SETREM Faculdade Três de Maio

Tecnologia em Redes de Computadores

BRUNA ROBERTA THOMÉ

EDUARDO LUÍS HENTGES

ANÁLISE E COMPARAÇÃO DE FERRAMENTAS *OPEN SOURCE* DE COMPUTAÇÃO EM NUVEM PARA O MODELO DE SERVIÇO IAAS

Três de Maio

2013

BRUNA ROBERTA THOMÉ
EDUARDO LUÍS HENTGES

ANÁLISE E COMPARAÇÃO DE FERRAMENTAS *OPEN SOURCE* DE COMPUTAÇÃO EM NUVEM PARA O MODELO DE SERVIÇO IAAS

Relatório de Estágio Supervisionado
Sociedade Educacional Três de Maio- SETREM
Faculdade Três de Maio
Tecnologia em Redes de Computadores

Professor Orientador:
M.Sc. Dalvan Jair Griebler

Três de Maio
2013

TERMO DE APROVAÇÃO

BRUNA ROBERTA THOMÉ
EDUARDO LUÍS HENTGES

ANÁLISE E COMPARAÇÃO DE FERRAMENTAS *OPEN SOURCE* DE COMPUTAÇÃO EM NUVEM PARA O MODELO DE SERVIÇO IAAS

Relatório aprovado como requisito parcial para obtenção do título de **Tecnólogo em Redes de Computadores** concedido pela Faculdade de Tecnologia em Redes de Computadores da Sociedade Educacional Três de Maio, pela seguinte Banca examinadora:

Orientador: Prof. Dalvan Jair Griebler, Drndo.
PUCRS-RS

Prof. Cláudio Schepke, Dr.
Faculdade de Tecnologia em Redes de Computadores da SETREM

Prof. Vinicius da Silveira Serafim, M.Sc.
Faculdade de Tecnologia em Redes de Computadores da SETREM

Prof. Tiago Luis Cesa Seibel, M.Sc.
Faculdade de Tecnologia em Redes de Computadores da SETREM

Prof. Vera Lúcia Lorenset Benedetti, M.Sc.
Coordenação do Curso Superior de Tecnologia em Redes de Computadores
Faculdade de Tecnologia em Redes de Computadores da SETREM

Três de Maio, 14 de novembro de 2013.

RESUMO

A computação em nuvem é uma área que tem sido tema de constantes pesquisas, pois possibilita às empresas, independentemente do tamanho da mesma e de suas necessidades, ter acesso à tecnologia. Pode ser citado como exemplo de computação em nuvem, o IaaS, um modelo de serviço que fornece ao usuário a infraestrutura de *hardware*, sem que o mesmo necessite gerenciar ou realizar a manutenção. A principal motivação deste trabalho foi avaliar e comparar ferramentas *open source* de computação em nuvem para modelos de serviço IaaS para um ambiente de estações de trabalho. Em vista disso, foram usados para avaliação métodos como abordagem dedutiva e também qualitativa, os procedimentos de pesquisa exploratória e bibliográfica e por fim as técnicas de observação e teste. O objetivo do trabalho foi fazer uma análise comparativa do que a literatura descreve das principais ferramentas (OpenNebula, OpenStack, CloudStack, Eucalyptus, Ubuntu Enterprise Cloud, Abiquo, OpenQRM e ConVirt) e posteriormente analisar o comportamento de duas ferramentas (OpenNebula e OpenStack) em um ambiente de estações de trabalho, que foram escolhidas a partir da avaliação teórica e por serem ferramentas não abordadas na prática em nenhum trabalho relacionado. Dentre as características avaliadas no contexto da literatura, as ferramentas OpenNebula, OpenStack, OpenQRM e Eucalyptus demonstraram ser mais completas, por apresentarem uma maior gama de características. Já na implantação prática, os resultados obtidos a partir dos testes, serviram para realizar a avaliação entre as ferramentas em relação à coerência com a literatura e o desempenho apresentado durante o funcionamento da nuvem. E os resultados mostraram que o OpenNebula é a ferramenta que melhor se adequa a uma nuvem privada utilizando-se estações de trabalho, pois foi a que melhor se saiu nos testes realizados.

Palavras-chaves: Computação em nuvem, Nuvem privada, IaaS.

ABSTRACT

Cloud computing is an area that has been the subject of on going research, it allows companies, regardless of its size and its needing, to have access to technology. It can be considered as an example of cloud computing, the IaaS is a service model that provides the user the hardware infrastructure without needing to manage or perform maintenance. The main motivation of this work was to evaluate and compare open source tools in cloud computing for IaaS service models to an environment of workstations. To this, evaluation methods as well as deductive qualitative approach procedures were used for exploratory research and literature and finally the techniques of observation and testing. The objective was to make a comparative analysis of how the literature describes the main tools (OpenNebula, OpenStack, CloudStack, Eucalyptus, Ubuntu Enterprise Cloud, Abiquo, OpenQRM and ConVirt) and then analyze the behavior of two tools (OpenNebula and OpenStack) in an environment of workstations, which were chosen from the theoretical assessment tools because they are not addressed in any other work related. Among the evaluated parameters in the context of the literature tools, OpenNebula, OpenStack, and Eucalyptus OpenQRM, proved to be more complete because they show a great range of characteristics. In the practical implementation, the results obtained from the tests helped to conduct the evaluation between tools in relation to consistency with the literature and performance displayed during the operation of the cloud. And the results showed that the OpenNebula is the tool that best fits a private cloud using workstations, because it was best in the tests.

Keywords: Cloud computing, Private cloud, IaaS.

LISTA DE QUADROS

LISTA DE FIGURAS

LISTA DE SIGLAS

ACL – *Access Control List*

AMI- *Amazon Machine Image*

AoE- *ATA Over Ethernet*

API- *Application Programming Interface*

ARP- *Address Resolution Protocol*

CC- *Cluster Controller*

CLC- *Cloud Controller*

CRC- *Cyclical Redundancy Check*

CSA- *Cloud Security Alliance*

DNS- *Domain Name System*

EBS- *Elastic Block Storage*

EC2- *Elastic Compute Cloud*

FTP- *File Transfer Protocol*

HD- *Hard Disk*

HTTP- *HyperText Transfer Protocol*

IaaS- *Infrastructure as a Service*

IAM- *Amazon Identity and Access Management*

IP- *Internet Protocol*

iSCSI- *Internet Small Computer System Interface*

ISO- *International Standards Organization*

KVM- *Kernel-based Virtual Machine*

LAN- *Local Area Network*

LDAP- *Lightweight Directory Access Protocol*

LVM- *Logical Volume Management*

NC- *Node Controller*

NFS- *Network File System*

NNTP- *Network New Transfer Protocol*

OSI- *Open Systems Interconnection*

Paas- *Platform as a Service*

RBAC- *Role Based Access Control*

SaaS- *Software as a Service*

SC – *Storage Controller*

SETREM- Sociedade Educacional Três de Maio

SLA – *Service Level Agreement*

SMTP- *Simple Mail Transfer Protocol*

S3- *Simple Storage Service*

TCP- *Transmission Control Protocol*

TI- *Technology Information*

UDP- *User Datagram Protocol*

UEC- Ubuntu *Enterprise Cloud*

VCL- *Virtual Computing Lab*

VDC- *Virtual Data Center*

VM- *Virtual Machine*

SUMÁRIO

INTRODUÇÃO

O presente trabalho envolve a implantação de Computação em Nuvem Privada utilizando-se de ferramentas *open source*. Diante disso será efetuado o estudo sobre computação em nuvem privada e ferramentas que poderão ser utilizadas para essa tarefa, tendo como ambiente de teste estações de trabalho.

Levando em consideração o porte das empresas da Região Noroeste do RS, observa-se que as mesmas não devem possuir capital necessário para possuir servidores e grandes infraestruturas computacionais. Sendo assim, pode haver a oportunidade de implantação de Computação em Nuvem privada.

Isso motivou a desenvolver este trabalho que poderá ajudar as empresas a terem uma visão sobre Computação em Nuvem e aplica-lá em seu ambiente de trabalho.

A partir disso, pode ser verificada qual ferramenta é capaz de suprir as necessidades das empresas sem que haja a obrigação de altos investimentos, pois a proposta é a utilização de estações de trabalho que estão sem ser utilizadas.

O presente trabalho tem como intuito expor aos pesquisadores e demais interessados na área, as características e funcionalidades das ferramentas implantadas e apresentar uma avaliação sobre cada ferramenta, para que se possam sugerir aplicabilidades para as mesmas, possibilitando ao leitor a escolha da ferramenta que mais se adeque as suas necessidades.

Sendo assim, este projeto é dividido em três partes, o capítulo um, que apresenta os aspectos metodológicos, delimitação do tema, objetivos, hipóteses, período e procedimentos que foram utilizados na realização do estudo. O capítulo dois apresenta definições importantes para a realização deste trabalho. E o capítulo três que traz os resultados obtidos durante a realização do projeto.

CAPÍTULO 1 – ASPECTOS METODOLÓGICOS

1.1 TEMA

Análise e comparação de ferramentas *open source* para computação em nuvem privada que fornecem o modelo de serviço IaaS (*Infrastructure as a Service*) utilizando estações de trabalho.

1.1.1 Delimitação do tema

Análise e comparação das principais ferramentas *open source* para computação em nuvem. Neste sentido, através das características em comum foram comparadas as funcionalidades para facilitar a escolha de uma determinada ferramenta para criação de uma nuvem privada utilizando-se de estações de trabalho.

Esse projeto foi realizado pelos acadêmicos do Curso de Tecnologia em Redes de Computadores, da Sociedade Educacional Três de Maio- SETREM- RS, Bruna Roberta Thomé e Eduardo Luís Hentges, no período de junho a novembro de 2013, como Trabalho de Conclusão de Curso.

1.2 FORMULAÇÃO DO PROBLEMA

Empresas da Região Noroeste que possuem área de TI própria raramente possuem *hardware* robusto e de alto desempenho, que é comumente utilizado por *datacenters* provedores de serviços de computação em nuvem. Portanto, é um desafio desenvolver uma nuvem privada que seja uma alternativa viável para

adesão desta tecnologia nas pequenas empresas. Neste trabalho, o problema é identificar as características comportamentais das ferramentas *open source* para implantação de nuvem privada, usando estações de trabalho.

Considerando as características de computação em nuvem avaliadas, qual é a ferramenta mais apropriada para implantação de uma nuvem privada em estações de trabalho?

1.3 HIPÓTESES

1) Com o estudo, implantação e análise das ferramentas é possível identificar quais delas se desempenha melhor em uma nuvem com estações de trabalho.

    a. **Validação**: Serão elencadas todas as características presentes nas ferramentas, avaliando individualmente cada uma delas em cada uma dessas características. Isso é realizado em uma nuvem privada montada com estações de trabalho.

2) As ferramentas testadas em um ambiente de estações de trabalho são coerentes com o relato da bibliografia.

    a. **Validação**: Serão instaladas as ferramentas e durante o período de execução das mesmas, será possível verificar se ocorre um bom funcionamento. Além disso, será verificado se na prática as ferramentas apresentam as mesmas características descritas na literatura.

1.4 VARIÁVEIS

- Desempenho.

- Coerência.

1.5 OBJETIVOS

1.5.1 Objetivo Geral

Instalar, comparar e analisar as principais ferramentas *open source* para computação em nuvem voltadas para o modelo IaaS.

1.5.2 Objetivos Específicos

- Estudar sobre o assunto, as ferramentas, e trabalhos relacionados.

- Instalar e testar as ferramentas,

- Documentar, comparar e analisar os resultados.

1.6 JUSTIFICATIVA

A Computação em Nuvem é uma tecnologia em ascensão utilizada por empresas e instituições. Ela permite, por exemplo, o armazenamento de dados (que provê o acesso a qualquer hora e lugar, sendo necessária apenas uma conexão com a *Internet*) sem a necessidade de que os usuários possuam espaço pra armazenamento local. O uso de computação em nuvem viabiliza para as empresas o uso de ferramentas de TI, que agilizam o desenvolvimento do trabalho, sem que haja inicialmente a necessidade de altos investimentos em compra de *hardware* e gastos com manutenção dos equipamentos.

Com a grande utilização de Computação em Nuvem, surgiram diferentes tipos, dentre eles as nuvens privadas que assim como o nome diz, são nuvens particulares de uma pessoa ou empresa, onde o usuário possui controle sobre as aplicações utilizadas na nuvem.

Existem ferramentas que possibilitam desenvolver uma nuvem privada, mas é importante esclarecer as funcionalidades de cada ferramenta, para que a partir

destas informações, seja possível elucidar as pessoas o que cada ferramenta tem a oferecer.

Uma boa alternativa para empresas que dispõem de estações de trabalho sem utilidade é implantar uma nuvem privada, pois os recursos computacionais podem ser melhor aproveitados. Além disso, fornecer serviços como armazenamento, hospedagem de máquinas virtuais, processamento de dados e ferramentas de desenvolvimento.

O presente trabalho trará como benefício auxiliar os profissionais da área em escolher a ferramenta que melhor se adapta às suas necessidades em um ambiente de estações de trabalho, pois este trabalho faz uma avaliação comparativa das principais ferramentas *open source* para computação em nuvem privada.

1.7 METODOLOGIA

A metodologia científica em sua essência tem por finalidade estudar os métodos que identificam os caminhos percorridos para alcançar os objetivos propostos pelo plano de pesquisa. O pesquisador ao tomar conhecimento da existência de um problema, procura encontrar a solução e a partir do problema, inicia-se o processo da prática de pesquisa científica (LOVATO, EVANGELISTA, GÜLLICH, 2007, p.33).

