root $tmpfile/root/.ssh");
    system("$CHMOD 0700 $tmpfile/root/.ssh");
}
if (!open(OFH, ">>$tmpfile/root/.ssh/authorized_keys")) {
    print STDERR "cannot write to: $tmpfile/root/.ssh/authorized_keys\n";
    do_exit(1);
```

```
}
print OFH "\n";
if (!open(FH, "$key")) {
   print STDERR "cannot read from: $key\n";
   do_exit(1);
}
while(<FH>) {
   chomp;
   print OFH "$_\n";
}
close(FH);
close(OFH);
system("$CHOWN root $tmpfile/root/.ssh/authorized_keys");
system("$CHMOD 0600 $tmpfile/root/.ssh/authorized_keys");

if ($mounted && ($tmpfile ne "")) {
   require "/opt/pcmons/booting_vms/eucalyptus_custom.pl";
   $res = do_custom_setup();
}

do_exit(0);

sub do_exit() {
   $e = shift;

   if ($mounted && ($tmpfile ne "")) {
         system("$mounter umount $tmpfile");
   }
   if ($attached && ($loopdev ne "")) {
         system("$LOSETUP -d $loopdev");
   }
   if ($tmpfile ne "") {
         system("$RMDIR $tmpfile");
   }
   exit($e);
}
sub untaint() {
   $str = shift;
   if ($str =~ /^([ &:#-\@\w.]+)$/) {
         $str = $1; #data is now untainted
   } else {
         $str = "";
   }
   return($str);
}
```

APÊNDICE B - Script para preparação das imagens de máquinas virtuais para monitoramento e adaptação ao estudo de caso.

```
#!/usr/bin/perl
$CUSTOM_SETUP_DIR='/lib/custom_setup';
$BOOT_SCRIPTS_DIR='/opt/pcmons/booting_vms';

sub do_custom_setup(){
   $mon = do_monitoring_setup();
   $soft = do_software_setup();
   open(FILE,">$tmpfile/results.txt");
   print FILE "mon: $mon\n";
   print FILE "soft: $soft\n";
   close(FILE);
   return "Custom Setup Done: $mon, $soft";

}

sub do_monitoring_setup(){
   if ( -f "$tmpfile/etc/issue"){
      $res= `cat $tmpfile/etc/issue`;
      if ( $res =~ /CentOS/i ){
         do_centos_mconfig();
      }elsif ( $res =~ /Fedora/i ){
         do_fedora_mconfig();
      }elsif ( $res =~ /Debian/i ){
         do_debian_mconfig();
      }elsif ( $res =~ /Ubuntu/i ){
         do_ubuntu_mconfig();
      }else{
         do_generic_mconfig();
      }
   }else{
      return "no $tmpfile/etc/issue file found!";
   }
}
sub do_generic_mconfig(){
      return "not implemented yet!";
}
sub do_debian_mconfig(){
   return "not implemented yet!"
}
sub do_fedora_mconfig(){
   return "not implemented yet!"
}
sub do_ubuntu_mconfig(){
```

```
   system("/bin/cp $BOOT_SCRIPTS_DIR/Startup.sh
$tmpfile/etc/rc2.d/S98Monitoring.sh");
   system("/bin/cp $BOOT_SCRIPTS_DIR/monitoring.tar
$tmpfile/root/monitoring.tar");
}

sub do_centos_mconfig(){
   # adding monitoring
   # in case the original rc.local has exit 0, remove it
   $rc_local_file="$tmpfile/etc/rc.local";
   if (!open(OFH, ">>$rc_local_file")) {
      return "rc local file not opened!";
   }else{
      open(FILE,"<$rc_local_file");
      @LINES = <FILE>;
      close(FILE);
      open(FILE,">$rc_local_file");
      foreach $LINE (@LINES) {
         if ($LINE =~ /^exit 0$/){
            next;
         }else{
            print FILE "$LINE";
         }
      }
      close(FILE);
    }

   if (!open(OFH, ">>$rc_local_file")) {
      return "rc local file not opened!";
   }else{
      print OFH "\n";
      if (!open(FH, "$BOOT_SCRIPTS_DIR/Startup.sh")) {
         return "cannot read from: $BOOT_SCRIPTS_DIR/Startup.sh";
      }else{
         while(<FH>) {
            chomp;
            print OFH "$_\n";
         }
         close(FH);
      }
      close(OFH);
   }
   system("/bin/cp $BOOT_SCRIPTS_DIR/monitoring.tar
$tmpfile/root/monitoring.tar");
}
sub do_software_setup(){
```

