UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ
CURSO SUPERIOR DE TECNOLOGIA EM SISTEMAS PARA INTERNET

FABIO ANDRÉ GARALUZ DOS SANTOS

FUNDAMENTOS DA COMPUTAÇÃO EM NUVEM COM ABORDAGEM DAS PLATAFORMAS EUCALYPTUS E OPENNEBULA

TRABALHO DE CONCLUSÃO DE CURSO

CAMPO MOURÃO
2013

FABIO ANDRÉ GARALUZ DOS SANTOS

FUNDAMENTOS DA COMPUTAÇÃO EM NUVEM COM ABORDAGEM DAS PLATAFORMAS EUCALYPTUS E OPENNEBULA

Trabalho de Conclusão de Curso de graduação do Curso Superior de Tecnologia em Sistemas para Internet da Universidade Tecnológica Federal do Paraná – UTFPR, como requisito parcial para obtenção do título de Tecnólogo.

Orientador: Alessandro Kraemer, M.Sc.

CAMPO MOURÃO

2013

ATA DA DEFESA DO TABALHO DE CONCLUSÃO DE CURSO

Às **vinte horas** do dia **vinte e sete de setembro de dois mil e treze** foi realizada no Mini-auditório do EAD da UTFPR-CM a sessão pública da defesa do Trabalho de Conclusão do Curso Superior de Tecnologia em Sistemas para Internet do acadêmico **Fabio André Garaluz dos Santos** com o título **FUNDAMENTOS DA COMPUTAÇÃO EM NUVEM COM ABORDAGEM DAS PLATAFORMAS EUCALYPTUS E OPENNEBULA**. Estavam presentes, além do acadêmico, os membros da banca examinadora composta pelo professor **Me. Alessandro Kraemer** (Orientador-Presidente), pelo professor **Me. Rodrigo Hübner** e pelo professor **Me. Luiz Arthur Feitosa dos Santos**. Inicialmente, o aluno fez a apresentação do seu trabalho, sendo, em seguida, arguido pela banca examinadora. Após as arguições, sem a presença do acadêmico, a banca examinadora o considerou **APROVADO** na disciplina de Trabalho de Conclusão de Curso e atribuiu, em consenso, a nota _____. Este resultado foi comunicado ao acadêmico e aos presentes na sessão pública. A banca examinadora também comunicou ao acadêmico que este resultado fica condicionado à entrega da versão final dentro dos padrões e da documentação exigida pela UTFPR ao professor Responsável do TCC no prazo de **onze dias**. Em seguida foi encerrada a sessão e, para constar, foi lavrada a presente Ata que segue assinada pelos membros da banca examinadora, após lida e considerada conforme.

Observações:

_____

_____

_____

_____

_____

Campo Mourão, 27 de setembro de 2013.

Prof. Me. Rodrigo Hübner Membro	Prof. Me. Luiz Arthur Feitosa dos Santos Membro	Prof. Me. Alessandro Kraemer Orientador

A folha de aprovação assinada encontra-se na coordenação do curso.

RESUMO

SANTOS, Fabio André Garaluz dos. **Fundamentos da Computação em Nuvem com Abordagem das Plataformas Eucalyptus e OpenNebula**. 44 f. Trabalho de Conclusão de Curso (Graduação) – Curso Superior de Tecnologia em Sistemas para Internet. Universidade Tecnológica Federal do Paraná. Campo Mourão, 2013.

A computação em nuvem vem se popularizando nos últimos anos e estando presente na maioria das aplicações *online*. Tudo isso tem sido possível por meio de muitos avanços nas tecnologias de virtualização. O objetivo deste trabalho é elaborar uma abstração de alto nível da arquitetura de nuvem que permite compreender melhor os mecanismos das plataformas. Não encontramos trabalhos similares na literatura que mostre a arquitetura identificada. Identificar a arquitetura é importante para direcionamentos de pesquisa em diversos trabalhos científicos. As análises sobre nuvem e a abstração dos seus elementos tem como base as plataformas Eucalyptus e OpenNebula.

Palavras-chave: Plataformas de Computação em Nuvem. Eucalyptus. OpenNebula.

ABSTRACT

SANTOS, Fabio André Garaluz dos. **Fundamentals of Cloud Computing approaching the Eucalyptus and OpenNebula Platforms**. 44 f. Monograph of Course Conclusion - Technology for Internet Systems. Technological University of Paraná. Campo Mourão, 2013.

Cloud computing is gaining popularity in recent years and becoming useful for most online applications. The goal of this work is to elaborate a high level abstraction about cloud architectures that allows to better understand the platform mechanisms. We not found similar works in literature that show the identified architecture. Identifying the architecture is an important direction for research in several scientific papers. The analysis and abstraction of its elements is based on Eucalyptus and OpenNebula platforms.

Keywords: Cloud Computing Platforms. Eucalyptus. OpenNebula.

LISTA DE ILUSTRAÇÕES

LISTA DE TABELAS

SUMÁRIO

1 INTRODUÇÃO

Desde o surgimento da Internet em ambientes corporativos e domésticos a conectividade dos usuários foi aumentada, dando origem também a uma gama de serviços *online*. Com isso, a necessidade de armazenamento e processamento *online* foi crescendo gradativamente, criando o cenário ideal para o início da computação em nuvem.

A computação em nuvem surgiu recentemente e proporciona a configuração de recursos dinâmicos e flexíveis para atender as demandas computacionais, tanto para armazenamento quanto para processamento dos dados. Alguns anos atrás essa tecnologia estava limitada a grandes empresas e laboratórios, mas com o passar do tempo surgiram versões livres de plataformas de nuvem privada, permitindo que instituições e profissionais criassem sua própria nuvem. Entre diversos exemplos de plataformas, este trabalho apresenta o OpenNebula e o Eucalyptus.

A Computação em Nuvem, segundo Buyya's et al. (2008), é um tipo de sistema distribuído e paralelo que consiste de uma coleção de computadores virtualizados e interconectados. Esses computadores são provisionados dinamicamente e apresentados como um ou mais recursos computacionais em níveis de serviço estabelecidos através de negociação entre o provedor e seus consumidores.

Com o cenário atual transferindo aplicações locais para a nuvem, e a crescente utilização desse paradigma, pesquisadores de várias áreas da tecnologia procuram compreender e melhorar o desempenho dessas plataformas. Dentre as áreas envolvidas, citamos a computação paralela, a engenharia de *software* e de banco de dados.

Neste trabalho realizamos uma pesquisa sobre as plataformas de computação em nuvem analisando os fundamentos teóricos, mecanismos de controle e componentes essenciais. Como resultado dessa pesquisa, elaboramos uma visão arquitetural que auxília na escolha de trilhas pesquisa para trabalhos científicos. Não encontramos na literatura nenhuma abstração de arquitetura que pudesse ser suficientemente clara a esse respeito. Exemplificamos um caso de pesquisa abordando o OpenNebula e o Eucalyptus.

O trabalho está estruturado da seguinte maneira: o Capítulo 2 apresenta conceitos e aplicações da computação em nuvem; o Capítulo 3 mostra quais os principais componentes necessários para o funcionamento das plataformas; no Capítulo 4 mostramos algumas possibi-

lidades de investigação científica tendo como base a visão arquitetural elaborada; e, por fim, o Capítulo 5 relata um caso prático de implantação e observações sobre as plataformas Eucalyptus e OpenNebula.

2 CONCEITOS E APLICAÇÕES DE COMPUTAÇÃO EM NUVEM

A computação em nuvem surgiu como um processo de evolução tecnológica e vem atendendo a demanda de diversas áreas do conhecimento. A evolução de diversos sistemas e redes de computadores com alta escalabilidade contribuiu para o surgimento da computação em nuvem. Para compreender o seu surgimento e o que ela representa é importante observar alguns momentos históricos. A Encyclopedia Britannica (2010) cita a evolução dos serviços de redes de computadores e afirma que nos anos 1960 e 1970 os dados ficavam armazenados em *Mainframes,* acessados por meio de terminais na rede privada. Ou seja, os serviços de *software*, processamento e armazenamento eram centralizados. Nas décadas seguintes houve muitos avanços de desempenho e armazenamento nos computadores clientes, conhecidos comumente como *Desktops*. Passaram-se décadas até que as redes de telecomunicações permitissem a utilização da Internet para disponibilização de serviços na web em larga escala. No fim dos anos 90 surgiram empresas Provedoras de Serviços de Aplicação (ASPs) que tinham o papel de suprir demandas de companhias através da Internet. Com isso, parte da carga de processamento e armazenamento está sendo transferida novamente para um elemento comum. A principal diferença entre o modelo atual e o modelo antigo de processamento e armazenamento é que agora o computador *Desktop* tem a visão de que a nuvem é apenas um elemento (análogo ao *Main Frame*), mas que na realidade esse elemento é formado por muitos outros, como *clusters*, redes de alta velocidade, etc.

Em 1966, Douglas F. Parkhill previu em seu livro "*The Challenge of Computer Utility*" que a indústria da computação viria se assemelhar a um serviço de utilidade pública "em que muitos usuários estariam remotamente ligados em uma central de informática através de links de comunicação" (Encyclopedia Britannica, 2010). Em 2008 o termo computação em nuvem caiu no uso popular e seus serviços vem atendendo desde pessoas com necessidades particulares até grandes corporações. Segundo a Encyclopedia Britannica (2010), a origem desse termo não é clara, mas acabou sendo bem aceita e é citada por importantes pesquisadores no mundo. O termo nuvem representa uma infraestrutura de rede de servidores e *software*s altamente escaláveis e totalmente abstraídos dos usuários finais.

O acesso ao armazenamento de arquivos e aplicações em uma rede local não pode ser considerado um serviço de computação em nuvem. Segundo Soror et al. (2010), o concei-

to de computação em nuvem é aplicado quando centenas de computadores estão interligados em redes locais e, as vezes, pela Internet, por meio de plataformas de gerenciamento que permitem que usuários possam processar arquivos de texto, banco de dados, sítios ou outros serviços de forma transparente em relação a infraestrutura utilizada, abstraindo o armazenamento físico dos dados e seus processamentos. Buyya's et al. (2008) afirmam que a computação em nuvem é um tipo de sistema distribuído e paralelo que consiste de uma coleção de computadores virtualizados e interconectados. Esses computadores são provisionados dinamicamente e apresentados como um ou mais recursos computacionais em níveis de serviço estabelecidos através de negociação entre o provedor e seus consumidores.

A infraestrutura de computação em nuvem pode ser definida como uma rede onde dezenas, centenas ou milhares de computadores (nós) estão interligados e são controlados por um ou mais nós principais. Esse mecanismo é similar ao mecanismo de *cluster*, onde existem nós principais que gerenciam os demais nós. Tendo como base o trabalho de Sousa et al. (2009), na computação em nuvem o nó central gerência recursos como redes virtuais, capacidade de armazenamento, memória RAM e processamento. No cenário de nuvem é possível criar Máquinas Virtuais (MVs) sob demanda a partir de uma imagem de sistema operacional. Essas MVs são distribuídas pelos computadores da nuvem de forma que seus requisitos de rede, memória e processamento sejam atendidos. Para o usuário final, que solicita instâncias de MVs, toda infraestrutura utilizada na nuvem é abstraída pela plataforma gerenciadora.