1.7.1 Método de Abordagem

O método de abordagem utilizado no presente projeto foi o método dedutivo onde a partir de uma pesquisa é possível obter informações com objetivo de deduzir algo. E também a abordagem qualitativa que busca explicações teóricas e conceituais para fundamentar o estudo.

A abordagem dedutiva foi utilizada ao estudar os assuntos envolvidos no trabalho para assim poder aplicá-los utilizando da dedução e alcançar os resultados esperados. Já abordagem qualitativa foi utilizada em explicações teóricas que fornecerão a base a todo o trabalho.

1.7.2 Método de Procedimento

Para realizar este projeto foi utilizado o procedimento de pesquisa exploratória, com a qual foi possível obter maiores informações sobre o assunto a ser abordado no projeto, adquirindo um maior conhecimento para formular a delimitação do tema, os objetivos e as hipóteses do mesmo.

> Pesquisa exploratória: aquela que possui como finalidade proporcionar maiores informações sobre o assunto que se vai investigar; propiciar melhores condições para a delimitação do tema da pesquisa; orientar a fixação de objetivos e formulação de hipóteses, bem como auxiliar na descoberta de novos tipos de enfoque para o assunto (LOVATO, EVANGELISTA, GÜLLICH, 2007, p.35)

Foi desenvolvida uma pesquisa bibliográfica, utilizando como fonte livros, tutoriais, *Internet*, entre outras, com o intuito de obter maiores conhecimentos sobre o assunto a ser abordado.

“Pesquisa bibliográfica: é a consulta de obras escritas por outros autores a respeito do assunto a ser pesquisado” (LOVATO, EVANGELISTA, GÜLLICH, 2007, p.35).

1.7.3 Técnicas

As técnicas que foram utilizadas no presente projeto foram a observação onde examinou-se os fatos e fenômenos do tema, sendo de grande importância para uma análise aprofundada sobre o funcionamento e as características das ferramentas *open source* para computação em nuvem, buscando maiores informações a respeito do mesmo. E a técnica de testes utilizada para avaliar as ferramentas implantadas.

Segundo Lovato, Evangelista, Güllich (2007), a técnica de observação não consiste apenas em ver e ouvir, mas examinar os fatos e fenômenos que se deseja analisar. E os testes são utilizados com a intenção de obter informações de maneira quantitativa sobre determinado assunto.

1.8 DEFINIÇÃO DE TERMOS

- TCP/IP: segundo Torres (2001), o TCP/IP é o protocolo mais utilizado em redes locais atualmente mais necessariamente por causa da popularização da *Internet*, afinal esse protocolo foi criado para ser usado na mesma.

- Modelo OSI: segundo Tanenbaum (2003), o Modelo de Referência ISO OSI (*Open Systems Interconnection*), trata de interconexão de sistemas abertos, ou seja, sistemas que estão abertos à comunicação com outros sistemas.

- Computação em Nuvem: segundo Vaquero et al (2009) apud Veras (2011), *Cloud Computing* (Computação em Nuvem) é um conjunto de recursos virtuais de simples utilização e acesso como *hardware*, *software*, plataformas de desenvolvimento e serviços. Estes recursos podem ser reconfigurados de maneira dinâmica para que se ajustem a uma carga de trabalho variável, permitindo assim a otimização dos recursos. O recurso de Computação em Nuvem é utilizado através do modelo conhecido como pague-pelo-uso.

- Nuvem Privada: segundo Veras e Tozer (2012), a nuvem privada envolve uma infraestrutura de Computação em Nuvem operada e na maior parte das vezes gerenciada pela própria organização cliente. Os serviços são proporcionados para serem usados pela organização e não ficam disponíveis ao público em geral, mas existe a possibilidade de serem gerenciados por terceiros. Existem dois tipos de nuvem privada: a nuvem privada hospeda pela empresa e a hospedada pelo provedor de serviço.

- Nuvem Pública: conforme Veras e Tozer (2012), a nuvem pública é disponibilizada publicamente utilizando o modelo denominado pague-por-uso, esse tipo de nuvem é fornecido por organizações públicas ou grandes empresas que possuem grande capacidade de processamento e armazenamento.

- Nuvem Comunitária: segundo Veras e Tozer (2012), para a Nuvem Comunitária a infraestrutura de Computação em Nuvem é compartilhada por diversas empresas e dá suporte a comunidades com interesses em comum. Podendo ser administrada

pelas empresas que fazem parte da comunidade ou ser terceirizada, também pode funcionar dentro ou fora das organizações.

- Nuvem Híbrida: conforme Veras e Tozer (2012), a nuvem híbrida possui a infraestrutura composta de duas ou mais nuvens que apesar disto continuam sendo entidades únicas, entretanto são conectadas por meio de tecnologias proprietárias ou padronizadas que permitem a portabilidade de dados e aplicações.

- Modelo de serviço IaaS: segundo Antonopoulos e Gillam (2010) IaaS é a entrega de Infraestrutura de computadores como um serviço. Além de alta flexibilidade, outra grande vantagem do IaaS é o pagamento por uso do serviço. Isto permite aos clientes aumentarem a demanda assim como crescem os negócios. Além disso, outra importante vantagem é a de estar sempre utilizando o que há de mais novo em tecnologia. Com isso os clientes terão mais rapidamente seus serviços funcionando e iniciará mais rapidamente a comercialização.

- Tolerância a falhas: segundo Veras e Tozer (2012), a tolerância a falhas é a capacidade de um sistema continuar operando mesmo que alguns de seus componentes venham a falhar.

- Disponibilidade: segundo Veras e Tozer (2012), a disponibilidade está relacionada a entrega das informações adequadas quando estas são exigidas pelo processo de negócio, além disso a disponibilidade também está associada ao cuidado com os recursos necessários e as capacidades associadas a eles.

- *Load Balancing*: segundo Antonopoulos e Gillam (2010), *Load Balancing* ou Balanceamento de Carga é utilizado para evitar gargalos do sistema provenientes de cargas desequilibradas. O Balanceamento de Carga também leva em consideração a implantação de *FailOver*, para que um serviço possa continuar após falha de um ou mais componentes. O Balanceador de Carga precisa fornecer mecanismos pelos quais as instâncias de aplicações possam ser provisionadas ou não, de forma automática sem que isso altere as configurações da rede.

1.9 CRONOGRAMA

No cronograma representado no Quadro 1 são apresentas as datas previstas para a realização das atividades, bem como o prazo de início à conclusão das mesmas. Os quadros sombreados representam o período proposto para a realização de cada atividade, e os quadros marcados com X representam o período em que foi realizada cada atividade.

Atividades	**Organização do Tempo/2013**				
	Jul	**Ago**	**Set**	**Out**	**Nov**
Estudar e fundamentar computação em nuvem.	X	X			
Estudar os modelos de serviço para computação em nuvem.	X	X			
Estudar ferramentas *Open source* de computação em nuvem voltadas para serviços IaaS.	X	X			
Estudo de trabalhos relacionados a este tema de pesquisa.	X	X			
Escolher e implantar duas ferramentas dentre as estudadas.			X		
Definir um conjunto de características a serem avaliadas nas ferramentas implementadas.			X		
Avaliar e comparar as ferramentas em um cenário com estações de trabalho.				X	
Documentar resultados.				X	
Analisar os resultados.					X
Apresentar os resultados obtidos.					X
Publicar um artigo relacionado a este trabalho.					X

Fonte: Hentges, Thomé, Griebler, 2013.

Quadro 1: Cronograma de atividades

1.10 RECURSOS

1.10.1 Recursos Humanos

Os recursos humanos envolvidos foram o Professor Orientador, acadêmicos do Curso de Tecnologia em Redes de Computadores.

1.10.2 Recursos Materiais

Os recursos de materiais utilizados foram: computadores, folhas, *pendrive*, livros e material didático em geral.

1.10.3 Recursos Institucionais

Os recursos institucionais utilizados foram: laboratórios de informática, *Internet*, biblioteca e central de cópias da SETREM.

1.11 ORÇAMENTO

Para realizar este projeto foi necessário da quantia exposta no Quadro 2, que está organizada da seguinte forma:

- A primeira coluna exibe o que gerou tal despesa.
- Na segunda coluna é exibida a quantidade.
- A terceira coluna exibe o valor.
- A quarta coluna exibe o valor total da quantidade.
- Na última linha da tabela está exibido o valor total.

Referente	Quantidade	Valor Unitário	Valor Total
Impressão	1000	R$ 0,15	R$ 150,00
Transporte	25	R$ 3,05	R$ 76,25
Encadernação espiral	5	R$ 3,00	R$ 15,00
Encadernação capa dura	2	R$ 50,00	R$ 100,00
Horas trabalhadas	200	R$ 40,00	R$ 8.000,00
		Total Geral	R$ 8.341,25

Fonte: Hentges, Thomé, Griebler, 2013.

Quadro 2: Orçamento

CAPÍTULO 2: REFERENCIAL TEÓRICO

O capítulo 2 a seguir apresenta os referenciais teóricos utilizados para a realização do presente trabalho. Expõe conceitos básicos de Redes de Computadores, Computação em Nuvem (Tipos de Nuvem e Modelos de Serviço), Virtualização e Ferramentas de Virtualização, Sistemas Distribuídos (Arquiteturas Distribuídas), Segurança em Computação em nuvem privada e também Ferramentas para Computação em Nuvem.

2.2 REDES DE COMPUTADORES

Segundo Torres (2001), as redes de computadores surgiram por causa da necessidade de trocar informações, com elas é possível ter acesso a dados que estão localizados fisicamente longe. A maior rede existente é a *Internet*, na qual grande parte das pessoas deseja ter acesso.

Segundo Tanenbaum (2003), rede de computadores é um conjunto de computadores que estão conectados e podem trocar informações. O método como é feita a conexão é diversificado, pode ser utilizado fio de cobre, fibra óptica, micro-ondas, infravermelho e satélites de comunicação. Além disso, existem muitos tamanhos, modelos e formas que pode ser constituída uma rede de computadores.

2.2.1 Modelo OSI

Segundo Tanenbaum (2003), o modelo OSI fundamenta-se em uma proposta que foi criada pela ISO (*International Standards Organization*) para que houvesse uma padronização para os protocolos empregados nas camadas de rede.

O modelo OSI (*Open Systems Interconnection*) é responsável por fazer a conexão de sistemas que estão abertos à comunicação com outros sistemas.

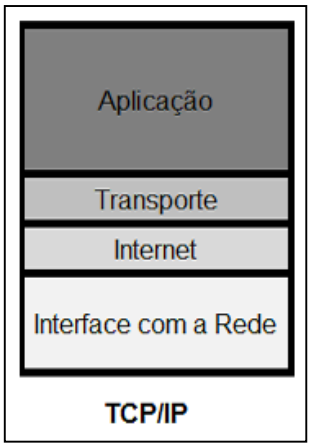
O modelo OSI possui sete camadas: camada física, enlace de dados, rede, transporte, sessão, apresentação e camada de aplicação. Esse modelo também especifica o que cada camada deve fazer.

2.2.2 TCP/IP

Segundo Torres (2001), o protocolo TCP/IP é atualmente o mais usado nas redes locais por causa da *Internet*. Afinal, ele foi criado para ser usado na *Internet*, mas hoje em dia até mesmo os sistemas operacionais de rede suportam o protocolo TCP/IP. Ele tem como vantagem ser roteável, pois foi criado pensando em redes grandes e de longa distância, já que nessas redes podem haver vários caminhos para o dado seguir e atingir o computador de destino.

Conforme Torres (2001), o protocolo TCP/IP tem arquitetura aberta, com isso, qualquer fabricante pode utilizar sua própria versão do TCP/IP no seu sistema operacional, não é necessário pagar direitos autorais para isso. Assim, todos acabaram adotando o TCP/IP, transformando-o em um protocolo universal que possibilita que todos os sistemas comuniquem-se entre si.

A Figura 1, a seguir mostra a arquitetura do protocolo TCP/IP: suas quatro camadas.



Fonte: Torres, 2001.
Figura 1: Arquitetura TCP/IP.

2.3 COMPUTAÇÃO EM NUVEM

Segundo Velte et al. (2010), a computação em nuvem está em toda parte, observando qualquer revista de tecnologia ou site de TI (*Technology Information*) é possível ver algum tópico falando de computação em nuvem. Porém, ainda nem todos os profissionais de TI possuem a mesma visão sobre o que é a computação em nuvem, o que justifica- se pelo termo Computação em Nuvem ter sido usado em exagero e aplicado a quase tudo no mundo da informática.

> Com a computação em nuvem, os programas de *software* que você usa, não são executados a partir do seu computador pessoal, eles são armazenados em servidores e acessados via *Internet*. Se o seu computador trava, o *software* ainda está disponível para outras pessoas utilizarem. O mesmo vale para os documentos criados, eles são armazenados em um conjunto de servidores acessados via *Internet*. Qualquer pessoa com permissão, não só pode acessar os documentos, mas também pode editar e colaborar com esses documentos em tempo real (MILLER, 2009, p. 13).

Conforme Velte et al. (2010), a computação em nuvem surgiu com a promessa de reduzir os custos operacionais, custo de capital e também auxiliar os setores de TI, para que possam focar em projetos estratégicos ao invés de preocupar-se com o *datacenter*.

> A chave para a definição de computação em nuvem é a "nuvem" em si. Para os nossos propósitos, a nuvem é um grande grupo de computadores interconectados. Esses computadores podem ser computadores pessoais ou servidores de rede, pois eles podem ser públicos ou privados (MILLER, 2009, P.14).