```
    if ( -f "$tmpfile/etc/issue"){
        $res= `cat $tmpfile/etc/issue`;
        if ( $res =~ /CentOS/i ){
            do_centos_sconfig();
        }elsif ( $res =~ /Fedora/i ){
            do_fedora_sconfig();
        }elsif ( $res =~ /Debian/i ){
            do_debian_sconfig();
        }elsif ( $res =~ /Ubuntu/i ){
            do_ubuntu_sconfig();
        }else{
            do_generic_sconfig();
        }
    }else{
        return "no $tmpfile/etc/issue file found!";
    }
}
sub do_generic_sconfig(){
    return "not implemented yet!";
}
sub do_fedora_sconfig(){
    return "not implemented yet!";
}
sub do_debian_sconfig(){
    return "not implemented yet!";
}
sub do_ubuntu_sconfig(){
    if ( !-d "$tmpfile$CUSTOM_SETUP_DIR" ) {
        system("$MKDIR -p $tmpfile$CUSTOM_SETUP_DIR");
    }
    system("/bin/cp $BOOT_SCRIPTS_DIR/ubuntu9.04/ubuntu9.04lamp.sh
$tmpfile$CUSTOM_SETUP_DIR/custom_install.sh");
    system("/bin/cp $BOOT_SCRIPTS_DIR/ubuntu9.04/ubuntu9.04.tar.gz
$tmpfile/root/ubuntu9.04.tar.gz");
}
sub do_centos_sconfig(){
    if ( !-d "$tmpfile$CUSTOM_SETUP_DIR" ) {
        system("$MKDIR -p $tmpfile$CUSTOM_SETUP_DIR");
    }
    system("/bin/cp $BOOT_SCRIPTS_DIR/centos5/centos5lamp.sh
$tmpfile$CUSTOM_SETUP_DIR/custom_install.sh");
    system("/bin/cp $BOOT_SCRIPTS_DIR/centos5/centos5.tar.gz $tmpfile/root");
}

1;
```

APÊNDICE C – Arquivo de Configuração de MVs para o Nagios

```
define host { use linux-server
                address             150.162.63.128
                host_name           shirlei_i-4DAD0913_cloud
                alias               shirlei/i-4DAD0913(cloud-150.162.63.33)
            }
define service {  host_name shirlei_i-4DAD0913_cloud
                service_description         RAM
                max_check_attempts          4
                contact_groups              admins
                notification_options        w,u,c,r,n
                notification_interval       60
                notification_period         24x7
                check_command               check_dummy!2
                active_checks_enabled       0
                passive_checks_enabled              1
 }
define service {  host_name shirlei_i-4DAD0913_cloud
                service_description         HTTP_CONNECTIONS
                max_check_attempts          4
                contact_groups              admins
                notification_options        w,u,c,r,n
                notification_interval       60
                notification_period         24x7
                check_command               check_dummy!2
                active_checks_enabled       0
                passive_checks_enabled              1
 }
define service {  host_name shirlei_i-4DAD0913_cloud
                service_description         LOAD
                max_check_attempts          4
                contact_groups              admins
                notification_options        w,u,c,r,n
                notification_interval       60
                notification_period         24x7
                check_command               check_dummy!2
                active_checks_enabled       0
                passive_checks_enabled              1
 }
define service {  host_name shirlei_i-4DAD0913_cloud
                service_description         SSH
                max_check_attempts          4
                contact_groups              admins
notification_options         w,u,c,r,n
                notification_interval       60
```

```
                notification_period         24x7
                check_command               check_ssh
                active_checks_enabled       1
}
define service {  host_name shirlei_i-4DAD0913_cloud
                service_description         PING
                max_check_attempts          4
                contact_groups              admins
                notification_options        w,u,c,r,n
                notification_interval       60
                notification_period         24x7
                check_command               check_ping!100.0,20%!500.0,60%
                active_checks_enabled       1
}
define host { use linux-server
                address             150.162.63.102
                host_name               shirlei_i-3E6C07A6_cirrus
                alias               shirlei/i-3E6C07A6(cirrus-150.162.63.25)
            }
define service {  host_name shirlei_i-3E6C07A6_cirrus
                service_description         RAM
                max_check_attempts          4
                contact_groups              admins
                notification_options        w,u,c,r,n
                notification_interval       60
                notification_period         24x7
                check_command               check_dummy!2
                active_checks_enabled       0
                passive_checks_enabled                  1
}
define service {  host_name shirlei_i-3E6C07A6_cirrus
                service_description         HTTP_CONNECTIONS
                max_check_attempts          4
                contact_groups              admins
                notification_options        w,u,c,r,n
                notification_interval       60
                notification_period         24x7
                check_command               check_dummy!2
                active_checks_enabled       0
                passive_checks_enabled                  1
}
define service {  host_name shirlei_i-3E6C07A6_cirrus
                service_description         LOAD
                max_check_attempts          4
                contact_groups              admins
                notification_options        w,u,c,r,n
```