As aplicações de computação em nuvem vão desde um simples documento armazenado na Internet, que pode ser editado e visualizado por uma aplicação *online* remota e por vários usuários ao mesmo tempo, até empresas que terceirizam toda infraestrutura de servidores armazenando suas aplicações e bancos de dados em redes de terceiros. Os benefícios dessa prática vão desde uma menor preocupação com segurança em alguns aspectos, visto que as empresas que prestam este serviço são bem mais preparadas por serem especialistas nesse tipo de negócio, até a questão do custo da manutenção e da atualização do *hardware*, criando uma abstração onde o usuário pode solicitar a qualquer momento uma maior capacidade de processamento ou armazenamento.

Segundo Sun et al. (2011), a computação em nuvem pode ser dividida em três classes, conforme a abrangência do atendimento dos usuários: Nuvem Pública, Nuvem Privada e Nuvem Híbrida. Por outro lado, Mell et al. (2012) e Sousa et al. (2009) classificam a computação em nuvem em quatro classes: Nuvem Pública, Nuvem Privada, Nuvem Comunidade, e

Nuvem Híbrida. Nuvem pública é uma infraestrutura disponibilizada para o público em geral ou para um aglomerado de indústrias. Essa infraestrutura é de propriedade de uma organização que pode vender ou distribuir gratuitamente os serviços em nuvem. Segundo Armbrust et al. (2009), a nuvem é classificada como pública quando ela é um serviço disponibilizado do tipo pague somente pelo que usar (*pay-as-you-go)*. Como exemplo disso citamos a Amazon Web Services, Google AppEngine e Microsoft Azure. Nuvem Privada pertence a uma determinada organização e é gerenciada pela equipe de TI. A equipe de TI tem o controle total dos recursos, desde a infraestrutura de *hardware* até o *software*. A utilização dos recursos da nuvem é feita pela própria corporação, não disponíveis para uso geral. Nesse modelo de implantação citamos a possibilidade de uma organização utilizar as plataformas Eucalyptus, OpenNebula e OpenStack. A Nuvem Comunidade compartilha uma infraestrutura entre organizações que tem interesses em comum. Esse tipo de nuvem pode ser implantada local ou remotamente e é administrada por uma das organizações ou por terceiros. Por fim, a Nuvem Híbrida, segundo Mell et al. (2012), é a composição de duas ou mais nuvens (privadas ou públicas) unidas por uma tecnologia padronizada ou proprietária que permite a portabilidade de dados e aplicações. Uma Nuvem Comunidade pode fazer parte desse modelo de implantação Sousa et al., (2009). Os serviços oferecidos por essas classes de nuvem são classificados em modelos, apresentados a seguir.

2.1 MODELOS DE SERVIÇO

Segundo Ji et al. (2011), a computação em nuvem pode ser dividida em três modelos de serviços: IaaS (*Infrastructure as a Service*, Infraestrutura como um Serviço), SaaS (*Software as a Service*, *Software* como um Serviço) e PaaS (*Platform as a Service*, Plataforma como um Serviço). O IaaS é a categoria base. Isso significa que as demais são implantadas sob ela, como apresenta a Figura 1. A seguir são apresentadas mais informações sobre esses serviços.

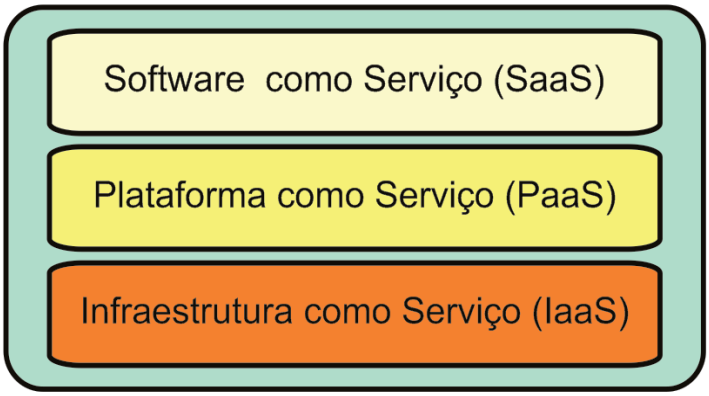

Figura 1 - Modelos de Serviço
Fonte: Sousa et. al (2010)

Segundo Ji et al. (2011), IaaS fornece a estrutura de *hardware* para o funcionamento de uma nuvem através da Internet, oferecendo armazenamento, virtualização de sistemas operacionais, *clusters* e outras configurações. IaaS permite ao usuário, em alguns casos, o gerenciamento completo, que vai desde a escolha e configuração do sistema operacional, até a criação e manutenção de MVs. Os serviços IaaS mais conhecidos mundialmente são Amazon *Elastic Cloud Computing* (*EC2)* e o *Blue Cloud* da *IBM.*

Software como Serviço, ou SaaS, é um serviço direcionado ao usuário final que disponibiliza aplicações *online* que executam em nuvem conforme demanda. As aplicações podem ser acessadas através de um navegador web, ou algum aplicativo cliente específico. Nesse tipo de serviço o usuário não pode escolher a configuração de *hardware*, plataforma ou sistema operacional, apenas algumas configurações específicas do aplicativo estarão disponíveis

para alteração pelo usuário. Um exemplo de SaaS é o *Google Docs*, onde o usuário pode editar e compartilhar arquivos *online* com seus colaboradores. Citamos também o *DropBox, que* gerencia armazenamento e compartilhamento de arquivos.

Na Plataforma como Serviço, ou PaaS, o usuário tem controle apenas de um serviço ou aplicações específicas com acesso somente às configurações. O usuário não têm acesso à infraestrutura, ao sistema operacional ou a configuração da nuvem. Um exemplo de PaaS consiste em hospedagem de sítios, como é o caso do *Google App Engine*. A Tabela 1 apresenta um resumo sobre os modelos de serviço.

Tabela 1 - Tipos de Nuvem e Modelos de Serviço

Modelos de Serviço	Nuvem Privada	Nuvem Pública
SaaS	Portal Corporativo	Google Apps
PaaS	Windows Azure	Google App Engine
IaaS	Eucalyptus, OpenNebula, OpenStack	Amazon Web Services

Fonte: Jha et al. (2011)

Neste trabalho implantamos o modelo de serviço IaaS por meio das plataformas Eucalyptus e OpenNebula. No próximo capítulo apresentamos os principais componentes que formam uma plataforma de computação em nuvem.

3 PRINCIPAIS COMPONENTES QUE FORMAM UMA PLATAFORMA DE COMPUTAÇÃO EM NUVEM

Componentes que garantem o gerenciamento e o funcionamento de uma nuvem podem ser diferentes entre diversas plataformas. Mesmo havendo diferenças entre algumas plataformas de código aberto, podemos destacar alguns componentes em comum, como o c*luster* de máquinas físicas, as formas de virtualização de recursos computacionais, o gerenciador de máquinas virtuais e o gerenciador de recursos da nuvem. Esse último, é um controlador centralizado de toda a plataforma. Detalhamos cada um desses componentes a seguir. Entre as plataformas de computação em nuvem de código aberto abordamos o Eucalyptus e o OpenNebula. A Amazon é um caso de plataforma proprietária onde não temos conhecimento de seus componentes formadores da nuvem. Os mecanismos descobertos sob esses componentes são representados em nossa arquitetura, no Capítulo 4.

3.1 *CLUSTER* DE MÁQUINAS FÍSICAS

Sadashiv et al. (2011) apresentam a computação em *cluster* como uma alternativa aos supercomputadores. Um *cluster* pode ser composto por uma coleção de computadores paralelamente interligados entre si em uma rede de alta velocidade que realizam atividades de processamento comuns. Buyya's et al. (2008) definem *cluster* como um tipo de sistema distribuído e paralelo que consiste em uma coleção de computadores interconectados trabalhando como um único recurso computacional integrado. Alguns cenários, onde algoritmos que necessitam de uma grande capacidade de processamento, seriam inviáveis de serem executados em apenas um simples computador. Para o usuário final a visão é de que existe apenas um computador executando as tarefas. Entre os benefícios da computação em *cluster*, Sadashiv et al. ressaltam a alta escalabilidade, a disponibilidade, o balanceamento de carga e a alta capacidade de processamento. Por conta disso é que as plataformas de nuvem surgiram como uma camada sob os *clusters*.

Um *cluster* pode processar grande quantidade de dados, que através de um nó princi-

pal, divide o processamento desses dados entre os nós escravos. Em alguns algoritmos essa divisão pode melhorar significativamente a velocidade de resolução de problemas computacionais. Bibliotecas de processamento paralelo como o MPICH e o OpenMPI, entre muitas outras, podem ser utilizadas para aproveitar os nós de processamento e memória.

O termo computação em grade (ou *grid*) tem sido tratado como uma modalidade de *cluster*. Buyya et al. (2008) definem a computação em grade como um tipo de sistema paralelo e distribuído que permite compartilhamento e acesso remoto transparente aos recursos. A Grade oportuniza recursos para redução do tempo de execução de aplicações de processamento em larga escala. Outra característica destacada por Buyya et al. é a seleção e a agregação dinâmica de recursos autônomos em tempo de execução que ocorrem de acordo com a disponibilidade. Sadashiv et al. (2011) complementam que a computação em grade funciona como uma combinação de computadores de múltiplos domínios administrativos que tem o objetivo de resolver tarefas. A computação em grade funciona de forma distribuída, pode ser desde uma simples rede de estações de trabalho de uma corporação até uma grande rede de colaboração entre muitas companhias.

3.2 VIRTUALIZAÇÃO DE RECURSOS COMPUTACIONAIS

Um dos principais recursos utilizados pela computação em nuvem é a virtualização, que é a base para o funcionamento dessa tecnologia. Rego (2012) cita em sua dissertação que a virtualização é a criação de uma camada de abstração dos recursos de *hardware* para uma determinada finalidade. Rego divide a virtualização nas seguintes formas:

- Virtualização de Estações de Trabalho: o usuário pode acessar uma máquina virtual através da rede e ter acesso ao sistema operacional e aos aplicativos disponíveis na MV.
- Virtualização de Armazenamento: abstração do armazenamento lógico e do armazenamento físico. Vários discos físicos podem ser convertidos para somente um disco lógico.
- Virtualização de Redes: criação de *switch* virtual. O usuário final pode ter uma rede virtual exclusiva dentro de uma rede física comum.

- Virtualização de Servidores: recurso mais comumente utilizado. Vários servidores podem ser implantados dentro de MVs.

Rego (2012) ainda cita vários benefícios da virtualização, como: redução de custos, facilidade de gerenciamento, facilidade de implantação de servidores, aumento da disponibilidade e facilidade na recuperação de falhas.