2.3.1 Características

Segundo CSA- *Cloud Security Alliance* (2011), os serviços em nuvem oferecem cinco características importantes, são elas: auto atendimento sob demanda, amplo acesso à rede, grupo de recursos, elasticidade e serviços mensuráveis.

- Auto atendimento sob demanda, segundo CSA (2011), é a possibilidade do cliente poder de forma parcial ter a capacidade de computação a sua disposição como, por exemplo, a hora do servidor e o armazenamento em rede, de maneira

automática sem que haja a necessidade de interação humana com o provedor do serviço;

- Conforme CSA (2011), o amplo acesso à rede se trata da questão de que os recursos devem estar disponíveis através da rede e possam ser acessados por qualquer dispositivo que utilize *softwares* tradicionais ou baseados em nuvem;

- Grupo de recursos, conforme CSA (2011) é a possibilidade dos recursos do provedor serem agrupados para que possam atender diversos clientes utilizando-se de um modelo multiusuário, tendo os recursos físicos e virtuais atribuídos de forma dinâmica conforme a demanda do cliente. O cliente na maior parte das vezes não possui conhecimento de onde estes recursos (armazenamento, memória, máquinas virtuais, etc.) estão exatamente localizados;

- De acordo com o CSA (2011), a elasticidade é a possibilidade de os recursos serem fornecidos de forma rápida e elástica, podendo ser automaticamente. Para o cliente os recursos disponíveis podem parecer limitados, mas com a possibilidade de se adquirir qualquer volume a qualquer momento;

- Serviços mensuráveis para o CSA (2011) é a possibilidade dos serviços em nuvem controlar e aprimorar o uso dos recursos, aplicando a capacidade de medição ao nível de abstração mais adequada para cada tipo de serviço. A mensuração dos serviços permite que o uso de recursos possa ser monitorado, controlado e reportado, disponibilizando para o cliente e ao provedor de serviço maior transparência.

2.3.2 Arquitetura

Segundo Buyya et al. (2009) apud Sousa, Moreira e Machado (2009), a computação em nuvem tem sua arquitetura fundamentada em camadas, e cada camada possui uma característica própria ao disponibilizar os recursos para as aplicações.

Conforme Sousa, Moreira e Machado (2009), a camada é uma divisão de componentes de *hardware* e *software*. Sendo que os recursos podem ser arranjados para que executem uma determinada tarefa do sistema. Cada camada tem seu gerenciamento e monitoramento de maneira individual com o objetivo de melhorar o desempenho sem que as mesmas afetem entre si.

Segundo Ruschel, Zanotto e Mota (2010), a camada mais baixa corresponde à infraestrutura e através dela, são disponibilizados serviços de rede e armazenamento. É nesta camada que se encontram os *datacenters*, *clusters* e demais recursos de *hardware*.

Segundo Sousa, Moreira e Machado (2009), a camada seguinte é a camada de *middleware* que gerencia a infraestrutura e fornece núcleo lógico da nuvem. Este serviço possui gerenciamento dos SLA's (*Service- Level Agreement*), serviços de cobrança, serviços de gerenciamento de virtualização, entre outros.

Conforme Sousa, Moreira e Machado (2009), acima da camada *middleware* localiza-se a camada que é responsável por fornecer suporte, ferramentas, ambientes para o desenvolvimento de aplicações. Essa camada é mais utilizada por usuários que desenvolvem aplicações para computação em nuvem e não por usuários finais.

De acordo com Sousa, Moreira e Machado (2009), a última camada é a das aplicações em nuvem, essa é a camada utilizada pelo usuário, pois é através dela que é possível utilizar dos aplicativos.

Conforme Sousa, Moreira e Machado (2009), as camadas mais baixas têm como objetivo controlar a escalabilidade, disponibilidade e alto desempenho. Algumas soluções incluem uma camada extra que fornece algum tipo de adaptação para as mesmas, realizando tudo de forma automática ou semiautomática para diminuir a necessidade do contato humano para realizar determinada atividade.

A Figura 2 demonstra as quatro camadas da arquitetura de computação nuvem segundo Vecchiola, Chu e Buyya (2009).

Adaptado de: Vecchiola, Chu e Buyya, 2009, p.4.
Figura 2: Arquitetura da computação em nuvem.

2.3.3 Tipos de Nuvem

As empresas podem definir se suas aplicações serão implementadas em nuvens Públicas, Privadas, Comunitárias ou Híbridas, porém isso não irá definir a localização das aplicações.

2.3.3.1 Nuvens Públicas

Segundo Marks e Lozano (2010), as nuvens públicas são multiusuários e tendem a concentrar-se em camadas específicas. Esse tipo de nuvem é um setor muito importante e fonte de inovação, porém não tende a possuir todas as áreas de informática.

> As nuvens públicas são executados por terceiros e executam aplicações de diferentes clientes. Tendem a ser misturados nos servidores da nuvem, sistemas de armazenamento e redes. As nuvens públicas são mais frequentemente hospedadas fora das instalações dos clientes, fornecendo uma maneira de reduzir o risco para o cliente, assim como o custo, proporcionando uma extensão, mesmo que temporária para a infraestrutura corporativa (SUN MICROSYSTEMS, 2009, p.9).

Segundo Escalante (2010), os recursos de computação em uma nuvem pública são gerenciados por terceiros, e aplicações de diferentes clientes tendem a

estarem “misturadas” no mesmo servidor, sistema de armazenamento e rede, sendo estes recursos acessados via *Internet* através de serviços *Web*.

2.3.3.2 Nuvens Privadas

Segundo Marks e Lozano (2010), nuvens privadas assim como o nome indica, são nuvens funcionais que são operadas e de uso restrito de determinada empresa/cliente. E vem se tornando crescente a oferta de *softwares* e serviços projetados para esse tipo de nuvem.

> As nuvens privadas são construídas para o uso exclusivo de um cliente, proporcionando o máximo de controle sobre os dados, segurança e qualidade do serviço. A empresa possui a infraestrutura, e tem controle sobre como as aplicações são implementadas nele. As nuvens privadas podem ser implantadas em um *datacenter* da empresa, e podem também ser implantados em uma instalação de terceiros (SUN MICROSYSTEMS, 2009, p.10).

Segundo Escalante (2010), nuvens privadas são construídas para uso exclusivo de um cliente, lhe permitindo controle total sobre seus dados, maior segurança e qualidade do serviço, podendo essas nuvens serem construídas pela própria empresa ou por um provedor.

2.3.3.3 Nuvens Comunitárias

Segundo Marks e Lozano (2010), nuvens comunitárias são nuvens públicas organizadas entre concorrentes, empresas que trabalham em determinado mercado. As nuvens comunitárias são capazes de fornecer recursos como governança, auditoria e segurança, possuindo diversos serviços e localizados de forma que ajude a alcançar uma quantidade maior de clientes interessados.

De acordo com Ribas (2012), o modelo de nuvem comunitária promove o compartilhamento da infraestrutura entre diversas organizações, onde todos os envolvidos devem concordar com requisitos de segurança, políticas de segurança e sobre a flexibilidade. O gerenciamento desta nuvem pode ser realizado por alguma empresa que compartilha está nuvem ou até mesmo por terceiros.

2.3.3.4 Nuvens Híbridas

Segundo Marks e Lozano (2010), as nuvens híbridas são uma combinação de todos os outros tipos de nuvens. Existem plataformas que permitem as empresas criarem suas próprias nuvens híbridas fazendo uma combinação das demais e fazendo seu gerenciamento a partir de um único lugar e ao mesmo tempo. Isso permite que a empresa utilize a ferramenta mais adequada para cada setor.

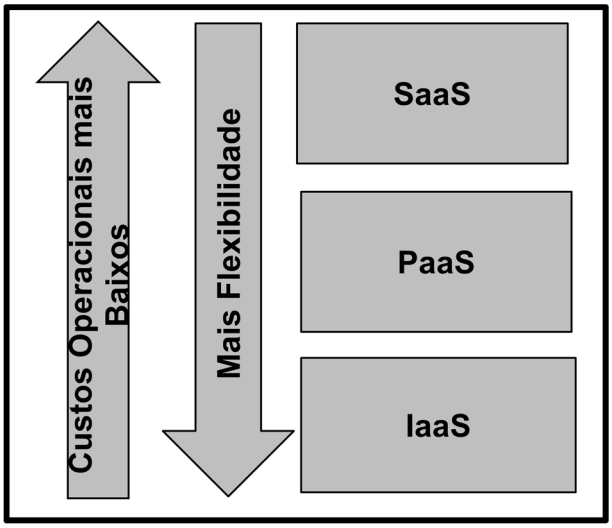
De acordo com Sun Microsystems (2009), uma nuvem hibrida é quando ocorre a mescla entre uma nuvem privada e uma nuvem pública. Isto pode ajudar a fornecer os serviços de maneira adequada em uma nuvem privada, já que os mesmos podem ser divididos entre as duas nuvens, aumentando o desempenho e podendo ajudar em momentos de picos de carga de trabalho.

“Um ambiente de nuvem híbrida combina vários modelos de nuvem pública e privada. Nuvens híbridas introduzem a complexidade de determinar a distribuição das aplicações entre uma pública e uma nuvem privada” (ESCALANTE, 2010, p. 7).

2.3.4 Modelos de Serviço

Segundo Rimal, Choi e Lumb (2009), existem diferentes categorias de serviços que podem ser oferecidos em nuvem, dentre eles podem ser oferecidos infraestrutura, plataforma, aplicações, *hardware*, desenvolvimento negócios, *framework* e organização. Estes serviços são entregues ao consumidor em tempo real através da *Internet*.

Os principais modelos de serviços da computação em nuvem são: IaaS (*Infrastructure as a Service*), Paas (*Platform as a Service*) e SaaS (*Software as a Service*). A Figura 3 demonstra a relação entre esses modelos de serviços. Sendo o IaaS a base que fornece a infraestrutura ao PaaS, esta é a plataforma onde funciona o último nível, o SaaS que são os serviços e as aplicações. Mostra-se também que quanto maior o nível menor são os custos operacionais e quanto menor o nível maior a flexibilidade.



Adaptado de: Jesús, 2012, p.8.
Figura 3: Relação entre modelos de serviços.

2.3.4.1 Modelo IaaS

Para Ribas (2012), o modelo IaaS é o responsável pela infraestrutura necessária para o modelo PaaS e o SaaS. Mas o principal objetivo do IaaS é tornar a utilização dos recursos computacionais mais fácil e acessível. Pois esse modelo possui uma interface única para administrar infraestruturas e fazer a interação com diferentes equipamentos de uma maneira simples e transparente.

Conforme Tomsho (2011), o modelo IaaS fornece infraestrutura computacional para os clientes, como por exemplo, uma empresa necessita de 100Gb de armazenamento. Se a mesma possui servidores próprios, ela irá comprar um HD (*Hard Disk*) adicional para suprir. Utilizando-se do modelo IaaS, ela irá pagar por esse armazenamento que será na nuvem e o mesmo serve para o processamento, se for preciso mais, ela irá pagar por esta carga extra de processamento. Além disso, a nuvem pode fornecer um Sistema Operacional operante, que pode ser acessado via *Terminal Service* ou semelhante. Para administrar a nuvem é realizado o acesso remoto através da *Internet*. O mesmo é utilizado por empresas para suprir demandas de TI.

Conforme Voras et al. (2011), o modelo IaaS é implementado e oferecido por provedores como a Amazon (Amazon EC2), Rackspace (Rackspace *Cloud*) e FlexiAnt (FlexiScale).

2.3.4.2 Modelo PaaS

Segundo Ribas (2012), o modelo PaaS é uma plataforma que fornece serviços que suportam o desenvolvimento de aplicações e por isso é utilizado pelos usuários para desenvolver aplicações e poder deixar disponível na nuvem. Sendo assim, o objetivo desse modelo é facilitar o desenvolvimento de aplicações dedicadas aos usuários das nuvens, o PaaS agiliza esse processo, pois possui uma infraestrutura de integração que permite implementar e testar aplicações na nuvem.

De acordo com Tomsho (2011), o modelo PaaS é uma plataforma de desenvolvimento de aplicações, que concede ao cliente todas as ferramentas necessárias a ele para o desenvolvimento, sempre disponibilizando as atualizações para as ferramentas. Pode ser utilizado por empresas de desenvolvimento de *software* e de criação de multimídia. Devido à plataforma ser baseada em infraestrutura externa as empresas, a mesma acessará através da *Internet* as estações de desenvolvimento, utilizando-se da ferramenta oferecida pelo fornecedor.

Conforme Voras et al. (2011), o modelo PaaS fornece para os desenvolvedores uma plataforma de desenvolvimento e implantação de aplicações ou alteração de aplicações já existentes. Como exemplos de plataforma podem ser citadas plataformas oferecidas pela Google (Google App Engine11), Microsoft (Windows Azure12) e Salesforce.com (Force.com13).

2.3.4.3 Modelo SaaS

Conforme Ribas (2012), o modelo SaaS oferece sistemas de *software* disponíveis aos usuários a partir da *Internet* com finalidades especificas. Nesse modelo, o cliente não necessita de licenças ou instalar a aplicação em seu próprio equipamento, pois no SaaS as aplicações são disponibilizadas via *browser,* podendo ser acessadas de qualquer lugar e a todo momento.

De acordo com Tomsho (2011), o modelo SaaS disponibiliza aos usuários serviços e aplicações que são acessadas pela *Internet*, normalmente através de um *browser*. São utilizadas tanto por empresas como por usuários comuns que utilizam,

por exemplo, o Gmail oferecido pela Google. Os serviços podem ser gratuitos ou pagos, sendo que os pagos define-se o preço através da quantidade de usuários que irão utilizar aquela aplicação. Podem ser fornecidos diferentes serviços, desde *email* e planilhas eletrônicas até bancos de dados.