```
                notification_interval        60
                notification_period          24x7
                check_command               check_dummy!2
                active_checks_enabled    0
                passive_checks_enabled                  1
 }
define service {  host_name shirlei_i-3E6C07A6_cirrus
                service_description          SSH
                max_check_attempts       4
                contact_groups               admins
                notification_options       w,u,c,r,n
                notification_interval        60
                notification_period          24x7
                check_command               check_ssh
                active_checks_enabled    1
 }
define service {  host_name shirlei_i-3E6C07A6_cirrus
                service_description          PING
                max_check_attempts       4
                contact_groups               admins
                notification_options       w,u,c,r,n
                notification_interval        60

                notification_period          24x7
                check_command               check_ping!100.0,20%!500.0,60%
                active_checks_enabled    1
 }
define hostgroup { hostgroup_name    Virtual Machines
                   alias   vms
                   members shirlei_i-4DAD0913_cloud,shirlei_i-3E6C07A6_cirrus
}
define hostgroup { hostgroup_name Node              150.162.63.33
                     alias   Vms on Node 150.162.63.33
                     members                shirlei_i-4DAD0913_cloud
                    }
define hostgroup { hostgroup_name Node              150.162.63.25
                     alias   Vms on Node 150.162.63.25
                     members                shirlei_i-3E6C07A6_cirrus
                   }
```

APÊNDICE D - Arquivo Basic_monitoring.xml com as métricas de monitoramento a serem exibidas no Nagios

```
<?xml version="1.0" encoding="UTF-8"?>
<basic_monitoring>
<services>
   <service>
      <service_description> RAM </service_description>
      <max_check_attempts> 4</max_check_attempts>
      <contact_groups> admins </contact_groups>
      <notification_options>w,u,c,r,n </notification_options>
      <notification_interval> 60 </notification_interval>
      <notification_period> 24x7</notification_period>
      <check_command>check_dummy!2</check_command>
      <active_checks_enabled>0 </active_checks_enabled>
      <passive_checks_enabled>1 </passive_checks_enabled>
   </service>
   <service>
      <service_description> HTTP_CONNECTIONS </service_description>
      <max_check_attempts> 4</max_check_attempts>
      <contact_groups> admins </contact_groups>
      <notification_options>w,u,c,r,n </notification_options>
      <notification_interval> 60 </notification_interval>
      <notification_period> 24x7</notification_period>
      <check_command>check_dummy!2</check_command>
      <active_checks_enabled>0 </active_checks_enabled>
      <passive_checks_enabled>1 </passive_checks_enabled>
   </service>
   <service>
      <service_description> LOAD </service_description>
      <max_check_attempts> 4</max_check_attempts>
      <contact_groups> admins </contact_groups>
      <notification_options>w,u,c,r,n </notification_options>
      <notification_interval> 60 </notification_interval>
      <notification_period> 24x7</notification_period>
      <check_command>check_dummy!2</check_command>
      <active_checks_enabled>0 </active_checks_enabled>
      <passive_checks_enabled>1 </passive_checks_enabled>
   </service>
   <service>
      <service_description> SSH </service_description>
      <max_check_attempts> 4</max_check_attempts>
      <contact_groups> admins </contact_groups>
      <notification_options>w,u,c,r,n </notification_options>
      <notification_interval> 60 </notification_interval>
```

```
        <notification_period> 24x7</notification_period>
        <check_command>check_ssh</check_command>
        <active_checks_enabled>1 </active_checks_enabled>
    </service>
    <service>
        <service_description> PING </service_description>
        <max_check_attempts> 4</max_check_attempts>
        <contact_groups> admins </contact_groups>
        <notification_options>w,u,c,r,n </notification_options>
        <notification_interval> 60 </notification_interval>
        <notification_period> 24x7</notification_period>
        <check_command>check_ping!100.0,20%!500.0,60%</check_command>
        <active_checks_enabled>1 </active_checks_enabled>
    </service>
</services>
</basic_monitoring>
```

APÊNDICE E - Script Startup.sh, responsável por implantar *plug-ins* de monitoramento nas MVs, durante o boot.

```
#!/bin/sh
#Author : Shirlei Chaves / shirlei@gmail.com
#Date: 2010-07-20
#Description : This script sets up monitoring files
touch /root/install_log.txt
DO_INSTALL=1
if [ -e /lib/custom_setup/custom_install.sh ]; then
   . /lib/custom_setup/custom_install.sh
fi
config_monitor(){
   echo "doing config_monitor" >> /root/install_log.txt
   # Creating/moving necessary files
   if [ ! -d /opt/monitoring ] ; then
      mkdir -p /opt/monitoring
      echo "directory /opt/monitoring created" >> /root/install_log.txt
   fi
   #running the monitor
   if [ -e /root/monitoring.tar ]; then
      tar -xf /root/monitoring.tar -C /opt/monitoring
      python /opt/monitoring/Monitor.py > /dev/null 2>&1 &
      echo "started Monitor.py" >> /root/install_log.txt
   fi
   #removing unecessary files
   rm /root/monitoring.tar
   echo "removed /root/monitoring.tar" >> /root/install_log.txt
}
config_custom_install(){
   echo "doing custom_install" >> /root/install_log.txt
   do_install
   echo "custom_install finished" >> /root/install_log.txt
}
case "$1" in
start)
   config_monitor
   if [ $DO_INSTALL -eq 0 ];then
      config_custom_install
   fi
   ;;
*)
   echo "usage: start"
   exit 1
esac
exit 0
```