Laureano (2006) divide a virtualização em três formas:

- Virtualização de linguagens de programação: nessa categoria citamos a máquina virtual Java. Esse tipo de MV cria uma camada de abstração na comunicação entre o *software* e o sistema operacional, possibilitando assim a portabilidade de uma aplicação para diversos sistemas operacionais.
- Virtualização de sistemas operacionais: trata-se da exportação de um sistema operacional como abstração de um sistema específico. Nesse tipo de virtualização cada MV é vista como um processo nativo do sistema operacional anfitrião. Os discos da MV correspondem aos arquivos do sistema anfitrião e as interfaces são disponibilizadas através de serviços. Exemplo: *User-Mode Linux*.
- Virtualização de *hardware*: trata-se da exportação do sistema físico, ou seja, abstração do *hardware*. Nesse tipo de virtualização o usuário pode criar várias MVs com diferentes sistemas operacionais que compartilham uma mesma máquina física anfitriã. Cada MV tem seu processador, memória, armazenamento e dispositivos que trabalham paralelamente, compartilhando os recursos da máquina física, um exemplo é utilização do VirtualBox para virtualização.

A Figura 2 apresenta a virtualização de *hardware*. Esse tipo de virtualização é amplamente utilizada dentro das plataformas de nuvem. A camada de *hardware* real é formada pelo *cluster* físico. Embora Laureano (2006) destaque nessa figura o *Hypervisor* como um componente acima sistema operacional, em algumas plataformas ele tem acesso direto ao *hardware*.

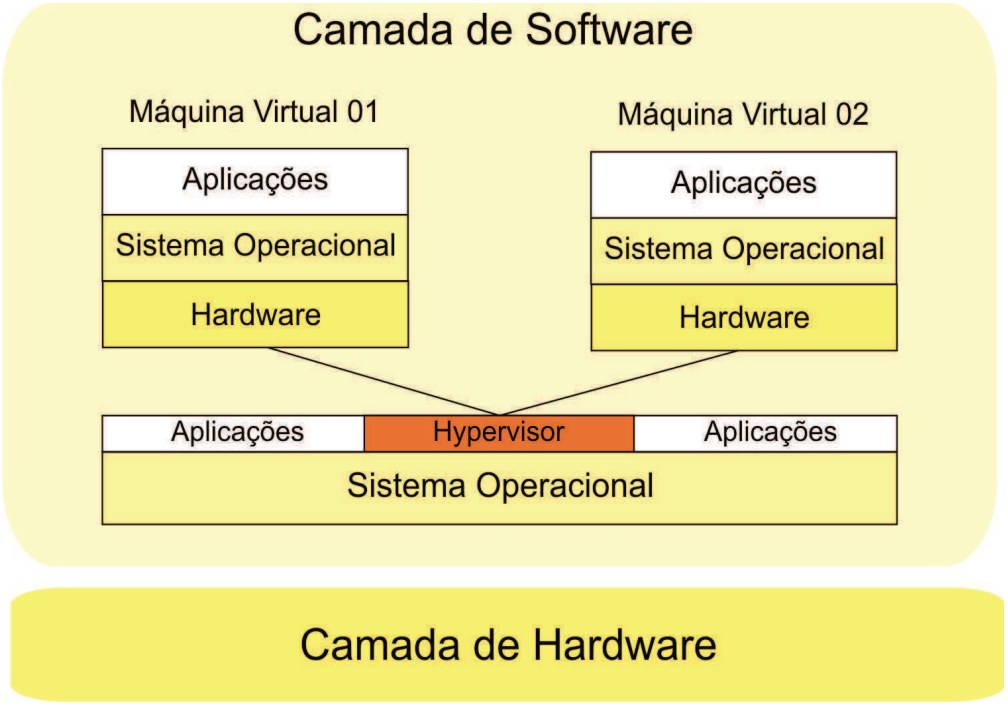

Figura 2 - Virtualização de *Hardware*

Fonte: Laureano (2006)

Laureano (2006) cita uma abordagem da IBM em que a MV pode ser definida como "uma duplicata eficiente e isolada de uma máquina real". Segundo Smith et al. (2005), a virtualização não se limita a apenas um subsistema, como um disco, mas a um computador como um todo. Esse mesmo autor comenta que para implementar uma MV desenvolvedores adicionam uma camada de *software* em um computador real para que suporte a arquitetura desejada. Laureano também cita vários benefícios da utilização de MVs. Entre algumas finalidades podemos citar testes de sistemas, auxílio no ensino prático de sistemas operacionais, execução de sistemas operacionais distintos simultaneamente em um mesmo computador, diminuição de custos com a aquisição de *hardware* e facilidades de gerenciamento.

Para que seja possível a virtualização em nuvem as plataformas incorporam os Gerenciadores de máquinas virtuais, também conhecidos como H*ypervisors.* Entre diversos *Hypervisors* destacamos o KVM e o Xen, que detalhamos nos capítulos a seguir. O KVM é comum em nuvens privadas e o Xen é adotado pela Amazon.

3.3 GERENCIADORES DE MÁQUINAS VIRTUAIS

O *Hypervisor* faz o papel de intermediador entre a MV e a máquina física. Segundo Smith et al.(2005), o *Hypervisor* é responsável por capturar as instruções enviadas pela MV, tratar, traduzir e encaminhar para o *hardware* da máquina física de forma que ele possa entender a instrução. Após o *hardware* retornar a resposta da solicitação, o *Hypervisor* trata novamente as instruções e encaminha a resposta traduzida para a MV.

Existem vários *Hypervisors,* por exemplo, o Xen, o KVM (*Kernel Virtual Machine*), o Virtual Box, o VMWare ESX e ESXi e o Microsoft Hyper-V. Entretanto, na Figura 3 destacamos apenas o Xen, o KVM e o VMWARE. O Xen é utilizado pela Amazon que atualmente é uma das maiores provedoras de nuvem. O KVM é amplamente utilizado em plataformas de código aberto, como o OpenNebula. Por fim, o VMWARE é um *Hypervisor* proprietário amplamente utilizada nas corporações.

Nas arquiteturas dos *Hypervisors* pesquisados são suportadas[1] três tipos de tecnologias de virtualização, conforme indicado na tabela 2, são elas:

- Paravirtualização: Tecnologia introduzida inicialmente pelo *Hypervisor* XEN e adotada posteriormente por outras soluções de virtualização. O sistema convidado deve sofrer alterações para ser executado. Os dispositivos de *hardware* da máquina anfitriã são acessados através dos *drivers* da máquina virtual. Uma característica especial é que permite virtualização em *hardwares* anfitriões que não possuem suporte a Virtualização.
- Virtualização total: O sistema convidado não sofre alterações para ser executado. Oferece-se para o sistema convidado uma cópia do *hardware* da máquina anfitriã. O desempenho da máquina virtual é mais lento em relação a paravirtualização.
- Virtualização de *Hardware* Assistida: O sistema convidado não sofre alterações para ser executado. É um tipo de virtualização total que recebe auxilio da *CPU* anfitriã para executado.

1 http://wiki.xen.org/wiki/Xen_Overview

Tabela 2 - Comparação entre os *Hypervisors*

Características	XEN	KVM	VMWARE vSphere ESXi
Licença	Open Source (GNU GPL v2)	Open Source	Proprietário
Sistema Operacional Anfitrião Suportado	CentOS, Debian Sarge, Fedora, Free BSD, Gentoo, OpenSUSE, SUSE Linux, Ubuntu.	CentOS, Fedora, Red Hat Linux, Ubuntu.	Free BSD, MS DOS
Arquitetura	Bare Metal (Type 1)	Hosted (Type 2)	Bare Metal (Type 1)
Tecnologias de Virtualização Suportadas	Paravirtualização, Virtualização total, Virtualização de Hardware Assistida	Paravirtualização, Virtualização total, Virtualização de Hardware Assistida	Paravirtualização, Virtualização total, Virtualização de Hardware Assistida
Formato de Disco Virtual	raw, qcow2, vhd	cow, qcow, qcow2, qed, vmdk, vpc	vmdk, raw

Fonte: http://virtualization.findthebest.com/compare (09/2013).

Segundo o site oficial[2], o *Hypervisor* KVM (*Kernel Based Virtual Machine*) é uma solução de virtualização de código aberto para os processadores x86_64, PPC 440, PPC 970, S/390, e ARM (Cortex A15). O KVM fornece um módulo kernel, o kvm.ko, e dois módulos de processadores: kvm-intel.ko e kvm-amd.ko. O KVM é disponibilizado através da licença GNU *General Public Licence.* A partir desse *Hypervisor* é possível virtualizar e executar múltiplas MVS que podem conter os sistemas operacionais Linux ou Windows. Esse *Hypervisor* fornece os dispositivos de *hardware* necessários para simular uma máquina física, como placa de vídeo, disco rígido, placa de som, placa de vídeo etc. A Figura 3 apresenta a arquitetura do KVM.

2http://www.linux-kvm.org/page/Main_Page

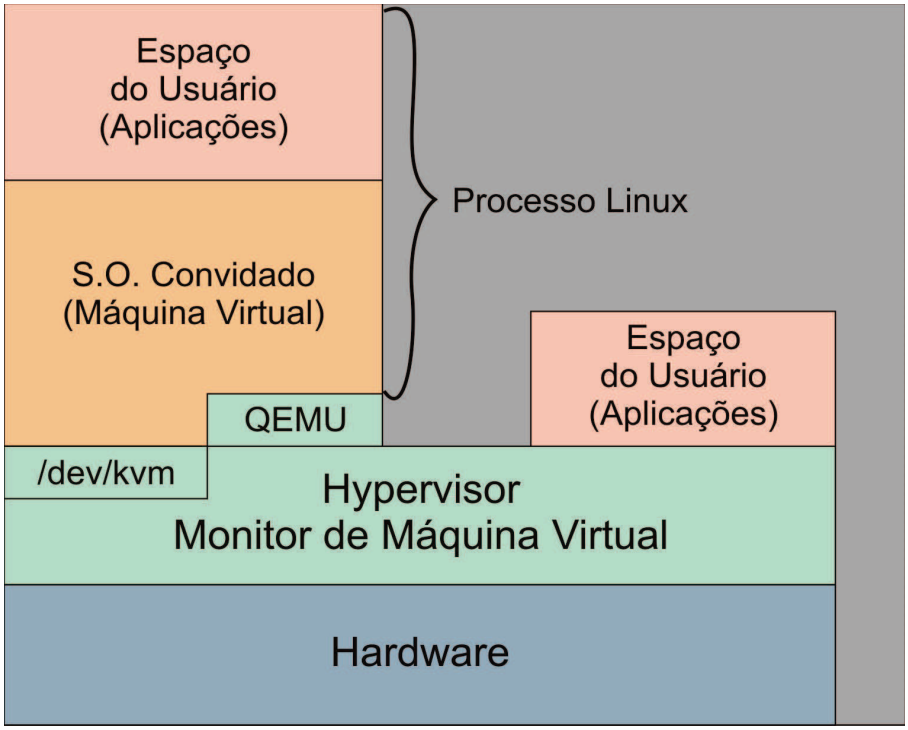

Figura 3 - Arquitetura do Hypervisor KVM

Fonte: http://www.ibm.com/developerworks/linux/library/l-linux-kvm/

O XEN[3] é um monitor de MV desenvolvido pelo Laboratório de Computação da Universidade de Cambridge, disponibilizado através da licença GNU (General Public Licence). O XEN permite instanciar paralelamente várias máquinas virtuais com sistemas operacionais distintos através de uma única máquina física. Esse *Hypervisor* suporta[4] totalmente, na versão 4.3, as arquiteturas x86_64, e parcialmente as arquiteturas ARM v7 e ARM V8. Segundo o sítio xenproject.org[5], o *Hypervisor* XEN está presente no Eucalyptus, no OpenNebula, no OpenStack e na Amazon. O XEN pode ser instalado em dois tipos de arquitetura: o *Bare Metal Hypervisor*, que executa diretamente no *hardware,* proporcionando segurança e confiabilidade; e OS *Hosted*, que executa sob um sistema operacional hospedeiro utilizando seus recursos para virtualização das MVs. Nesse tipo de arquitetura a instalação é simples e não é necessário nenhum dispositivo adicional. As Figuras 4 e 5 apresentam uma visão arquitetural do *Bare Metal* e do *Hosted*.