Conforme Voras et al. (2011), o modelo SaaS é oferecido para os usuários finais, são amplamente utilizados como por exemplo serviços de *email* (exemplos: Google *Mail*, Hotmail, Yahoo *Mail*), aplicativos para compartilhamento de fotos (por exemplo, Picasa e Flickr), compartilhamento de vídeo (por exemplo Youtube e Vimeo), aplicações de escritório (por exemplo Apps Google, Microsoft *Exchange Hosted Services*, Microsoft *Office Live*).

2.3.5 Formas de Acesso

Para que se possa acessar as ferramentas, podem ser utilizadas duas maneiras, sendo uma o SSH e a outra através de uma interface *web,* utilizando-se do protocolo HTTP.

2.3.5.1 SSH (Secure Shell)

Segundo Barrett et al. (2005), o SSH é um protocolo que especifica como deve ser realizada uma comunicação segura através de uma rede. Ele provê a autenticação e comunicação criptografada. O protocolo SSH envolve autenticação, criptografia e integridade de dados transmitidos através de uma rede.

Conforme Barrett et al. (2005), com a autenticação o protocolo SSH determina a identidade de alguém. Se tentar fazer *logon* em uma conta em um computador remoto, SSH pede por uma prova digital de sua identidade. Se passar no teste, pode conectar, caso contrário, rejeita a conexão SSH.

De acordo com Barrett et al. (2005), a criptografia no protocolo SSH embaralha os dados de modo que é ilegível, exceto para os destinatários. Isso protege os seus dados à medida que passa através da rede. E a integridade garante

que os dados que trafegam pela rede cheguem inalterados. Se um terceiro capturar e modificar os dados que estão sendo trafegados, o SSH detecta este fato.

2.3.5.2 HTTP (Hypertext Transfer Protocol)

Segundo Torres (2001), o HTTP (*Hypertext Transfer Protocol*) é um protocolo de comunicação que trabalha na camada de aplicação do modelo OSI, utiliza por padrão a porta 80 e é utilizado na comunicação de sites. O HTTP existe pela necessidade de trocar informações pela *Internet*, como uma forma de padronização para a comunicação entre os clientes/servidores ligados a *Internet*.

> O protocolo de transferência de arquivo em toda a WWW é o HTTP. Ele especifica as mensagens que os clientes podem enviar aos servidores e que respostas eles receberão. Cada interação consiste em uma solicitação ASCII, seguida por uma resposta RFC 822 semelhante ao MIME (TANENBAUM, 2003, p. 493).

2.4 VIRTUALIZAÇÃO

A virtualização é uma tecnologia que permite ao usuário emular diferentes sistemas operacionais, com diferentes arquiteturas. Onde cada um tem sua função específica e todos executam no mesmo *hardware*. Os recursos de *hardware* são divididos de acordo com a necessidade de uso de cada sistema operacional.

Segundo Hess e Newman (2009) virtualização se refere à abstração física dos recursos computacionais. Significa que os recursos físicos que são alocados para uma máquina virtual são abstraídos dos limites físicos. Discos virtuais, redes virtuais, LAN's virtuais, *switches* virtuais, CPU virtual, e memória virtual, são todos recursos do sistema do computador físico.

Conforme Veras (2011), a virtualização permite a um *datacenter* alocar os recursos disponíveis de acordo com a demanda. Essa otimização melhorou muitos aspectos, como a redução do espaço físico, menor consumo de energia, melhora na recuperação a desastres e aumento no nível de disponibilidade das aplicações.

2.4.1 Tipos de virtualização

Existem dois tipos de virtualização que podem ser citados, a paravirtualização e a virtualização plena, sendo cada método utilizado de acordo com a capacidade de *hardware* do usuário, já que funcionam de maneiras distintas.

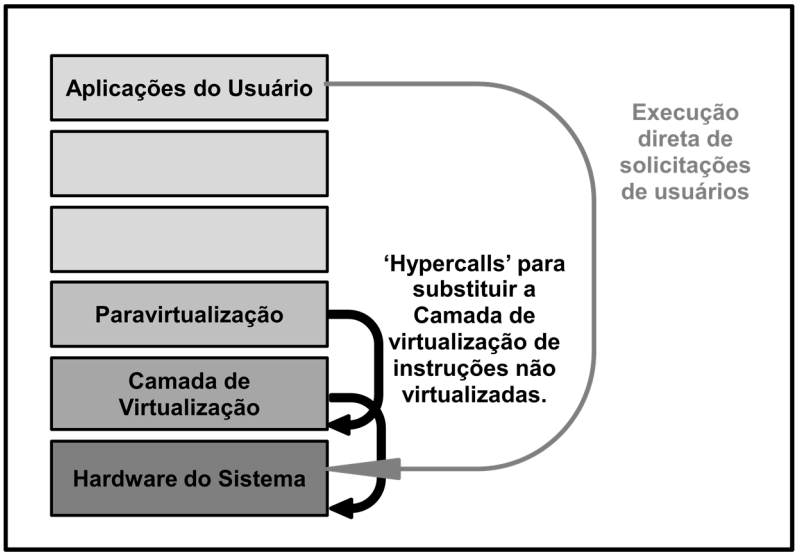
2.4.1.1 Paravirtualização

Segundo Barrett e Kipper (2010), paravirtualização envolve a modificação do *kernel* do sistema operacional virtualizado. O *kernel* do sistema operacional age como uma ponte entre as aplicações e o processamento é feito pelo *hardware*. Paravirtualização substitui as instruções não virtualizadas com chamadas que comunicam diretamente com a camada de virtualização do *hypervisor*.

Conforme Vugt (2008) para que se comunique com a camada de virtualização e possa fazer suas requisições, a paravirtualização efetua a alteração do *kernel* do sistema operacional virtualizado. Para que o virtualizador realize a transferência dessas requisições para o *hardware*.

Conforme Vugt (2008), a paravirtualização requer a modificação do sistema operacional. Esta versão gera instruções que são de fácil entendimento do virtualizador, que interpreta as instruções virtuais e transmite para o *hardware*. Nesse modo de virtualização o sistema operacional tem conhecimento que é virtualizado para que ele mesmo gere instruções otimizadas para seu uso.

A Figura 4 descreve o funcionamento da paravirtualização. As aplicações hospedadas no sistema operacional virtualizado fazem as chamadas que são entregues ao *hardware*. Enquanto isso, o sistema operacional envia as *hypercalls* para a camada de virtualização que substitui as instruções não virtualizadas, para então repassa-las para o *hardware*.



Adaptado de: Vmware, 2007, p.5.
Figura 4: Funcionamento paravirtualização.

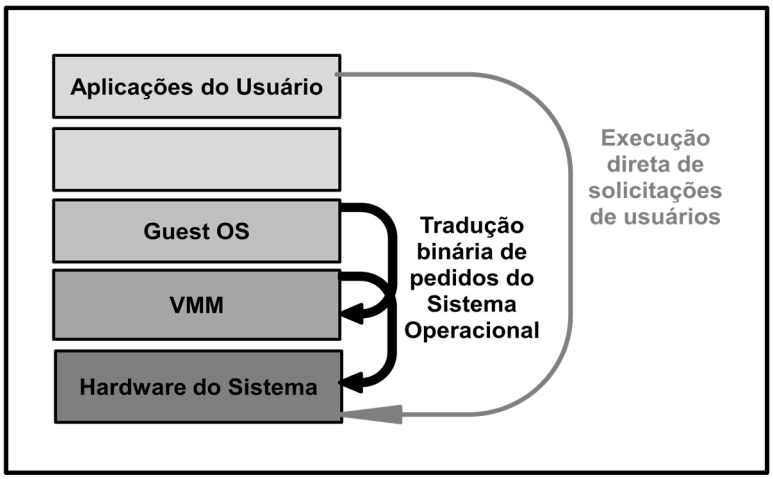
2.4.1.2 Virtualização plena

Conforme Barrett e Kipper (2010), a virtualização plena necessita de *hardware* específico para que possa ser executada. O processador necessita ter instruções de virtualização para que possa entender e se comunicar com o sistema operacional, possibilitando ao sistema operacional usufruir da melhor maneira possível os recursos computacionais.

Conforme Barrett e Kipper (2010) na virtualização plena a VM simula o *hardware* necessário para permitir que um sistema operacional possa executar isoladamente. A virtualização plena oferece o melhor isolamento e segurança para as VM's e simples migração e portabilidade para os sistemas operacionais hospedados.

Segundo Vugt (2008), a virtualização plena permite o uso de um sistema operacional como se estivesse instalado diretamente no *hardware*. Porém, ele exige que os processadores ofereçam um recurso que permite este tipo de virtualização. Devido a esta modificação nos processadores, é possível que estes sistemas virtualizados trabalhem de maneira mais eficiente que na paravirtualização.

A Figura 5 descreve o funcionamento da virtualização plena, onde as aplicações fazem as chamadas que executam diretamente no processador para melhor desempenho. Enquanto isso, as instruções do sistema operacional virtualizado são passadas ao virtualizador que as testa, e após isso passa as mesmas para o *hardware*, que irá executar.



Adaptado de: VMware, 2007, p.4.
Figura 5: Funcionamento virtualização plena.

2.4.2 Ferramentas de virtualização

Existem diversas ferramentas utilizadas para virtualização. Neste item serão citadas as mais frequentemente utilizadas, são elas: o VMware que permite a virtualização de qualquer sistemas operacional, o Xen *Server* mais utilizado para o gerenciamento e virtualização de servidores, o KVM que permite virtualizar sistemas operacionais Windows e Linux e gerenciar as máquinas virtuais criadas e o Mycrosoft Hyper-V que é a ferramenta de virtualização disponibilizada pela Microsoft.

2.4.2.1 VMware

Uma das ferramentas mais utilizadas na área da virtualização é o VMware, possui diversas versões que possibilitam virtualizar qualquer sistema operacional. Além disso, permite empregar funcionalidades necessárias que auxiliam o bom desempenho e gerenciamento de servidores que possam estar executando nele.

Assim como, recursos fundamentais para alto desempenho, como tolerância a falhas e balanceamento de recursos dinâmico.

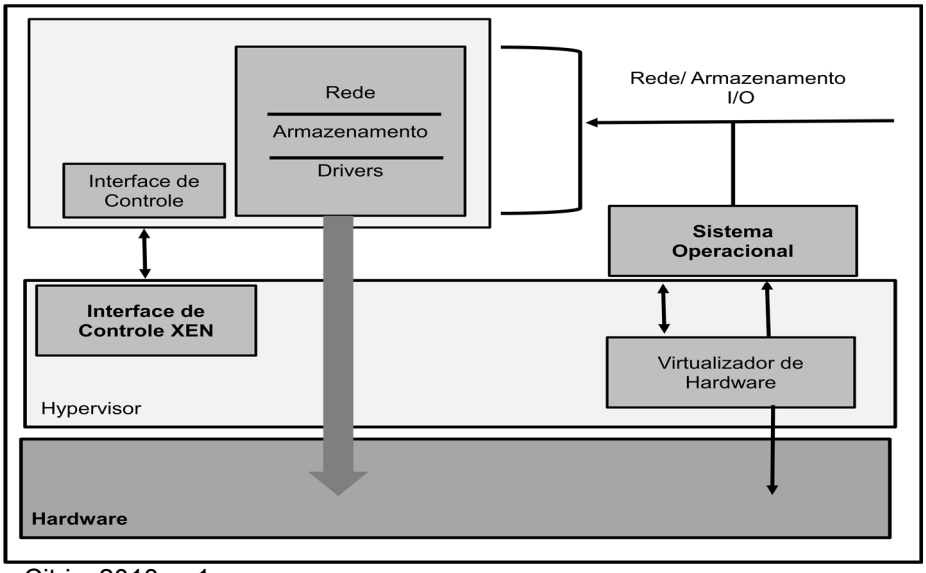
Segundo Haletky (2011) VMware ESX e ESXi, oferecem incríveis funcionalidades para virtualização como: tolerância a falhas, balanceamento de recursos dinâmico, melhor *hardware* para máquina virtual, rede virtual e *failover*.

Conforme VMware (2013), sua virtualização funciona inserindo uma camada de *software* de forma direta ao *hardware* ou ao sistema operacional instalado no *host*. Conta com um monitor de máquinas virtuais, que tem o objetivo de alocar os recursos de *hardware* dinamicamente e transparente. Com isso, podem ser executados vários sistemas virtuais, em um único computador, compartilhando os mesmos recursos.

2.4.2.2 Xen Server

Conforme Tosatto (2012) Xen *Server* é uma ferramenta que permite gerenciar servidores virtualizados. Pode usar para virtualizar tanto servidores Microsoft Windows quanto Linux facilmente e eficientemente. Xen *Server* proveem uma plataforma de virtualização para nuvem que contém todas as necessidades para criar e administrar uma infraestrutura virtual. Possibilita atualmente capacidade de 500 máquinas virtuais por virtualizador. Possui uma gama de recursos, que permitem automatizar processos a serem executados e otimizar o desempenho.

A Figura 6, demonstra a estrutura do Xen *Server*, sendo a camada mais baixa a de *hardware*, separada pela camada do virtualizador Xen *Server*, que virtualiza o *hardware* para receber as instruções do sistema operacional e executá-lo através das instruções específicas para virtualização que estão contidas no processador. Além disso, o virtualizador conta com uma interface controladora de rede, armazenamento e *drivers* que controla estes componentes de *hardware*.



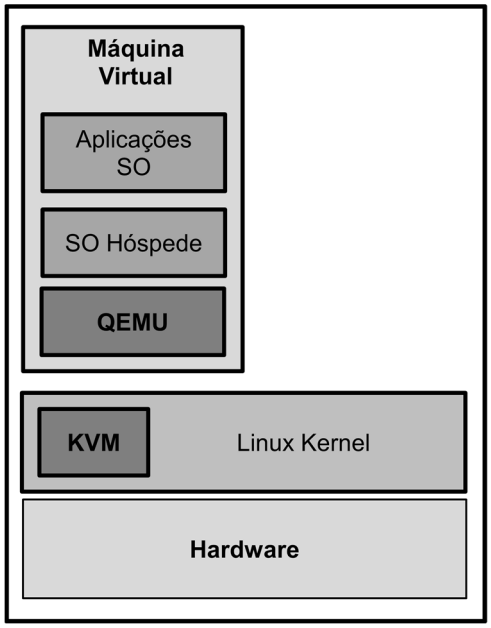
Adaptado de: Citrix, 2013, p.1.
Figura 6: Estrutura do Xen *Server*.

2.4.2.3 KVM

O KVM (*Kernel-based Virtual Machine*) possibilita assim como outras ferramentas de virtualização a criação e gerenciamento de maneira fácil de suas VM's (*Virtual Machine*). Possibilita a instalação de sistemas operacionais Windows e Linux e a virtualização plena. Além disso, possui recursos para fazer o gerenciamento das máquinas virtuais.

Segundo KVM (2013) KVM é uma solução para virtualização plena para Linux que suportam a arquitetura x86. Ele consiste em carregar o módulo do *kernel* (kvm.ko), que providencia a infraestrutura de virtualização do núcleo e um módulo específico para o processador.

A Figura 7 demonstra a estrutura do KVM, sendo a camada mais baixa a de *hardware* que precisa ter as instruções para suportar este tipo de virtualização. Depois se encontra o virtualizador KVM junto ao *kernel* Linux, utilizando-se do emulador QEMU, ele consegue criar as máquinas virtuais rodando como se fossem processos.



Adaptado de: IBM, 2013, p.1.
Figura 7: Estrutura do KVM.

2.4.2.4 Windows Hyper- V

Segundo Finn, Lownds (2011), Hyper-V disponibiliza uma robusta, confiável e segura plataforma que isola as aplicações e os sistemas operacionais de seu *hardware*, reduzindo drasticamente a complexidade de implementar e testar serviços contínuos para negócios.

Conforme Finn, Lownds (2011), os componentes de integração do Hyper-V podem ser vistos como um conjunto de dispositivos de *driver*. Quando instalados eles oferecem a melhor performance possível para a máquina virtual. Eles ainda permitem usar adicionalmente tipos de dispositivos virtuais que não são disponíveis em uma máquina virtual emulada. Ele também disponibiliza serviços que permitem a máquina virtual integrar-se com o *host* onde o Hyper-V está instalado.

2.5 SISTEMAS DISTRIBUIDOS

Sistemas distribuídos consistem em diversos computadores interligados a uma rede compartilhando ou executando tarefas em comum. As tarefas executam de forma transparente, onde para o usuário parece um sistema único, no qual executam as tarefas.

Conforme Couloris, Dollimore e Kindberg (2007), um sistema distribuído se trata de componentes localizados em computadores interligados que se comunicam e combinam as ações por mensagens. Isto apresenta algumas características dos sistemas distribuídos: concorrência de componentes, falhas de componentes e falta de um relógio global.

Segundo Tanenbaum e Steen (2007), um sistema distribuído é um conjunto de computadores independentes que se apresenta aos usuários como um coerente e único sistema, ou seja, consiste em componentes autônomos que os usuários acreditam se tratar de um único sistema. Mas para que isso funcione os componentes devem colaborar.

2.5.1 Arquiteturas distribuídas

Modelos de arquitetura de sistemas distribuídos: segundo Coulouris, Dollimore e Kindberg (2007), um modelo de arquitetura de sistema distribuído simplifica e abstrai funções dos componentes individuais de um sistema distribuído e considera o posicionamento desses componentes na rede de computadores, para poder definir padrões para a distribuição de dados e de carga de trabalho. O modelo de arquitetura também considera o inter-relacionamento entre os componentes (papéis funcionais, padrões de comunicação).

De acordo com Coulouris, Dollimore e Kindberg (2007), os modelos de arquitetura de um sistema distribuído também chamados modelos arquitetônicos são aqueles que descrevem a estrutura de organização dos componentes de um sistema. Tem a função de descrever como são estabelecidas as comunicações entre os nós.

Conforme Coulouris, Dollimore e Kindberg (2007), o modelo de arquitetura envolve a posição e a forma de relacionamento que existe entre as partes envolvidas. Como exemplo, podem ser citados o modelo cliente-servidor e o modelo *peer-to-peer.* O modelo cliente-servidor pode sofrer alterações em determinadas partes, como por exemplo: Através da partição dos dados, ou pela replicação de

servidores, pelo uso de código e de agentes móveis, ou pelo uso de servidores proxies para colocação de dados em cache.

2.5.1.1 Cluster

Segundo Veras e Tozer (2012), um *cluster* consiste em um aglomerado de computadores que estão conectados através de uma rede interna, confiável e de alta velocidade, onde estes computadores executam em conjunto as mesmas tarefas com o objetivo de aumentar a produtividade.

> A configuração de um típico *cluster* é um grupo de computadores conectados através de um *switch* local e acessados a partir do nó *frontend*. Nesta configuração, os trabalhos são recebidos pelo nó *frontend*, que utiliza uma tarefa local para encaminhar os trabalhos para nós internos (WILKINSON, 2010, p.67).

Conforme Keswani (2008), *cluster* é um grupo de computadores conectados utilizando uma rede de alta velocidade e costumeiramente rodando o mesmo sistema operacional e o mesmo conjunto de ferramentas. Um *cluster* possui alta escalabilidade, o que permite a adição e até mesmo a remoção de nós, sem a necessidade de parar todo o sistema, sendo preciso apenas realizar alguma ação no nó mestre.

Segundo Keswani (2008), *clusters* podem ser classificados em três tipos: *cluster* de alta performance, *cluster* com balanceamento de carga e *cluster* de alta disponibilidade.

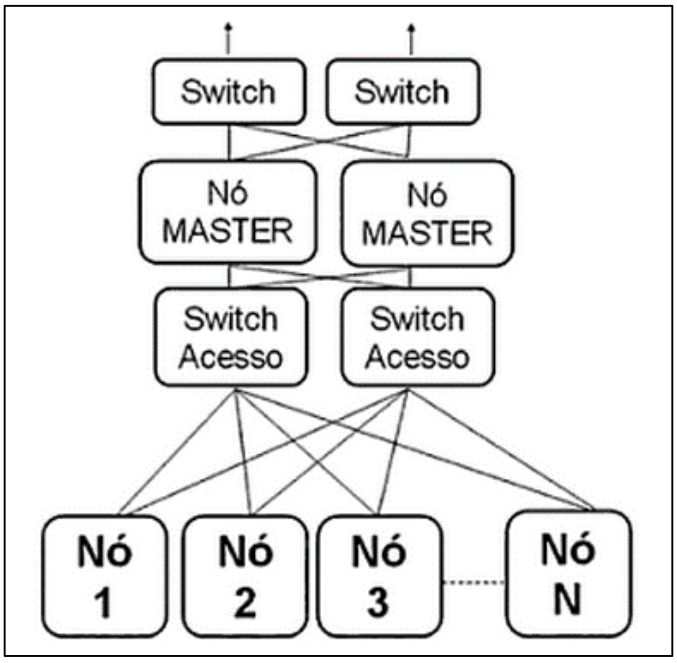
De acordo com Keswani (2008), os *clusters* de alta performance são implementados para providenciar aumento na performance. Isto é realizado através da divisão dos processos a serem executadas em muitos sub processos, que são executados em diferentes nós do *cluster.*

Segundo Gibson (2010), um *cluster* de balanceamento de carga contempla dois ou mais servidores que são configurados juntos para que a carga de processamento seja dividida entre eles. Quando existe uma nova conexão que será redirecionada para um servidor no *cluster*, o mecanismo assegurará que a carga

esteja balanceada. O principal objetivo do balanceamento de carga é aumentar a escalabilidade de um serviço ou aplicação.

De acordo com Gibson (2010), um *cluster* tolerante a falhas são dois ou mais servidores configurados juntos, dentre estes alguns são designados como nós ativos e outros como nós inativos. Os nós ativos providenciam os serviços para os clientes e os nós inativos monitoram os nós ativos. Se algum nó ativo falhar, o nó inativo tomará o lugar com uma pequena interrupção para os clientes, o principal objetivo deste *cluster* é a alta disponibilidade.

A Figura 8 mostra a arquitetura de um *cluster*, onde são exibidos os nós que fazem parte do mesmo, conectados aos *master nodes* por meio de *switches,* correspondendo a uma rede interna.



Fonte: Veras e Tozer, 2012, p.129.
Figura 8: Arquitetura de um *cluster*.

2.5.1.2 Grid

Segundo Veras e Tozer (2012), um *grid* pode ser um conjunto de *clusters*, estes podem estar distribuídos em diversos lugares e possuem uma grande

escalabilidade, o que proporciona uma grande ajuda em aplicações que requerem alto desempenho.

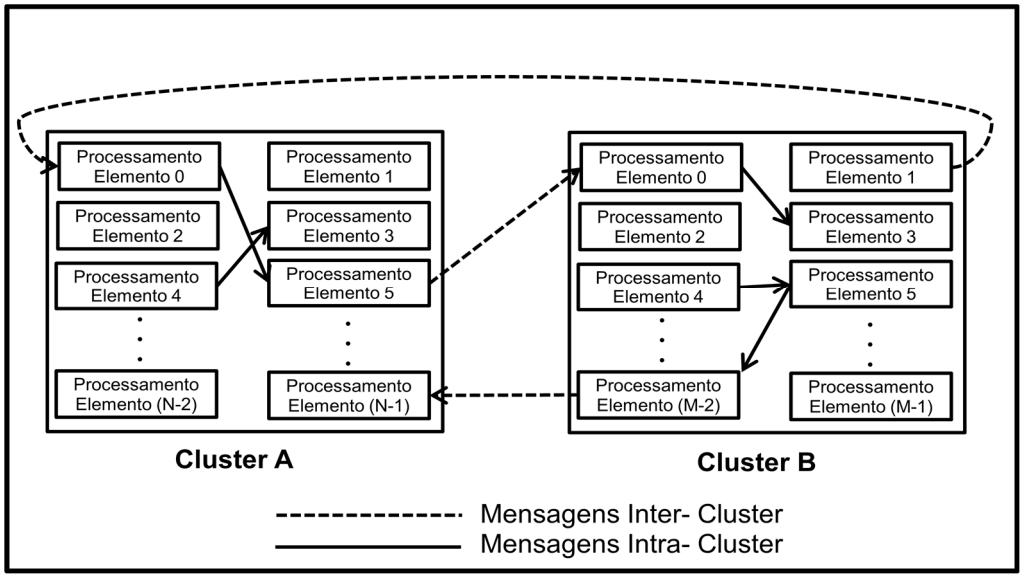
Conforme Joseph e Fellenstein (2004) computação em *grid*, fornece alta escalabilidade, alta segurança e extremo alto desempenho, negociando acesso remoto aos recursos computacionais de maneira continua. Isto se torna possível pelo compartilhamento de recursos computacionais, em um escala sem precedentes, entre um grande número de grupos distribuídos geograficamente.

De acordo com Minoli (2004) o *grid computing* ou *grid* computacional é o processamento de alto desempenho. Nele são alocados recursos computacionais. Neste caso, quem fornece os recursos são os nós que compõem o *grid*, que são servidores de alta performance.

Conforme Minoli (2004), o *grid* de dados é utilizado para armazenar e prover acesso aos dados de múltiplas organizações. O usuário não foca onde o dado se encontra, mas sim se o dado está disponível.

Segundo Fox (2006), o *grid* colaborativo é um modelo que fornece aplicações ou serviços compartilhados que podem ser utilizados simultaneamente por diversas pessoas, onde através da *Internet*, pode ser desenvolvida uma aplicação por pessoas geograficamente distantes na mesma plataforma.

A Figura 9 mostra a arquitetura de um *grid*, representando dois *clusters* que se comunicam entre si (constituindo o *grid)*, onde juntamente processam informações em comum.



Adaptado de: Koenig, 2007, p. 2.
Figura 9: Arquitetura de um *grid*.

Segundo Keswani (2008), computação em *grid* pode ser definida como um grupo de *clusters* conectados através da *Internet*. O *grid* é a próxima geração da computação distribuída. É um conjunto de recursos de *hardware* e *software* que juntos disponibilizam grande poder computacional e capacidade de armazenamento.

2.6 SEGURANÇA EM COMPUTAÇÃO EM NUVEM PRIVADA

Conforme Smoot e Tan (2012) as organizações tem mais controle sobre a segurança da infraestrutura das nuvens privadas em comparação a nuvens comunitárias e públicas. As nuvens privadas tem menor exposição a ameaças, além de atender as exigências regulamentares.

Segundo Smoot e Tan (2012), na segurança em nuvem podem ser usadas técnicas já existentes para segurança da infraestrutura que podem ser reutilizadas com algumas adaptações e adições. Existem *firewalls*, sistemas de prevenção contra intrusos na rede, *firewalls* de aplicações *web* podem ser aplicados para garantir a segurança em uma nuvem privada.

De acordo com Winkler (2011), há uma grande gama de componentes que implementam autenticação e controle de acesso em uma nuvem. Autenticação de usuário pode ser feita de diversas maneiras, mas todas são baseadas em uma

combinação de fatores: alguma coisa que o usuário saiba, como uma senha, algo que ele possui, como um *token*, ou algo que pode ser comparado como uma impressão digital.