3 http://www.xenproject.org/about/history.html

4 http://wiki.xenproject.org/wiki/Xen_Release_Features

5 http://www.xenproject.org/developers/teams/hypervisor.html

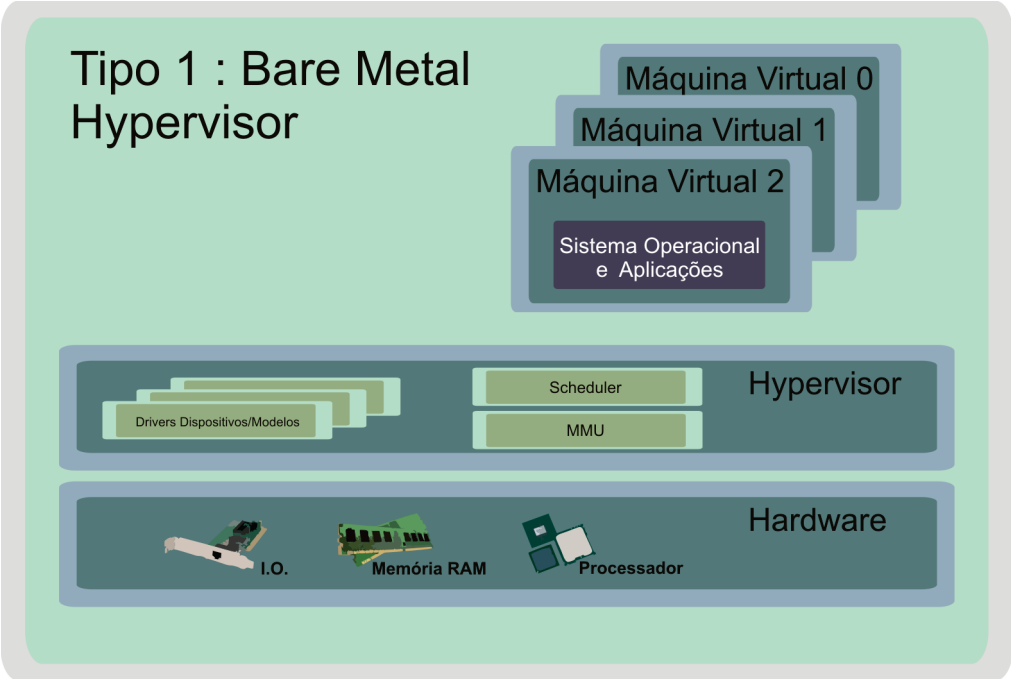

Figura 4 - XEN - Arquitetura Bare Metal

Fonte: http://www.xenproject.org/developers/teams/hypervisor.html

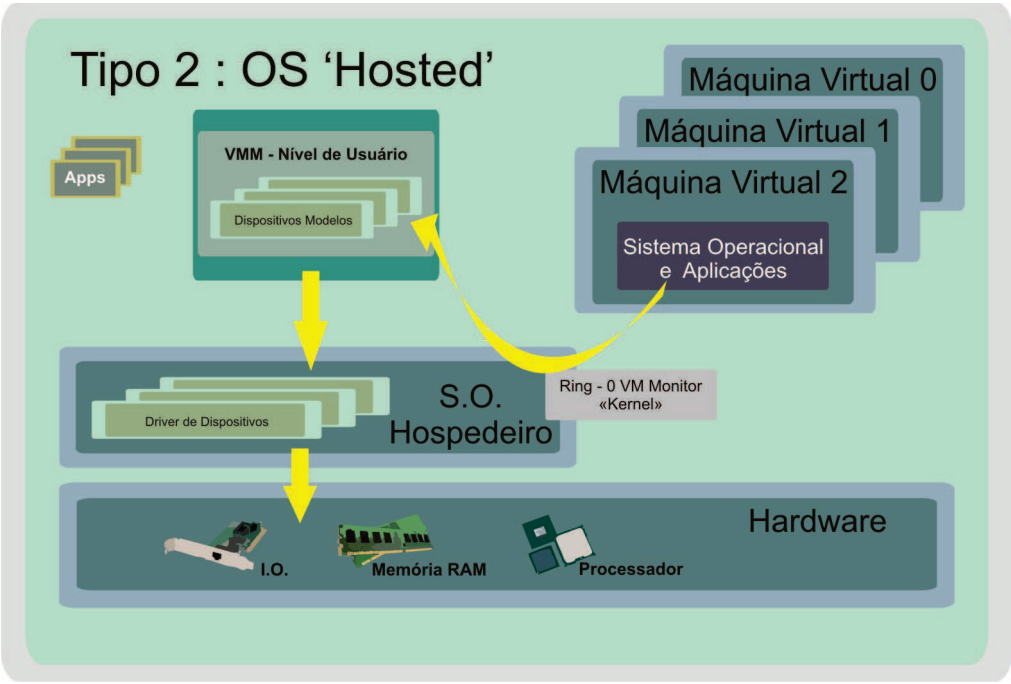

Figura 5 - XEN - Arquitetura OS 'Hosted'

Fonte: http://www.xenproject.org/developers/teams/hypervisor.html

No contexto de gerenciador proprietário (não livre) o VMWare é amplamente utilizado. O VMWARE ESX e ESXi[6] são *Hypervisors* desenvolvidos pela VMware que fornecem a capacidade de abstrair os recursos computacionais, como processadores, BIOS, armazenamento em disco, memória e rede para o instanciamento de várias máquinas virtuais. O VMWare ESX e o ESXi não necessitam de um sistema operacional para ser executado, pois podem ser instalados diretamente no *hardware*, sendo, portanto, do tipo *Bare Metal.*

3.4 GERENCIADOR DE RECURSOS DA NUVEM

As plataformas de nuvem diferem essencialmente nos métodos de gerenciamento do *cluster,* interferindo nas ações dos *Hypervisors*. O papel do Gerenciador de Recursos da Nuvem é coordenar e monitorar os *Hypervisors* implantados nas máquinas do *cluster* físico. Assim, esse gerenciador é o principal elemento de uma plataforma de nuvem. Entre as plataformas de computação em nuvem destacamos o Eucalyptus e o OpenNebula. O acesso a essas plataformas pode ser feito por meio de uma API (*Application Programming Interface* - Interface de Programação de Aplicativos), por meio de formulários web que abstraem a complexidade da API e interfaces REST. Essas formas de interação com a nuvem permitem ao usuário instanciar e gerenciar MVs sob demanda. A seguir são apresentados detalhes do Eucalyptus e do OpenNebula sob o ponto de vista de suas arquiteturas.

3.4.1 EUCALYPTUS

O Eucalyptus surgiu a partir de um projeto de pesquisa no Departamento de Ciência da Computação da Universidade da Califórnia, Santa Barbara (Eucalyptus, 2013). O Modelo de serviço do Eucalyptus é o IaaS, já que ele permite o funcionamento de uma nuvem através da Internet e oferece serviços como armazenamento, virtualização de sistemas operacionais,

6 http://www.vmware.com/files/br/pdf/products/VMW_09Q1_BRO_ESX_ESXi_BR_A4_P6_R2.pdf

clusters e outras configurações. Os tipos de nuvem em que ele pode ser implantado são pública, privada, híbrida e comunidade.

Segundo Eucalyptus (2013), a plataforma pode ser implantada em uma infraestrutura de redes de computadores existente para prover uma camada de serviços *web* segura e escalável. Seus serviços de virtualização podem ser ajustados de acordo com a necessidade, dependendo da carga de trabalho dos aplicativos a serem executados.

Eucalyptus é compatível com os *Hypervisors* XEN e KVM através da *libvirt Virtualization API*. Conforme Eucalyptus (2013), a arquitetura dessa plataforma é composta por cinco componentes principais: o Controlador da Nuvem, o Controlador do *Cluster*, o Controlador de Nó, o Controlador de Armazenamento e o Walrus. A Figura 6 apresenta a organização desses componentes.

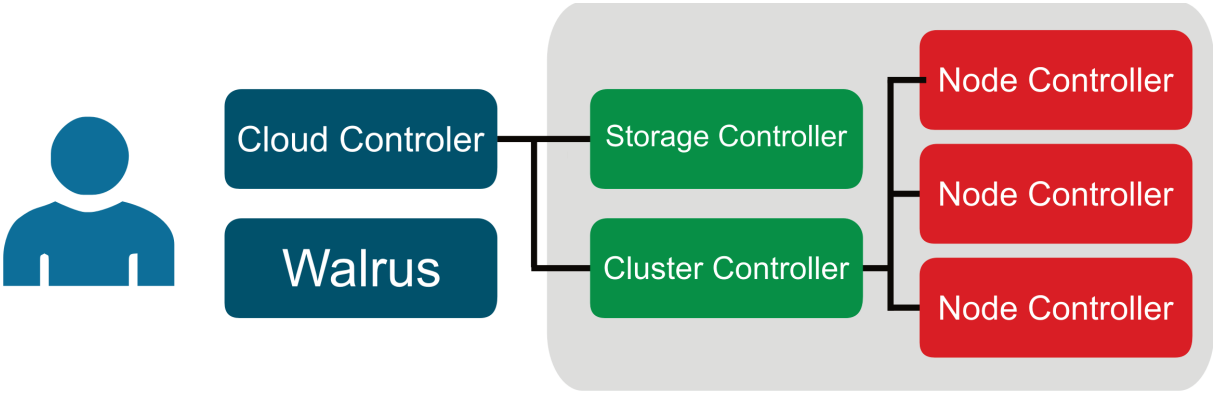

Figura 6 - Arquitetura do Eucalyptus
Fonte: Eucalyptus Documentation. http://www.eucalyptus.com/docs/eucalyptus/3.3/install-guide/

O Controlador da Nuvem, ou *Cloud Controller* (CLC) é o responsável pelo gerenciamento da nuvem. Esse controlador fornece ao usuário a possibilidade de controlar a nuvem e manipular seus recursos por meio de solicitações aos CCs. Ele é responsável por gerenciar os recursos subjacentes, como servidores, rede e armazenamento. O Controlador do *Cluster*, ou *Cluster Controller* (CC), é responsável pela administração da rede das MVs e mantém as informações de rede dos controladores de nós. O Controlador de Nó deve ser associado a um *Cluster Controller*. O Controlador de Nó, ou *Node Controller* (NC), é responsável pelo controle das atividades, execução e finalização das MVs. O Controlador de Armazenamento, ou *Storage Controller* (SC), é responsável por providenciar discos virtuais para as instâncias de MVs e é similar ao Amazon EBS. Segundo Amazon (2013), Amazon EBS, ou *Amazon Elastic*

Block Store, é um serviço que permite a criação de volumes de armazenamento que podem ser acoplados às instâncias como um dispositivo de armazenamento das máquinas virtuais.

O *Walrus* provê um mecanismo para o gerenciamento dos controladores de nó. Ele fornece um mecanismo para armazenar e acessar MVs e dados dos usuários, além de ser compatível com a Amazon S3. Amazon S3 ou *Amazon Simple Storage Service* é um serviço oferecido pela Amazon que possibilita ao desenvolvedor armazenar e recuperar *Bucket (Balde)* de dados. O usuário pode armazenar um objeto que contenha entre 1 byte e 5 terabytes de tamanho, segundo Amazon (2013). Para o sistema de arquivos o *Walrus*[7] tem como padrão o POSIX para armazenamento *Buckets* e Objetos. Por padrão o Eucalyptus utiliza o diretório *$EUCALYPTUS/var/lib/eucalyptus/bukkits* e cria um novo diretório a cada *bucket* criado. Os objetos são armazenados em um único arquivo dentro de um diretório de *bucket*. POSIX[8] é um padrão de interface que inclui um interpretador de comandos e utilitários que permitem a portabilidade de aplicações em nível de código.

O Eucalyptus é disponibilizado para *download* no formato *Faststart*[9], que é uma imagem de CD do CentOS 6 que permite ao usuário instalar e configurar rapidamente uma nuvem do tipo IaaS. O usuário pode instalar todos serviços em um computador ou dividí-los entre várias máquinas. Entre as opções disponíveis, o usuário pode optar por instalar o *Front-End*, que contém os serviços de CLC, Walrus, SC e CC em um computador e instalar o controlador de nó em outros computadores, podendo adicionar vários controladores de nós em uma mesma rede. Outra opção disponível é o *Cloud-in-a-Box*, onde o usuário pode instalar todos os serviços em um único computador.

O Eucalyptus fornece um repositório onde é possível efetuar o download modelos de MVs no formato EMI (*Eucalyptus Machine Images*). Para adicionar as imagens na plataforma é necessário utilizar a ferramenta Euca2ools, disponibilizada pela comunidade do Eucalyptus. Este utilitário foi escrito em Python e é um aplicativo CLI (C*ommand Line Interface)* que permite ao usuário interagir com os *Serviços Web* do Eucalyptus. Segundo a documentação do Euca2ools, a maioria dos comandos são compatíveis com a Amazon EC2, S3 e IAM. O IAM (*Identity and Access Management*) é um serviço da Amazon que possibilita o administrador da nuvem controlar as credenciais e permissões de usuários e grupos de usuários para o acesso

7 https://github.com/eucalyptus/eucalyptus/wiki/Walrus

8 http://pubs.opengroup.org/onlinepubs/9699919799/

9 http://www.eucalyptus.com/eucalyptus-cloud/get-started/try

aos recursos da Amazon *Web Services*. Na distribuição *fastStart* o Euca2ools é instalado juntamente com o Eucalyptus. A partir do Euca2ools é possível executar comandos para vincular a imagem de uma MV à plataforma e posteriormente instanciá-la.

3.4.2 OPENNEBULA

O OpenNebula é uma solução de código livre para um *datacenter* de virtualização, altamente escalável e adaptável para o gerenciamento de dados virtualizados tipo IaaS. Em 2005, Ignácio M. Lioriente e Rubén S. Montero criaram o OpenNebula em seu projeto de pesquisa. A primeira versão publicada do *software* foi em março de 2008, e desde então opera como um projeto de *software* livre. Foram muitos anos de pesquisa e colaboração da comunidade de desenvolvedores para chegar ao atual nível da aplicação. O OpenNebula se tornou um eficiente *software* de gerenciamento de MVs em larga escala.

De acordo com OpenNebula (2013), o OpenNebula é compatível com os *Hypervisors* XEN, KVM e VMWare. Segundo OpenNebula (2013), a arquitetura do OpenNebula é composta por um nó controlador principal, o *Front-end*, que controla os hosts, e nós escravos, que por sua vez são responsáveis por executar as MVs solicitadas pelo *Front-end.* As imagens das MVs ficam armazenadas nos *Datastores*. Os *Datastores* podem ser acessados através do Front-End utilizando tecnologias SAN (*Storage Area Networks*), NAS (*Network-Attached Storage*) ou um armazenamento de conexão direta.

A Figura 7 apresenta a arquitetura da plataforma OpenNebula. O serviço ONED é o principal *daemon*, responsável por gerenciar os demais componentes da plataforma.

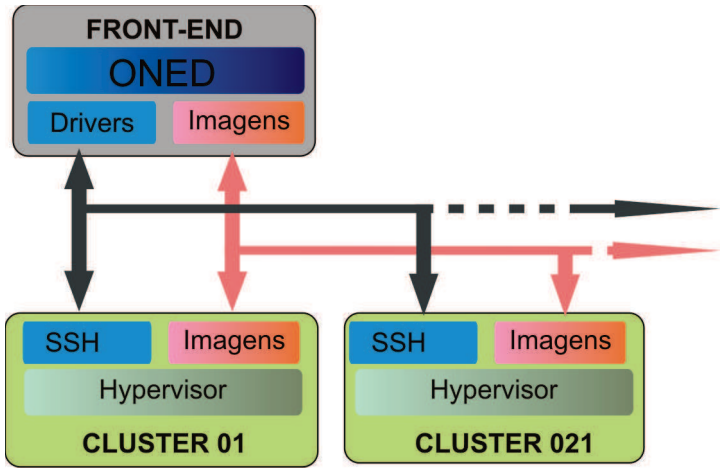

Figura 7 - Arquitetura do OpenNebula

Fonte: OpenNebula Documentation.

http://opennebula.org/documentation:rel4.2:plan

Segundo OpenNebula (2013), essa plataforma fornece 6 tipos de armazenamento de dados:

- *System:* neste tipo de armazenamento de imagem, dependendo da tecnologia de armazenamento, podem ser utilizadas cópias completas da imagem original, qcow (formato de imagem utilizado pelo qemu) de disco deltas ou *links* para o arquivo original.
- *File-System:* a imagem é armazenada em formato de arquivo. São armazenadas em um diretório montado com acesso a um servidor SAN/NAS.
- *ISCSI/LVM:* a imagem é armazenada em forma de dispositivo de bloco.
- *Vmfs:* um *datastore* especializado em formato Vmfs para ser utilizado com o *Hypervisor* VMWare.
- *Ceph:* a imagem de disco é armazenada utilizando dispositivo em bloco Ceph.
- *Files:* este tipo de armazenamento é para arquivos como Kernel e Ramdisk.

4 POSSIBILIDADES DE INVESTIGAÇÃO CIENTÍFICA EM PLATAFORMAS DE NUVEM

Durante a revisão bibliográfica notamos a tendência de pesquisadores utilizarem medições de desempenho e usabilidade para efetuar comparações entre as plataformas de computação em nuvem. Mas, certamente, essas não são as únicas formas de investigação existentes. Sempolinski et al. (2011) fazem uma comparação crítica entre OpenNebula, Eucalyptus e Nimbus, além de apresentarem algumas sugestões de melhorias nessas plataformas. Ueda et al. (2010) fazem uma avaliação da variação de desempenho quando há carga de trabalho no Eucalyptus e no OpenNebula. Cerbelaud et al. (2009) descrevem uma experiência em campo usando quatro plataformas de computação em nuvem, sendo elas o ECP, o Eucalyptus, o OpenNebula e o Virt. Cerbelaud et al. relatam as diferenças de desempenho em transferência de imagens, transferência de dados e armazenamento. Nurmi et al. (2009) apresentam o Eucalyptus e descrevem detalhes sobre seu funcionamento, a arquitetura e o gerenciamento das MVs nessa plataforma. Rego et al. (2011) propõem um Workflow para alocação de MVs utilizando características de processamento. Peixoto (2012) destaca em seu trabalho de dissertação de mestrado os mecanismos de distribuição *Round-Robin, First-Fit, Match-Making, Portable Batch System* e Oracle *Grid Engine*. Entretanto, ainda são encontradas dificuldades em abordar o funcionamento dessas plataformas. Necessitamos de uma visão arquitetural que identifique clara e simplificadamente que tipos de mecanismos podem ser explorados nas plataformas de nuvem.

Calheiros (2011) comenta que uma análise de nuvem pode ser dividida conforme o provisionamento da máquina virtual, o provisionamento de recursos, e o provisionamento da aplicação. Embora existam várias formas de uso para uma nuvem, estamos especificamente interessados nos casos em que há instâncias de MVs. No Eucalyptus podemos acompanhar o processo de provisionamento da MV somente com o estado "*Pending*", onde os recursos estão sendo provisionados, e o estado "*Running*" onde a MV está pronta para inicializar o sistema operacional. Informações sobre o que ocorre durante a inicialização do sistema operacional ou depois deves ser providas por outros sistemas, muitas vezes com soluções particulares. No OpenNebula os estados são mais detalhados, sendo divididos em várias etapas. Mas o problema de se obter informações sobre o sistema operacional dentro da MV ainda ocorre com essa

plataforma. No OpenNebula os estados são:

- *Pending:* primeiro estado da MV. Nessa fase o OpenNebula está aguardando a alocação dos recursos necessários. A MV passa para o próximo estágio se for inicializada pelo *Scheduler* ou quando o usuário executa o comando *deploy*. No caso do comando *deploy*, o usuário pode indicar explicitamente onde a MV será instanciada.
- *Prolog:* o sistema da nuvem está transferindo os arquivos da MV (imagem do disco e o arquivo de recuperação) para o *nó* da nuvem (máquina do *cluster* físico) onde ela será instanciada, conforme alocação dos recursos feita no estado *Pending* por meio do *Scheduler*. A instânciação da MV no nó da nuvem é papel do *Hypervisor*.
- *Boot:* o OpenNebula está aguardando o *Hypervisor* instanciar a MV.
- *Running:* a MV está inicializada. Mas isso não significa que o sistema operacional está carregado e sim que esse carregamento está em andamento.

Na Figura 8 podemos acompanhar o fluxo dos estados de uma MV mencionados anteriormente.