Uma das grandes razões para empresas adotarem a nuvem privada é a proteção dos dados. Segundo Winkler (2011), proteger os dados em uma nuvem privada, pode ser similar a proteger dados em um *datacenter* tradicionalmente utilizado. Autenticação, identificação, controle de acesso, criptografia dos dados, deleção segura e checagem de integridade dos dados são métodos que podem ser utilizados para garantir a segurança dos dados em uma nuvem privada.

2.7 FERRAMENTAS PARA COMPUTAÇÃO EM NUVEM

As ferramentas *Open Source* (Código Aberto) na computação em nuvem permitem a criação e utilização de nuvens públicas, privadas e demais tipos de nuvem. Nesta seção serão apresentadas definições, arquitetura e principais características de algumas das ferramentas existentes e mais utilizadas para computação em nuvem.

2.7.1 Eucalyptus

Segundo Sempolinski e Thain (2010), Eucalyptus foi feito para ser *open source* em resposta a nuvem comercial EC2. Possui uma interface para o usuário que inclui um programa chamado euca2ools, que é semelhante ao programa de *frontend* oferecido pela Amazon no EC2.

Conforme Sempolinski e Thain (2010), o Eucalyptus é muito indicado para computação em nuvem em ambientes de computação empresarial corporativa. Tendo bem definido a questão de acesso, onde somente o administrador pode ter acesso ao usuário root, enquanto outros usuários têm somente acesso a uma interface *web* para que possam utilizar o sistema da empresa.

De acordo com Sempolinski e Thain (2010), o administrador tem a capacidade de disponibilizar configurações pré-definidas, que incluem:

processadores, memória e espaço em disco. Tudo isso para facilitar a experiência do usuário, que deve se limitar a escolher uma configuração para a criação da máquina virtual.

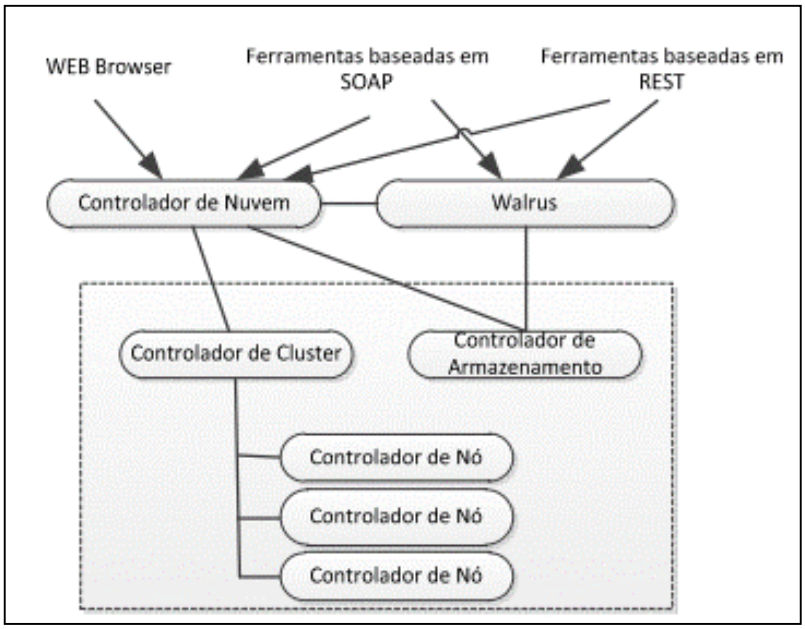
2.7.1.1 Arquitetura

Conforme Machado (2011), a estrutura do Eucalyptus possui cinco componentes básicos para fazer o gerenciamento de uma nuvem, cada um dos componentes exerce uma função própria. O Controlador de nuvem (CLC – *Cloud Controller*) é o componente de mais alto nível, ele disponibiliza a interface *web* para realizar o controle, além disso, gerencia requisições, faz escalonamento de recursos e gerencia contas.

O componente Walrus (WS3) é o responsável por implementar o sistema de armazenamento que é disponível interna e externamente por meio de uma interface que é compatível com o S3 da Amazon. Este componente armazena as imagens das máquinas virtuais e os dados dos usuários. Considerado como um sistema de armazenamento de arquivos muito simples.

Outro componente é o Controlador de *Cluster* (CC – *Cluster Controller*) que é implantado como serviço no Apache, cada *cluster* deve possuir um controlador. Esse controlador é o responsável pelo escalonamento a nível de *cluster*, controle de rede e controlador de armazenamento.

E o Controlador de Armazenamento (SC – *Storage Controller*), é o componente responsável por criar os *snapshots* e o volume das máquinas virtuais, este componente também fornece o armazenamento em bloco. E por fim o componente Controlador de nó (NC – *Node Controller*) que é o responsável pelo controle do *hypervisor*. A instalação deve ser feita em cada nó existente no *hypervisor*. A Figura 10 demonstra os cinco componentes dentro da estrutura do Eucalyptus.



Fonte: Machado, 2011, p.6.
Figura 10: Componentes e estrutura do Eucalyptus.

2.7.1.2 Características

Segundo Machado (2011), o Eucalyptus tem como características a possibilidade de separar em servidores diferentes, os componentes de gerenciamento fazendo com que a administração da nuvem seja mais simples para o controlador da nuvem. Porém a implantação dessa função pode tornar o custo da solução maior, principalmente se for para a finalidade de construir um ambiente redundante dos componentes de gerenciamento.

2.7.2 OpenNebula

O OpenNebula é uma ferramenta de código aberto, pode ser baixado de um repositório e de diversas distribuições comerciais, o que o tornou uma tecnologia avançada e muito utilizada para o gerenciamento de nuvens.

Conforme OpenNebula [a] (2010), é o resultado de muitos anos de pesquisa e desenvolvimento de uma gestão mais eficiente e escalável de máquinas virtuais em infraestrutura distribuídas. Muitas de suas características foram desenvolvidas para atender aos requisitos de empresas que utilizavam a ferramenta em computação em nuvem.

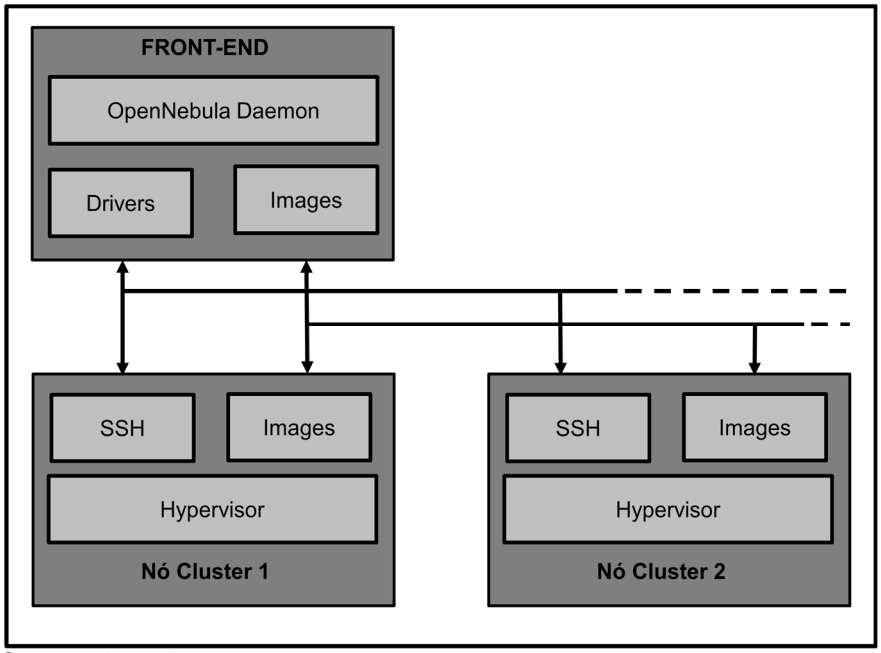
Segundo OpenNebula (2011) apud Soto (2011), o OpenNebula pode ser utilizado como ferramenta de virtualização pois auxilia no gerenciamento da infraestrutura virtual de *datacenters* e *clusters*, definindo uma nuvem privada. Além de nuvens privadas o OpenNebula também tem suporte a nuvens híbridas o que permite disponibilizar ambientes para hospedagem escalável e nuvens públicas.

Conforme OpenNebula (2011) apud Soto (2011), OpenNebula é um conjunto de armazenamento, rede, virtualização, monitoramento e tecnologias de segurança, que permite posicionamento dinâmico em infraestruturas distribuídas de máquinas virtuais interligadas, fazendo uma combinação de recursos de *datacenter* e nuvem remota com a utilização de políticas de alocação.

2.7.2.1 Arquitetura

Segundo Soto (2011), a arquitetura do OpenNebula tem a presença de um nó *frontend* que é responsável por executar serviços de administração da infraestrutura. Os outros nós são responsáveis por executar máquinas virtuais que utilizam de recursos disponíveis. Para isso, existe pelo menos uma rede de comunicação física que faz a interligação dos nós. O nó *frontend* executa o OpenNebula *Daemon* e representa o núcleo do sistema e administra o ciclo de vida das maquinas virtuais. Também são executados *drivers* da ferramenta e esse nó ainda atua como repositório de imagens de máquinas virtuais.

Os outros nós tem *hypervisors* habilitados para a execução de máquinas virtuais. A comunicação do *frontend* e os nós de *hypervisors* acontece por meio de canais SSH. A Figura 11 mostra como é essa arquitetura.



Adaptado de: Soto, 2011, p.26.
Figura 11: Arquitetura OpenNebula.

2.7.2.2 Características

Segundo site do OpenNebula [b] (2013), OpenNebula possui diversas características e funcionalidades que auxiliam na gestão integrada dos centros de dados virtualizados, permitindo a criação de nuvens privadas, públicas e híbridas.

Entre as características e funcionalidades encontram-se:

- a gestão de segurança do usuário: que fornece segurança e eficiência aos usuários e grupos do subsistema para autenticação e autorização de pedidos que possuem funcionalidades para o gerenciamento de usuários;

- a multi-locação com grupos de gestão: onde cada grupo tem configurado o acesso a recursos compartilhados, permitindo assim um ambiente multiusuário com vários grupos que compartilham a mesma infraestrutura;

- a prestação de *datacenters* virtuais sob demanda: onde o *Virtual Data Center* (VDC) é um ambiente de infraestrutura virtual totalmente isolado, onde um

grupo de usuários sob o controle do administrador VDC, pode criar e gerenciar computação, armazenamento e capacidade de rede.

Ainda pode ser citadas como características e funcionalidades o controle avançado e monitoramento da infraestrutura virtual, controle avançado e monitoramento de infraestrutura física, otimização de recursos distribuídos, gerenciamento centralizado de várias regiões, alta disponibilidade, compartilhamento de dispositivos virtuais entre várias instâncias. Interfaces de nuvem padrão e portal de auto atendimento simples para os usuários, além de múltiplas opções de implantação, fácil extensão e integração, confiabilidade, eficiência e escalabilidade.

Segundo Toraldo (2012), o OpenNebula não adota um *hypervisor* definido, também não possui requisitos específicos de infraestrutura, adequando-se a qualquer ambiente existente de armazenamento, rede ou políticas de gerenciamento de usuário.

2.7.3 OpenStack

Segundo OpenStack apud Laszewski et al. (2012), OpenStack é um conjunto *open source* de elementos que disponibilizam nuvens públicas e privadas. Entre esses componentes estão o OpenStack *Compute*, conhecido por Nova, OpenStack *Object Storage* denominado *Swift* e OpenStack *Image Service* chamado *Glance.*

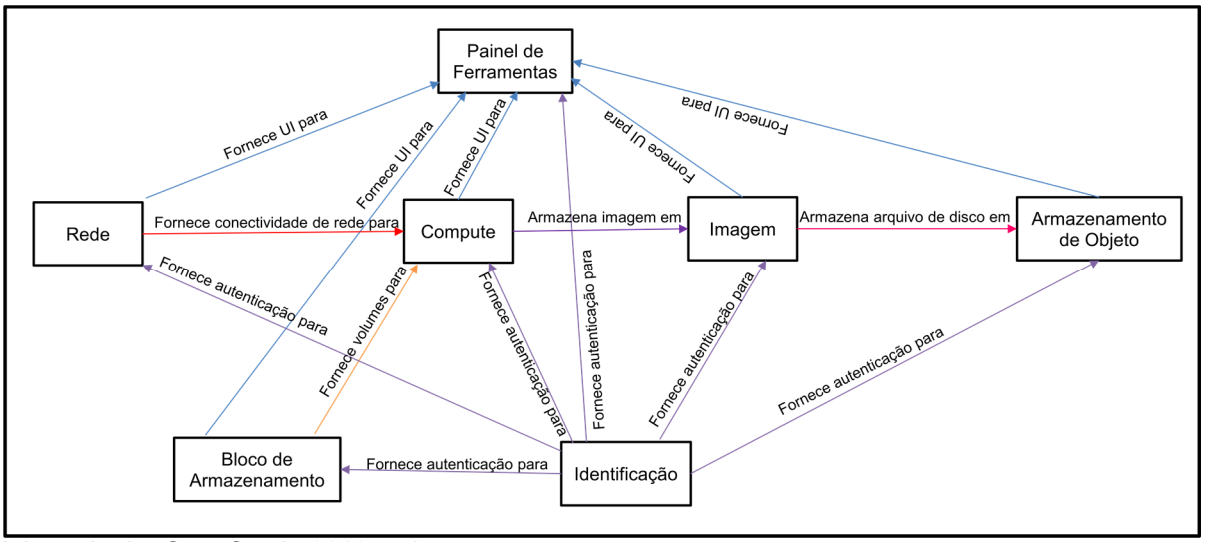
De acordo com Laszewski et al. (2012), o componente Nova, foi criado para fornecer e gerenciar grandes redes de máquinas virtuais, preservando uma plataforma de computação em nuvem redundante e escalável. O *Swift* é um sistema de armazenamento para grandes quantidades de dados permanentes ou estáticos por longo período de tempo. O *Glance* é um componente que permite criar, registrar e entregar serviços para imagens de disco virtuais.