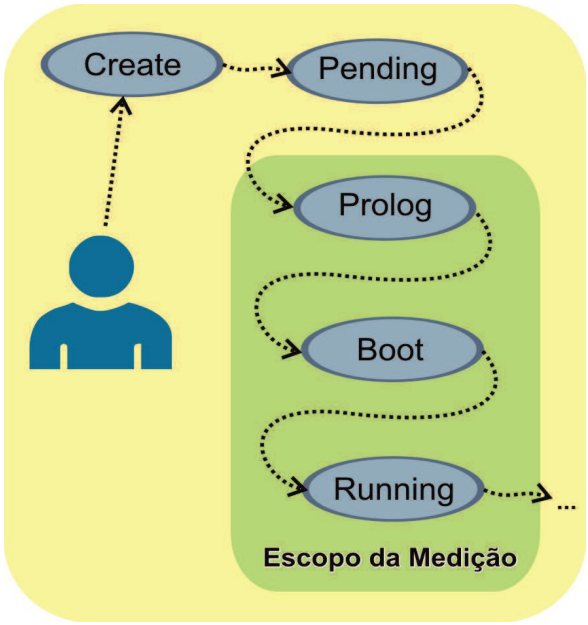

Figura 8 - Fluxograma de Estados no OpenNebula

Fonte: OpenNebula Documentation.

http://opennebula.org/documentation

O escopo de medição considerando os estados de uma MV é apenas um escopo entre outros que também tem potencial de investigação científica. Ao considerarmos que uma nuvem em execução depende de seus usuários para que recursos sejam acionados dentro dela, também podemos supor que o ponto de partida é o que o usuário solicita. Complementarmente, o ponto final é a instância total do recurso solicitado. Após essas fases, ainda é possível que o usuário use recursos dentro das instâncias com o objetivo de executar aplicativos. Para identificar os principais recursos que estão envolvidos entre a solicitação do usuário e a instância efetivada, ilustramos na Figura 9 uma arquitetura no nível de abstração que é comum para qualquer plataforma de nuvem.

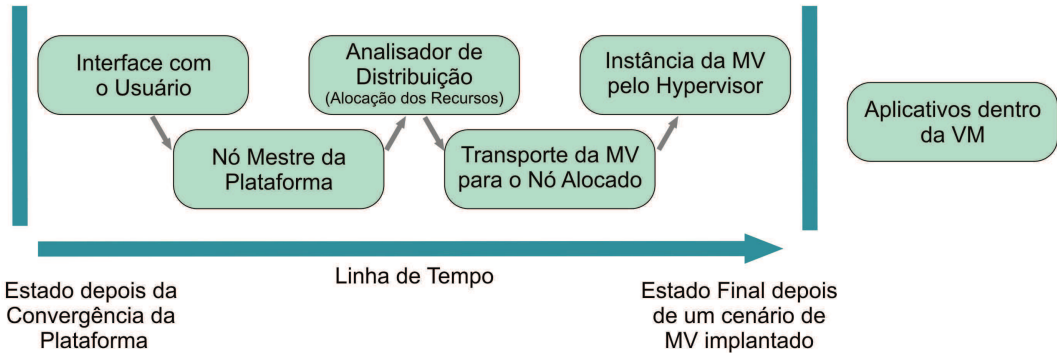

Figura 9 - Componentes comuns entre plataformas de nuvem.

Quando uma plataforma de nuvem é inicializada, o computador mestre do *cluster* físico se comunica com os demais computadores, chamados de nós da nuvem, a fim de obter informações sobre os recursos físicos disponíveis. A replicação do computador mestre também é possível nas plataformas de nuvem, assim como implantação de mecanismos de tolerância a falhas nos nós escravos. Chamamos de convergência da nuvem o estado após a inicialização completa de todos os nós do *cluster* e depois da obtenção de informações realizadas pelo nó mestre. Entre outras palavras, depois da convergência, a nuvem estará disponível para seus usuários.

O usuário da nuvem solicita instâncias de MVs por meio do módulo de Interface com o Usuário, comumente feito usando Serviços Web, REST, API, entre outros. Tal módulo de Interface com o Usuário formata e solicita os recursos para o nó mestre da plataforma. No nó mestre existem algoritmos de decisão que avaliam onde os recursos devem ser alocados para

garantir as instâncias solicitadas. Periodicamente, o nó mestre requisita informações dos demais nós. Dessa forma, ele monitora o uso dos recursos e pode garantir a alocação de novas instâncias de forma balanceada. O algoritmo de decisão mais comum em plataformas de nuvem é o *Round-Robin*, que distribuí de forma circular os recursos. Após a alocação de recursos, a imagem da MV é transmitida para o nó selecionado. Existem várias formas de transmissão. Os serviços SCP e NFS são os mais comumente encontrados. Quando a transmissão se completa, o *Hypervisor* do nó da nuvem instancia a MV. Chamamos de estado final a fase em que o *Hypervisor* instanciou a MV. Entretanto, esse é um estado final apenas do ponto de vista do nó mestre. A convergência total para o usuário ainda depende da inicialização completa dos sistemas que estão dentro da MV.

Para cada um dos elementos apresentados na Figura 9, podemos identificar potenciais questões problemas, como seguem abaixo:

- Interface com o usuário → Quais vantagens e desvantagens existem entre cada tipo de Interface (como REST, API, etc)? Existe interface ajustada para certos problemas específicos?
- Nó mestre da Plataforma e Analisador de Distribuição → O quanto de CPU e memória são necessários para cada plataforma? Quais são os mecanismos e em quê eles diferem? Como é realizado o monitoramento dos nós da nuvem? Como é realizada a tomada de decisão sobre a alocação das MVs? Ainda há espaço científico para novos algoritmos de alocação otimizada?
- Transporte da MV → Quais são as formas de transportar imagens pelo *cluster* formador da nuvem? É possível otimizar essa comunicação?
- Instância da MV pelo *Hypervisor* → Quais são as vantagens e as desvantagens entre diversos *Hypervisors*? O que pode ser otimizado?

Dentro da mesma arquitetura da Figura 9 também podem ocorrer investigações sobre o *cluster* que dá forma a nuvem, avaliando, por exemplo, questões de segurança, tolerância a falhas, desempenho da rede, desempenho dos dispositivos de armazenamento, entre outras. Nesse sentido, vale a pena conhecer como as instâncias de MV se comunicam em rede entre diversos nós de nuvem. A Figura 10 apresenta uma arquitetura que representa todas as

plataformas de nuvem.

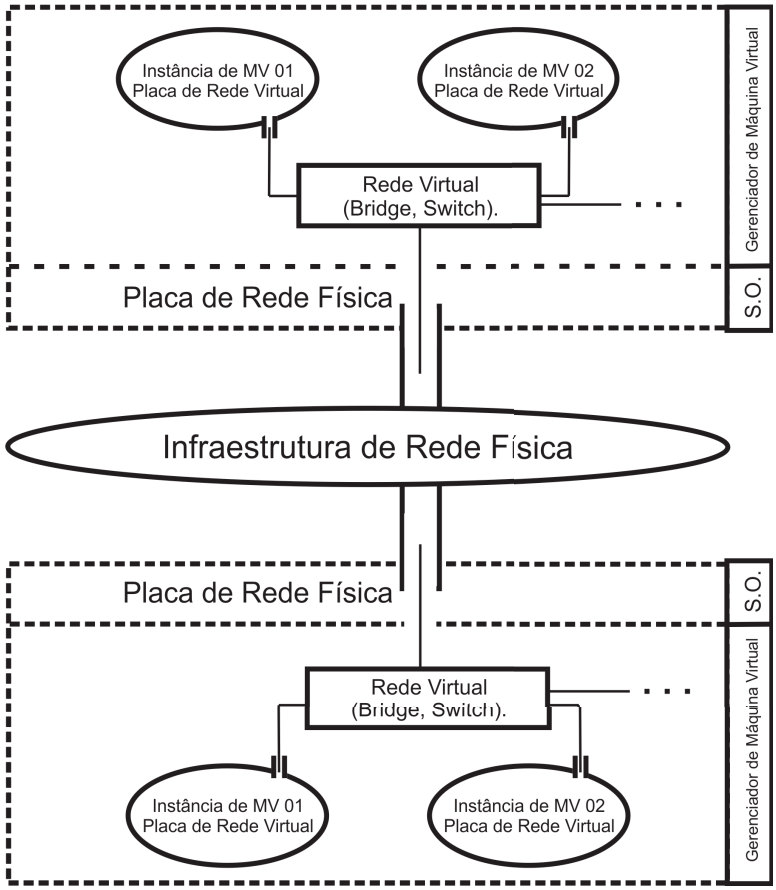

Figura 10 - Arquitetura da comunicação em rede feita por MVs nas plataformas de nuvem.

Fonte: Kraemer et al. (2013)

Diversas investigações podem envolver otimização na comunicação entre MVs. Existem três formas de comunicação: (1) comunicação entre MVs no mesmo nó da nuvem; (2) comunicação entre MVs em nós de nuvem diferentes; e (3) comunicação mista, envolvendo diversas MVs no mesmo nó e em nós diferentes.

As possibilidades de investigação tendo como foco o que existe dentro das MVs é imensurável. A principal razão para isso é que existem muitos tipos de aplicativos, que resolvem muitos tipos diferentes de problemas. Uma questão relevante sobre esse contexto é: Qual a influência da arquitetura da nuvem e sua rede de comunicação nas aplicações que são executadas dentro das instâncias de MVs? Contudo, ainda existe muito campo de investigação

sob diversos aspectos envolvendo plataformas de nuvem.

No capítulo 5 abordamos um caso prático de implantação e observações sobre as plataformas.

5 CASO PRÁTICO DE IMPLANTAÇÃO E OBSERVAÇÕES SOBRE AS PLATAFORMAS

Realizamos experimentos práticos envolvendo as plataformas OpenNebula e Eucalyptus no intuito de descobrir algumas características que pudessem impactar na nossa visão de arquitetura simplificada e contribuir com novas observações sobre pesquisa em nuvem. Para tanto, instalamos as plataformas no mesmo computador usando o esquema de *dual boot*. O computador possui 1 Processador AMD 3 FX-6 6100 de 3.3Ghz com 6 núcleos, 8GB de Memória RAM com barramento de 1333mhz, 1TB de HD 7200rpm Sata II e Placa-mãe AsRock N68-S3. O sistema de arquivos utilizado na implantação das plataformas foi o Ext4 padrão nas instalações dos sistemas operacionais Ubuntu e CentOS6.

Para implantação do Eucalyptus utilizamos a versão 3.3.1. O *download* da imagem do disco de instalação foi feito diretamente do sítio da plataforma. O sistema operacional da imagem é o CentOS 6. A imagem oferece três tipos de instalação: instalação do *Front-End*, *Node Controller*, e *Cloud-in-a-box*. O modelo de instalação *Front-End* compreende a instalação do controlador da nuvem, ou seja, a instalação dos serviços do nó mestre, o *Cloud Controller, o Walrus, o Cluster Controller* e o *Storage Controller*. O modelo de instalação *Node Controller* abrange somente a instalação dos serviços do controlador de nó escravo. Esse tipo de instalação somente deve ser feito após a instalação e configuração do *Front-End, nó mestre*. O modelo de instalação *Cloud-in-a-box* contempla todos os serviços da nuvem instalados em um único computador. Utilizamos a opção *Cloud-in-a-box* para implantação da plataforma Eucalyptus no equipamento.