Conforme OpenStack [a] (2013), através de uma interface, o administrador pode gerenciar a capacidade de computação, armazenamento e recursos de rede,

presentes no *datacenter* administrado. Oferece também uma interface *web* aos usuários, que possibilita alocar recursos conforme necessário.

2.7.3.1 Arquitetura

Segundo OpenStack [b] (2013), a arquitetura do OpenStack mostrada na Figura 12, é uma ilustração simplificada da arquitetura, presumindo-se que seja utilizado todos os serviços que são oferecidos. A Figura 12, apresenta os componentes que fazem parte de uma nuvem onde é implementado o OpenStack, nele existem o Painel de Ferramentas (Horizon), que oferece um *frontend web* para outros serviços oferecidos. O *Compute* (Nova) armazena e recupera discos virtuais. A Rede (*Quantum*) fornece a rede virtual. O Bloco de Armazenamento (*Cinder*) fornece armazenamento e o Imagem (*Glance*) armazena os arquivos do disco virtual real no Armazenamento de Objeto (*Swift*). Por fim, a identificação (*Keystone*), fornece autenticação.



Adaptado de: OpenStack, 2013, p.1.

Figura 12: Arquitetura do OpenStack.

2.7.3.2 Características

O OpenStack [c] (2013) descreve algumas de suas características juntamente com seus benefícios. São citadas como características, o Gerenciamento de virtualizador de recursos do servidor de *commodities* (CPU,

memória, disco e interfaces de rede), que beneficia na melhor utilização e automação de recursos o que diminui os custos.

Gerenciar redes LAN (*Local Area Network*), permite alocar IPs e VLANs de forma programática para prover rapidamente recursos de rede e de segurança. A API (*Application Programming Interface*) com limitação de taxa e autenticação serve para gerenciar os usuários que tem acesso aos recursos de computação e impedir que uns afetem aos outros com o uso excessivo da API.

Outra característica é a arquitetura distribuída e assíncrona, tornando o sistema altamente escalável e disponível. O gerenciamento de máquinas virtuais facilita o armazenamento, importação, divisão e consulta de imagens VM (por exemplo, executar, reiniciar, suspender, redimensionar e encerrar casos).

O recurso de endereço de IP *floating*, é a capacidade de atribuir endereços IP para as VMs. Os grupos de serviço fornecem flexibilidade para atribuir e controlar o acesso a instâncias de VM. *Role Based Access Control* (RBAC) garante a segurança por usuário, papel e projeto.

Por fim, a característica de projetos e cotas, beneficia a capacidade de alocar, monitorar e limitar a utilização de recursos. Isso inclui: Proxy VNC através do navegador *web*; armazenamento e gerenciamento de arquivos de programação via API; acesso root separado para a gestão e serviços; painel com suporte totalmente integrado para provisionamento *self-service*; *VM Image Caching* em nós de computação, ou seja, provisionamento rápido de VMs.

2.7.4 CloudStack

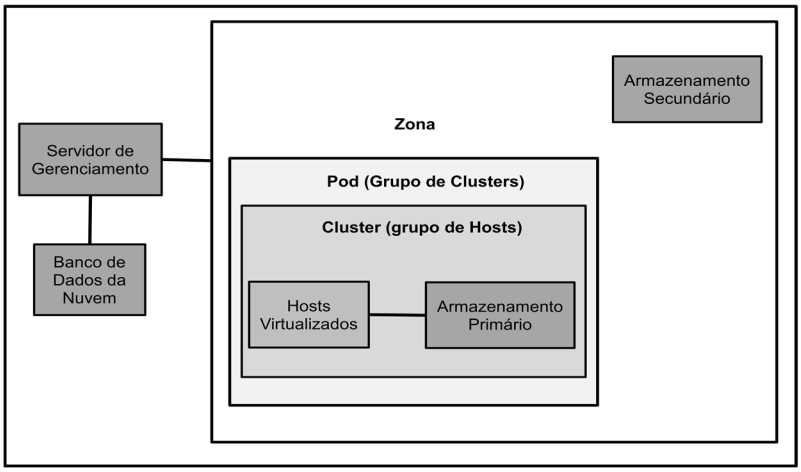
Segundo CloudStack (2013), o *software open source* CloudStack foi desenvolvido para implantar e gerenciar grandes redes de máquinas virtuais, pois possui alta disponibilidade de infraestrutura e também é escalável. É utilizado por prestadoras de serviços por fornecer serviços de nuvem pública, privada ou híbrida.

De acordo com CloudStack (2013), o CloudStack fornece uma gama de recursos necessários para a maioria das organizações que utilizam uma nuvem que oferece a infraestrutura como serviço (Iaas). Suporta os virtualizadores mais populares: VMware, Xen e KVM.

Conforme Sabharwal e Shankar (2013), CloudStack oferece uma solução para criação de nuvens privadas, híbridas e públicas que podem fornecer infraestrutura de TI, *hosts*, rede e armazenamento como um serviço para os usuários, fornecendo a demanda necessária.

2.7.4.1 Arquitetura

A Figura 13 mostra a estrutura de *cluster* onde está instalado o ClouStack, onde são mostrados os *hosts* virtualizados que fornecem os serviços. O Armazenamento Primário, que armazena os discos virtuais. O *cluster* que é o grupo de *hosts*. O *pod* que é um grupo de *clusters*. O Armazenamento Secundário, onde são salvas *snapshots* e ISO do *storage*, bem como a Zona onde todos os outros componentes se encontram.



Adaptado de: Sabharwal, Shankar, 2013, p.20.
Figura 13: Arquitetura do CloudStack.

2.7.4.2 Características

Conforme CloudStack (2013), a ferramenta possui características como: gerência de recursos de um servidor virtualizado; gerenciar LAN; arquitetura

distribuída e assíncrona; gerenciamento de imagens virtuais; capacidade de atribuir IP's para todas as máquinas virtuais; criação de grupos de segurança com acessos limitados a máquinas virtuais e recursos; e alocação de recursos.

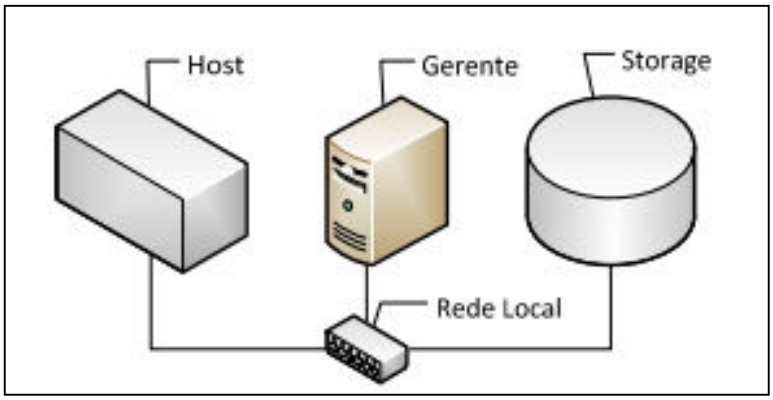
2.7.5 OpenQRM

Conforme Ransome e Rittinghouse (2009), OpenQRM é uma ferramenta que gerencia virtualização, armazenamento e a rede. É possível fazer o gerenciamento de toda a infraestrutura de TI a partir de um console, sendo este uma interface *web* ou por linha de comando. Permite criação de nuvens privadas com alta disponibilidade. É uma ferramenta com suporte a diferentes tecnologias de virtualização como: KVM, VMware, Xen e XenServer.

De acordo com Rechenburg (2010), a primeira versão do OpenQRM foi desenvolvida pela companhia Qlusters. A primeira versão do OpenQRM era baseada na linguagem Java. Em 2005, OpenQRM já incluía um sistema de provisionamento de recursos totalmente automatizado. Sendo a partir de 2006 *open source.* E a partir de 2008, foi totalmente reescrito, sendo mantidas as características e mecanismos.

2.7.5.1 Arquitetura

Segundo Machado (2011), o OpenQRM funciona de maneira gerente-agente, para isso o controlador de nuvem é o gerente e os recursos que são integrados a ele são os agentes. Essa ferramenta opera com o gerente centralizado, o que não torna possível separá-lo em componentes individuais. A estrutura do OpenQRM é formada da seguinte forma: gerente, *storage* e nó que são conectados por uma rede local, como mostra a Figura 14.



Fonte: Machado, 2011, p.6.
Figura 14: Estrutura do OpenQRM.

2.7.5.2 Características

Segundo o *site* OpenQRM (2013) é uma ferramenta que possui diversas características como suporte para diferentes tecnologias de virtualização. Com isso suporta e gerencia VMware, Xen, KVM e máquinas virtuais Citrix XenServer. O OpenQRM proporciona uma configuração do Nagios (ferramenta de monitoramento de ativos de redes, mais informações sobre ela na referência (NAGIOS, 2013)) de forma totalmente automática para monitorar todos os sistemas e serviços. Isso facilita bastante, já que o Nagios é uma ferramenta de monitoramento de serviços difícil e demorado de instalar.

OpenQRM fornece o *ready-made* para começar de forma rápida e fácil a troca de imagens dos servidores, podendo ser utilizado para Debian, Ubuntu e CentOS. Para isso, foi adicionado um *plugin image-shelf* que permite ao administrador do sistema a busca de *server-images* de maneira fácil através da interface *web*.

O gerenciamento de armazenamento integrado tem com vantagem ter apenas um lugar para *backup*/*restore* localizado no próprio armazenamento do servidor, permitindo aos usuários usar seus *cloning/snapshot* de recursos para outra vez criar *hot-backups* de seus servidores sem interrupções nos serviços.

Conforme o *site* OpenQRM (2013), esta ferramenta ainda possui alta disponibilidade, suporte para todos os diferentes tipos de implantação, apoio a

distribuição pois possui suporte para diferentes distribuições Linux. Tem também características como ser pequeno, fácil de instalar, *developer-friendly*, rápido para construir, tem seu sistema integrado de pacotes para criar rpms e/ou pacotes deb e possui suporte para vários tipos de bancos de dados.

2.7.6 *Ubuntu Enterprise Edition*

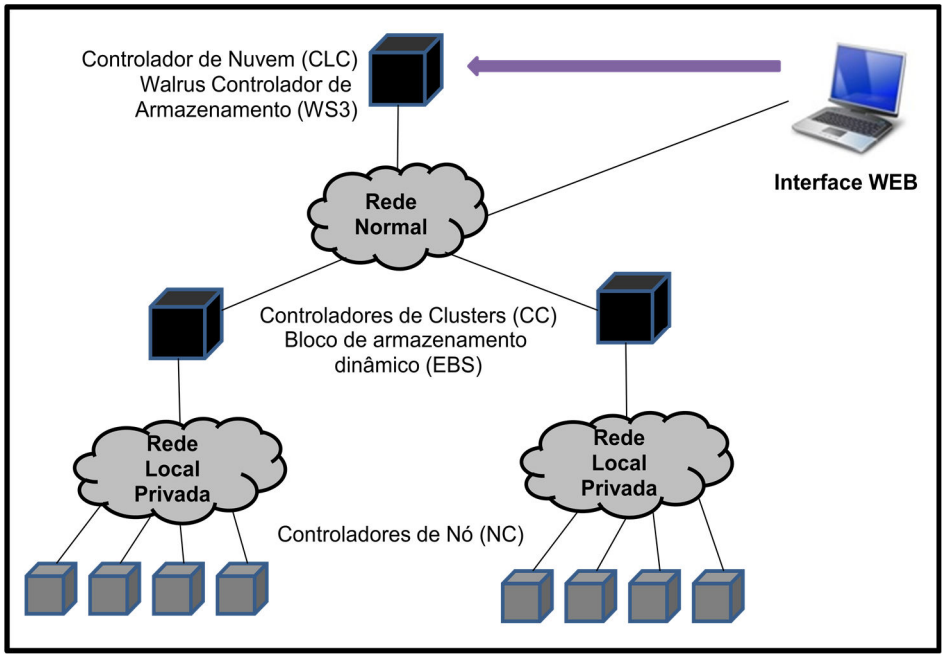
De acordo com Wardley, Goyer e Barcet (2009), é uma ferramenta que possibilita a criação de nuvens privadas e públicas para o modelo de serviço IaaS. Foi desenvolvido com o objetivo de agregar as melhores funcionalidades e componentes e aprimora-los. Além disso, possui suporte a virtualizadores como Xen e KVM.

Segundo Campesato e Nilson (2010), o Ubuntu *Enterprise Cloud* (UEC) é baseado na ferramenta Eucalyptus. É uma opção para soluções baseadas em IaaS. O UEC pode criar uma nuvem privada dentro da organização. O UEC também influência muitas ferramentas chamadas de Euca2ools que são baseadas em bibliotecas de código aberto de *Python.*

De acordo com Wardley, Goyer e Barcet (2009), o UEC foi concebido para simplificar a construção e gestão de uma nuvem interna para empresas de qualquer tamanho, permitindo as empresas criar sua própria nuvem com sua própria infraestrutura para auto atendimento.