Para implantação do OpenNebula escolhemos o sistema operacional linux Ubuntu *Server* 12.04 LTS, disponível no sítio da plataforma. O sítio onde é disponibilizada a plataforma não oferece, na versão atual 4.2 do OpenNebula, uma opção de instalação do tipo *Cloud-in-A-Box*. O processo de instalação é mais complexo e demorado que o Eucalyptus, requerendo do administrador um bom conhecimento sobre o sistema operacional Linux. A instalação do OpenNebula é disponibilizada em pacotes: o componente de s*oftware* OpenNebula e o *AppMarket*. O componente de *software* contém os pacotes de instalação .deb:

- OpenNebula: Fornece o *daemon* que é o serviço responsável por gerenciar os

clusters, redes virtuais, máquinas virtuais, usuários, grupos e repositório de imagens;

- OpenNebula *Tools*: Interface para execução de linhas de comandos;
- OpenNebula Sunstone: Interface Web para gerenciamento da nuvem, como criação de *templates*, usuários, gerenciamento de imagens, *clusters*, redes e máquinas virtuais.;
- OpenNebula *Node*: Responsável por preparar o computador para ser um nó da nuvem;
- OpenNebula *Gate*: Módulo responsável por habilitar a comunicação entre as MVs e o OpenNebula;
- OpenNebula *Flow*: Gerencia os serviços e a elasticidade da nuvem;
- OpenNebula *Commom*: Providência a criação do usuário e os arquivos comuns entre os módulos da nuvem;
- libOpenNebula Ruby: Instala todas as bibliotecas Ruby necessárias para o funcionamento do OpenNebula.

Todos os pacotes do OpenNebula foram instalados em um único computador, OpenNebula *Master* e OpenNebula *Node*, seguindo o procedimento disponibilizado na documentação do sítio do OpenNebula. Observando essa plataforma, verificamos que ela disponibiliza alguns métodos de transferência de MVs do *DataStore* para o *Node* da Nuvem:

- Compartilhamento NFS: Esse modo assume que o diretório onde estão localizadas as máquinas virtuais é compartilhado entre o OpenNebula *Master* e os OpenNebula *Nodes*. A MV é uma cópia da imagem compartilhada.
- Sem Compartilhamento – SSH: Nesse cenário não há compartilhamento de imagens entre o OpenNebula *Master* e os OpenNebula *Nodes*. Os nós de um mesmo *cluster* poderão compartilhar as imagens para facilitar a migração em tempo real. As imagens são copiadas usando SCP para o nó da nuvem onde a MV será instanciada.

As Figuras 11 e 12 apresentam respectivamente essas abstrações NFS e SSH de transferência de MVs.

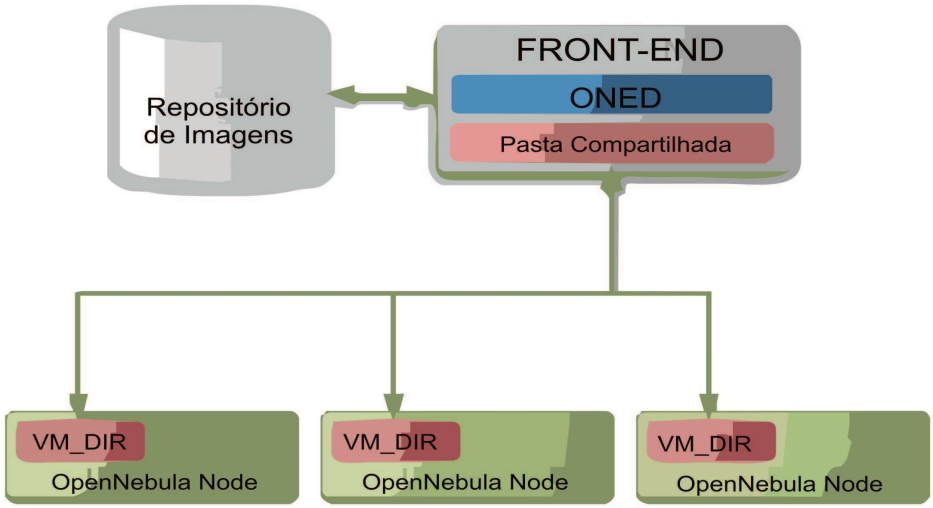

Figura 11 - Modo Compartilhado – NFS

Fonte: OpenNebula Documentation. http://opennebula.org/documentation

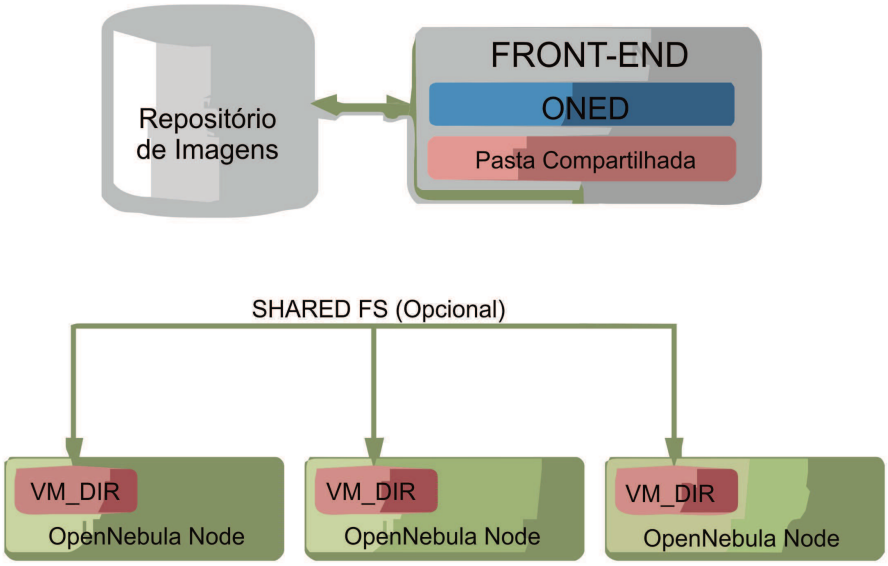

Figura 12 - Modo SSH

Fonte: OpenNebula Documentation. http://opennebula.org/documentation

No Eucalyptus não encontramos uma forma de alterar o tipo de compartilhamento de imagens. Por outro lado, não podemos afirmar que essa possibilidade não existe. Ueda et al. (2010) citam que o Eucalyptus armazena as imagens em um local de armazenamento centralizado. Quando o usuário solicita a criação de uma nova MV, a imagem é copiada para o

disco do nó da nuvem onde é utilizada *pelo Hypervisor*.

Após a implantação das plataformas, mensuramos o uso de memória no estado de convergência e o uso da CPU nos momentos em que uma instância era solicitada até entrar em estado *Running*. Para os testes com instâncias, criamos 4 modelos de Máquinas Virtuais: 5 GB, 10 GB, 15 GB e 20 GB. Com isso, podemos observar se há alguma diferença no tempo de carregamento das MVs de diferentes tamanhos. Para cada tamanho, executamos 30 experimentos. O tempo total observado agrega as latências dos módulos Interface do Usuário (*submit* do usuário), receptação da solicitação pelo nó mestre, alocação dos recursos, transporte da MV e instância do *Hypervisor* até o estado *Running*. O transporte da MV (NFS ou SSH) gera a maior latência nesse cenário.

O OpenNebula não oferece a capacidade de alterar o tamanho do disco rígido da imagem do sistema operacional, mas permite acoplar discos virtuais à MV. Por esse motivo, foi necessário criar MVs com o Virtual Box 4.2 e convertê-las para o formato qcow2 usando o qemu-img. Dentro das MVs optamos por instalar o Debian 6. O Eucalyptus disponibiliza imagens através do Eucalyptus *Image Store*. O formato de imagem utilizado no Eucalyptus foi o RAW, com a extensão .img. Apesar do OpenNebula fornecer um serviço similar no OpenNebula *Market Place*, pelo fato de não podermos alterar o tamanho do disco das imagens utilizamos as que foram criadas pelo Virtual Box.

O OpenNebula permite a criação de *templates* de imagens. No *template* é possível alterar características das instâncias, vinculando configurações da virtualização (principalmente CPU e RAM), rede, outras opções. Através da interface de gerenciamento do Eucalyptus verificamos que essas configurações também são possíveis. Nos dois casos configuramos o *template* para usar 1 CPU virtual e 512 GB de Memória RAM.

A Figura 13 apresenta os resultados dos experimentos com MVs de 5 GB, 10 GB, 15 GB e 20 GB. O detalhamento dos dados pode ser observado na Tabela 3. Utilizamos o intervalo de confiança de 95% no rol de 30 experimentos para cada tamanho de MV.

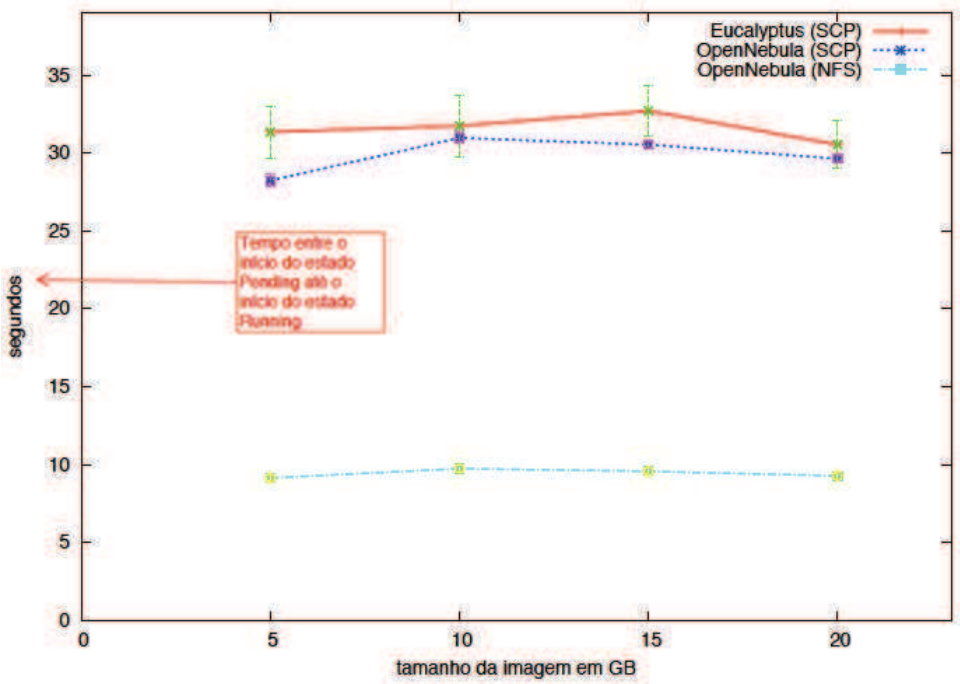

Figura 13 - Resultado dos experimentos com transporte de MVs.

No estado de convergência das plataformas e antes das instâncias dos usuários observamos que o OpenNebula consumia 270 MB de RAM e o Eucalyptus 1680 RAM. Uma explicação para o Eucalyptus consumir mais memória é que ele possui mais módulos de gerenciamento sendo usados no cenário com um só computador (*Cloud-in-a-box*). Sobre o consumo de CPU durante os processos de instanciação utilizamos o *mpstat* para acompanhar o processamento. No OpenNebula o pico de consumo médio da CPU chegou em 23,9% para configuração NFS e 58,7% para a configuração SSH. Enquanto no Eucalyptus chegou a 41,6%.

Sobre os testes com NFS, utilizamos a configuração padrão do NFSv4. Entretanto, otimizações podem ser feitas a fim de reduzir o tempo de tráfego, conforme Kuroda (2011). Podemos concluir que não há diferenças significativas entre SCP no Eucalyptus e no OpenNebula usando apenas um computador. Embora o Eucalyptus tenha apresentado mais variações (desvio no gráfico), ambos podem estar em uma mesma expressão de linha (há interseções). Por outro lado, há diferença significativa entre NFS e SCP. Verificamos o código *node* da imagem usando o comando "ls -i" para se certificar de que o NFS local não havia feito apenas um *link* para a cópia da imagem, apontando para o mesmo *id node*. O resultado desse comando mostrou *nodes* diferentes para a imagem original e sua cópia para

instanciação. Contudo, a transferência com NFS gera muito menos latência. Também devemos considerar que o SCP é um serviço SSH e por isso implanta um certo nível de segurança que pode impactar na latência.

Em cenários com muitos computadores, os diferentes módulos de gerenciamento distribuídos ao longo do *cluster* podem gerar mais impactos nas latências, já que cada plataforma de nuvem é um sistema distribuído distinto. Com isso, recomenda-se experimentos com muitos computadores formando o *cluster* da nuvem. O que há de comum entre as plataformas são os módulos abstraídos apresentados pela nossa arquitetura. Ainda são encontradas muitas dificuldades em separar as latências, por exemplo, que ocorrem nos módulos de alocação de recursos, transporte da MV e instanciações feitas pelo *Hypervisor*. Uma solução alternativa consiste em subtrair o tempo de transferência da MV (tendo como base a vazão da comunicação em bps) do tempo total computado. Com isso, o tempo restante representa as demais latências.

Na tabela 3 descrevemos algumas comparações entre as características encontradas nas plataformas.

Tabela 3 - Comparação entre as plataformas Eucalyptus e OpenNebula

Características	OpenNebula	Eucalyptus
Licença	Open Source	Open Source
Repositório de Imagens de Máquinas Virtuais	OpenNebula MarketPlace[10]	EMI (Eucalyptus Machine Images)[11]
Hypervisors	XEN, KVM e VMWare	XEN, KVM e VMWare
Interface WEB	Sunstone, AWS APIs, OCCI, CLI	Eucalyptus User Interface, AWS APIs, CLI
Transporte de Imagens (padrão)	NFS e SSH	SSH
Modelo de Serviço	IaaS	IaaS

10 http://marketplace.c12g.com
11 http://emis.eucalyptus.com/

6 CONCLUSÃO

Diante das diversas plataformas de computação em nuvem disponíveis e da crescente utilização do paradigma de computação em nuvem, é fundamental a realização de pesquisas e comparações associadas aos componentes disponíveis em cada plataforma. O conteúdo deste trabalho procurou contribuir com essa questão apresentando e oferecendo para novos trabalhos uma base teórica sobre fundamentos, mecanismos e componentes das plataformas. Com base nesses conteúdos abordados, elaboramos uma arquitetura generalizada em nível de abstração que permite uma fácil visualização das trilhas de pesquisa. Os trabalhos futuros podem ser direcionados para essas trilhas.

As plataformas OpenNebula e Eucalyptus foram implantadas e comparadas para ilustrar uma linha de pesquisa. A comparação consistiu de instanciar máquinas virtuais de diferentes tamanhos (5GB, 10GB, 15GB e 20 GB) para avaliar principalmente a carga da CPU e o tempo da plataforma em atribuir o estado *Running*. Nesse contexto, um modelo de máquina virtual é transportado do repositório da plataforma para uma pasta de instâncias temporárias. Pelo fato da implantação ter sido feita no modelo *Cloud-in-a-box,* todos os *softwares* da plataforma foram instalados em apenas um computador. Os resultados mostram que a transferência via NFS obtêm melhor desempenho se comparada à forma SCP. Por outro lado, não há diferenças significativas entre as plataformas a respeito do tempo entre a solicitação do usuário e o estado *Running* no modelo *Cloud-in-a-box*. Embora os experimentos tenham contribuído para identificar características das plataformas, recomendamos que novos experimentos sejam feitos em *cluster* físico com mais computadores e mais MVs sendo solicitadas ao mesmo tempo, o que certamente produzirá maior distanciamento entre tempos de latência que ocorrem entre cada fase dos elementos da arquitetura.

REFERÊNCIAS

AMAZON. **Amazon Documentation**. Disponível em: http://aws.amazon.com/pt/documentation/ . Acesso em: 15/08/2013

ARMBRUST, M; FOX, A; GRIFFITH, R; JOSEPH, A. **Above the Clouds: A Berkeley View of Cloud Computing**, fev. 10, 2009.

BUYYA, R.; YEO, C. S.; VENUGOPAL, S.; BROBERG, J.; AND BRANDIC, I. (2009b). **Cloud computing and emerging it platforms: Vision, hype, and reality for delivering computing as the 5th utility**. Future Gener. Comput. Syst., 25(6):599–616.

CALHEIROS, R. N.; RANJAN, R.; BUYYA, R. **Virtual Machine Provisioning Based on Analytical Performance and QoS in Cloud Computing Environments.** ICPP '11 Proceedings of the 2011 International Conference on Parallel Processing. IEEE Computer Society Washington, DC, USA. 2011. p. 295-304

CERBELAUD, D.; GARG, S.; HUYLEBROECK, J. **Opening the clouds: qualitative overview of the state-of-the-art open source VM-based cloud management platforms**, 10th ACM/IFIP/USENIX International Conference on Middleware, Dezembro de 2009.

ENCYCLOPEDIA BRITANNICA. **Cloud Computing**. Disponível em: http://www.britannica.com/EBchecked/topic/1483678/cloud-computing. Acesso em: 06/11/2012.

EUCALYPTUS. **Eucalyptus Documentation**. Disponível em: http://www.eucalyptus.com/docs . Acesso em: 01/08/2013.

JHA, S.; KATZ, D. S.; LUCKOW, A.; MERZKY, A.; STAMOU, K. **Understanding Scientific Applications for Cloud Environments** Cloud Computing: Principles and Paradigms, March 2011. p. 345-371

JI, T.; SUN, A.; XIONG, F.; YUE, Q. **IaaS public cloud computing platform scheduling model and optimization analysis** International Journal of Communications, Network and Systems Sciences (IJCNS). 4.12 (Dec. 2011): p. 803.

KRAEMER, A.; OLIVEIRA, J. C. DE; SANTOS, F. A. G. DOS; MACIEL, A. C.; GOLDMAN, A.; CORDEIRO, D. **Dynamic Creation of BSP/CGM Clusters on Cloud Computing Platforms** 4TH International Conference on Emerging Intelligent Data and Web Technologies - Workshop: Third International Workshop on the Service for Large Scale Distributed Systems (SeDiS-2013), Xian, China. 978-1-4673-1986-7, 2013.

JI, T.; SUN, A.; Yue, Q.; Xiong, F. **IaaS Public Cloud Computing Platform Scheduling Model and Optimization Analysis**. International Journal of Communications, Network and System Sciences, 2011, Vol.04 (12), p.803.

KURODA, R. T. **Identificação Dos Recursos De Software Que Causam Impacto No Desempenho Do Perfil Móvel Em Ldap** Trabalho de Conclusão de Curso, Universidade Tecnológica Federal do Paraná, 2011.

LAUREANO, M. **Máquinas Virtuais e Emuladores. Conceitos, Técnicas e Aplicações.** São Paulo, Ed. Novatec, 2006.

MELL, P.; GRANCE, T. **Draft NIST Working Definition of Cloud Computing**. National Institute of Standards and Technology. Disponível em: http://www.cloudbook.net/resources/stories/the-nist-definition-of-cloud-computing. Acesso em: 12/11/2012

NURMI, D.; WOLSKI, R.; GRZEGORCZYK, C.; OBERTELLI, G.; SOMAN, S.; YOUSEFF, L.; ZAGORODNOV, D. **The Eucalyptus Open-source Cloud-computing System. Computer Science Department**, University of California, Santa Barbara, California, USA, 2009.

OPENNEBULA. **OpenNebula Documentation**. Disponível em: http://opennebula.org/documentation:rel4.2 . Acesso em: 11/08/2013.

PEIXOTO, M. L. M. **Oferecimento de QoS para computação em nuvens por meio de metaescalonamento**, Tese de defesa de doutorado, ICMC-USP, 2012.

Rego, P. A. L. **Uma Arquitetura para Provisionamento de Máquinas Virtuais Utilizando Características de Processamento** Dissertação de Mestrado, Universidade Federal do Ceará, Fortaleza, Ceará, Março 2012.

Rego, P. A. L.; Coutinho, E. F.; Souza, J. N. de. **Proposta de Workflow para Alocação de Máquinas Virtuais Utilizando Características de Processamento. Grupo de Redes de Computadores**, Engenharia de Software e Sistemas (GREat), Instituto UFC Virtual Universidade Federal do Ceará, Fortaleza, 2011.

SADASHIV, N.; KUMAR, S. M. D. **Cluster, Grid and Cloud Computing: A Detailed Comparison.** The 6th International Conference on Computer Science & Education. Singapore, 2011.

SEMPOLINSKI, P.; THAIN, D. **A Comparison and Critique of Eucalyptus, OpenNebula and Nimbus**, 2nd IEEE Cloud Computing Technology and Science (CloudCom), 2010, pp.417-426, Fevereiro de 2011.

SMITH, J.E; NAIR, R. **The architecture of virtual machines.** IEEE Computer, v.38,n.5, pp. 32-38, 2005.

SOROR, A. A.; MINHAS, U. F.; Aboulnaga, A.; Salem, K.; Kokosielis, P.; and Kamath, S. **Automatic virtual machine configuration for database workloads**. ACM Trans. Database Syst., 35(1):1–47. 2010.

SOUSA, F. R. C.; MOREIRA, L. O.; MACHADO, J. C. **Computação em Nuvem: Conceitos, Tecnologias,Aplicações e Desafios**. ERCEMAPI. 2009.

UEDA, Y.; NAKATANI, T. **Performance variations of two open-source cloud platforms**, Workload Characterization (IISWC), 2010 IEEE International Symposium on, pp.1-10, Dezembro de 2010.

WANG, L.; LASZEWSKI, G. VON; YOUNGE, A. ; HE, X.; KUNZE, M.; TAO, J.; FU, C. **Cloud Computing: a Perspective Study**, New Generation Computing, vol. 28, no. 2, pp. 137–146, 2010.