2.7.6.1 Arquitetura

Como mostra a Figura 15, a arquitetura do Ubuntu *Enterprise Cloud*, por ser baseado no Eucalyptus, acaba sendo muito semelhante a ele. É composta pelo: Controlador de Nuvem que providencia a interface de interação dos usuários, o Walrus Controlador de Armazenamento, que controla o armazenamento, o Bloco de Armazenamento Dinâmico, que cria *snapshots* das VM que estão armazenadas no Walrus, o Controlador de *Cluster* que controla os Controladores de Nó que executam nos nós do *cluster*.



Adaptado de: Wardley, Goyer e Barcet, 2009, p.8.
Figura 15: Estrutura do Ubuntu *Enterprise Cloud*.

2.7.6.2 Características

Segundo Canonical (2010) o Ubuntu Enterprise Cloud possibilita criar um perfil de instalação mínima de máquina virtuais, suporte a RAID, iSCSI, gerenciamento da máquina física e das máquinas virtuais, monitoramento da topologia da nuvem, gerenciamento dos IP's, credenciais de segurança, controle de mudança de versão.

2.7.7 Abiquo

É uma ferramenta utilizada para criação de nuvens privadas para o modelo de serviço IaaS. Tem como objetivo facilitar aos usuários a administração de uma nuvem privada através da limitação de escolha dos recursos a serem utilizados sem ter contato com os meios físicos. Fornece políticas de negócio que providenciam controle total sobre cada unidade operacional que possa executar uma tarefa ou armazenar dados. Fornece *logs* que permitem analisar quem e para que estão sendo utilizados os recursos.

Segundo Abiquo [a] (2013), Abiquo proporciona uma plataforma para seus serviços de TI, possibilitando criar nuvens privadas baseadas em uma infraestrutura já existente ou controlar o uso de serviços em nuvem pública. Abiquo utiliza recursos como computadores, rede e armazenamento já existentes e permite abstrair o contato do consumidor com a infraestrutura física, expondo aos usuários através de *datacenters* virtuais. Os recursos virtuais são controlados pelas políticas da corporação.

2.7.7.1 Arquitetura

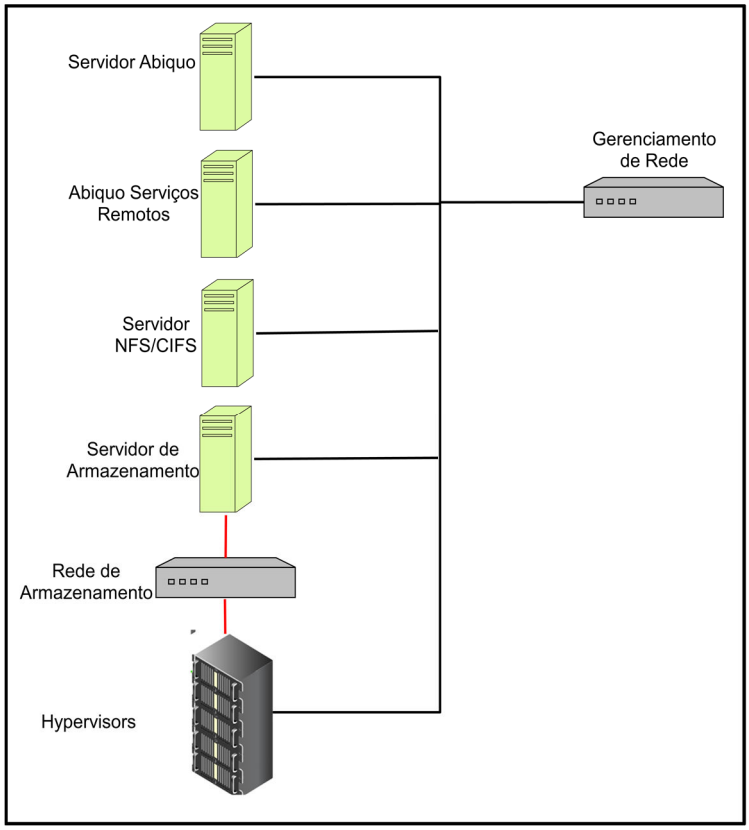
Segundo Smyth (2013), no ambiente de nuvem Abiquo, as nuvens clientes são separadas de recursos físicos pela nuvem de recursos, um conjunto de recursos disponíveis é controlado pelo administrador da nuvem. O Abiquo permite que o administrador da nuvem possa gerenciar a complexidade dos recursos físicos na nuvem de recursos. O administrador da nuvem pode ainda proporcionar facilidades para os usuários, configurando a sua alocação de recursos a partir da nuvem e permitindo-lhes *self-service*. Contudo o administrador da nuvem continua controlando a infraestrutura física para fazer o gerenciamento da complexidade. A infraestrutura física da nuvem de recursos está organizada em Abiquo *Datacenters*.

Segundo Smyth (2013), a arquitetura básica do abiquo é constituída pelo Gerenciador de Rede que conecta o Servidor Abiquo ao *Cluster* da nuvem (*Hypervisor*). A rede do *Cluster* liga as máquinas virtuais na nuvem. A Rede de Armazenamento é uma rede para o tráfego de dados entre o *cluster* e a armazenamento iSCSI (*Internet Small Computer System Interface*).

Possui ainda o Abiquo de Serviços Remotos, que é o componente responsável por gerenciar o monitoramento, virtualização, armazenamento, aplicações e também o serviço DHCP. O Servidor NFS/CIFS é um diretório compartilhado necessário para o Abiquo, utilizado para armazenar modelos de máquinas virtuais.

E o Servidor de Armazenamento fornece armazenamento virtual de auto-atendimento para os usuários, podendo ser utilizado para alocar recursos de

armazenamento externo para dispositivos virtuais, porém esse tipo de armazenamento é opcional. A Figura 16 demonstra a arquitetura do Abiquo.



Adaptado de: Smyth, 2012, p.1.
Figura 16: Arquitetura Abiquo

2.7.7.2 Características

Segundo Abiquo [b] (2013), suas características principais são políticas de negócio que permite que a empresa defina, edite e gerencie políticas de negócio a partir de um editor único e centralizado. Virtual *Enterprises* autônomas, que dá capacidade aos usuários com *multi-tenant* Virtual *Enterprises* permitindo controle total sobre os recursos do servidor, armazenamento e rede.

Possui também mecanismo de preços e faturamento, que atribui um preço a qualquer recurso faturável seja CPU, RAM, armazenamento ou rede. Isso permite uma cobrança retroativa e faturamento de recursos com suporte multimoeda. Apresenta relatório completo, permitindo através de integração com JasperSoft, gerar relatórios completos, formatados, personalizados e exportáveis.

2.7.8 Convirt

O ConVirt é uma ferramenta que possibilita facilmente a transição para uma nuvem privada, podendo utilizar recursos de um *datacenter,* sendo que os recursos podem ser realocados se necessários. Também permite a criação de diversos *datacenters* virtuais, compartilhando os mesmos recursos de maneira transparente ao usuário.

Segundo Convirture [a] (2013), o ConVirt *Enterprise Cloud* possibilita centralizar o gerenciamento de *datacenter* virtuais, dando completa visibilidade e gerenciamento através de todo o ambiente. Como todo negócio demanda mudanças, é possível rapidamente propor a infraestrutura entre a nuvem e a tradicional carga de trabalho virtualizada.

De acordo Buyya, Broberg, Goscinski (2010), o Convirt é capaz de gerenciar todo o ciclo de uma máquina virtual. É possível monitorar os recursos do servidor e monitorar os clientes da máquina virtual. Isto ajuda a controlar a carga exercida sobre o servidor.

2.7.8.1 Arquitetura

Segundo Convirture [c] (2013), as três edições do Convirt no que se trata de arquitetura apresentam: Suporte multi-plataforma (permitindo rodar em diferentes sistemas), arquitetura *Agent-less* que possui informação detalhadas sobre as configurações dos itens em rede, *Universal web Access (*permitindo *o* acesso *à Web* de forma adaptável*)*, *Datacenter-wide Console* (tudo acessível através de um console de gerenciamento) e escalabilidade aprimorada. Em exceção, a versão do ConVirt *open source* não possui escalabilidade aprimorada.

2.7.8.2 Características

Segundo Convirture [b] (2013), as principais características dessa ferramenta são a possibilidade de gerenciar o ciclo de vida das máquinas virtuais, permitindo administrar, configurar e monitorar todas as máquinas virtuais a partir de

um único console. Gerenciar facilmente *pools* de servidor para aproveitar a agilidade proporcionada pela virtualização. Provisão rápida com *templates*, pois adota uma abordagem baseada em modelos de provisionamento de servidores, o que permite padronizar seus ambientes virtualizados.

Otimiza a implantação com a colocação de máquina virtual inteligente. Faz *download* e instala dispositivos virtuais, pois possui um navegador de catálogo integrado que detecta aparelhos prontos para implantar virtuais. Permite migrar máquinas virtuais arrastando e soltando, mover máquinas virtuais entre *hosts* apenas utilizando a operação conhecida como *drag-and-drop,* enquanto elas estão em execução, sem interromper as suas operações.

Monitora a disponibilidade e o desempenho. Configura e monitora o armazenamento de *pools* de servidores e implanta o ConVirt rapidamente com sua arquitetura sem agentes, não exigindo a implantação de um agente de *software* nos servidores que estão sendo gerenciados.

2.7.9 Apache virtual Computing Lab (VCL)

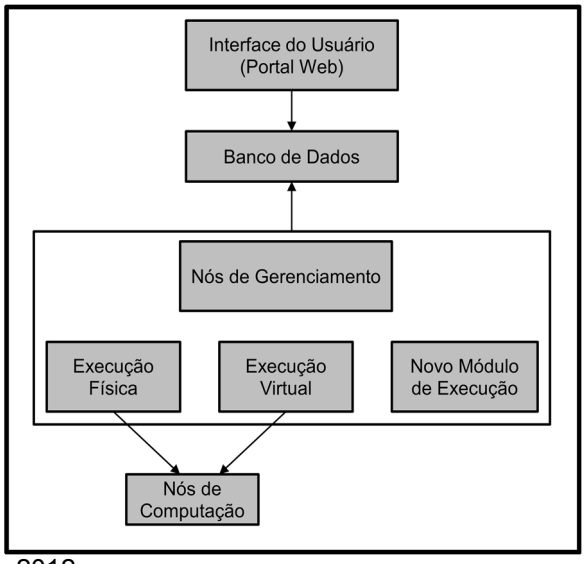
Segundo Apache VCL [a] (2012), VCL (*Virtual Computing Lab*), é uma plataforma *open source* para computação em nuvem que oferece ambientes personalizados aos seus usuários. Esses ambientes podem ser desde uma máquina virtual ou até mesmo um *cluster* de servidores físicos. O VCL tem suporte a tipos diferentes de recursos de computação, além de possuir portal *web self-service* para seus usuários.

Segundo Apache VCL (2010) apud Endo et al. (2010), o VCL é uma ferramenta *open source* utilizada para acesso remoto a partir da *Internet* de maneira dinâmica e utilizando-se de reservas de recursos computacionais, funciona como um *software* para soluções (SaaS). *Virtual Computing Lab* tem sua arquitetura formada por três camadas: Servidor *Web*, Servidor de banco de dados e Nós de gestão.

“O VCL oferece os modelos de serviço IaaS e PaaS, que poderiam ser usados para hospedar sistemas de colaboração (SaaS) em cima deles” (DOELITZSCHER et al., 2010, p.3).

2.7.9.1 Arquitetura

Segundo Apache VCL [b] (2012), a arquitetura do VCL é formada por quatro componentes, são eles: Portal *Web*, Banco de Dados, Nó de Gestão que permite a Execução Física, Virtual ou a criação de um novo módulo de execução e Nós de Computação podendo estes ser formados por servidores físicos, máquinas virtuais, máquinas em laboratório de informática ou recursos de computação em nuvem. A Figura 17 demonstra esses componentes e como eles se comunicam.



Adaptado de: Apache VCL, 2012.
Figura 17: Arquitetura VCL (*Virtual Computing Lab*).

2.7.9.2 Características

Segundo Apache VCL [c] (2013), suas principais características são ser *Free* e *OpenSource,* pois não possui requisitos de produtos comerciais para implantar uma nuvem VCL totalmente funcional. Além disso, possui o portal *web self-service*, que apresenta uma interface simples e agendamento de tarefas, suporte para vários métodos de autenticação, criação de imagem de forma simples tendo controle de

revisão de imagem e delegação de criação de imagem. Também possui suporte a vários métodos de provisionamento.

Possui também como característica importante a alocação de blocos, útil para a sala de aula ou oficinas, pois permite dar prioridade de carregamento e acesso dedicado. Outra característica são os Ambientes de *cluster*, *self-service* que permite construir e gerenciar um conjunto personalizado. E por fim, cita-se a característica de Estatísticas e dados de registro, pois é construído em páginas de estatísticas, fornece dados em formato legível e gráficos sendo utilizados para determinar de como e quando os ambientes são usados.

2.7.10 Nimbus

Segundo Pillai e Swasthimathi (2012), o Nimbus é considerado um conjunto de ferramentas *open source* que fornecem recursos IaaS. Possui três objetivos: permitir aos provedores de recursos a construção de nuvens privadas IaaS, permitir aos usuários o uso de nuvens IaaS e permitir aos desenvolvedores melhorar, testar e personalizar IaaS.

Conforme Pillai e Swasthimathi (2012), com o objetivo de permitir aos provedores de recursos a construção de nuvens privadas IaaS, o Nimbus implanta *clusters* virtuais autoconfigurados. O Nimbus também fornece uma ferramenta complementar chamada *Cumulus*, está permite que a execução do armazenamento em nuvem seja baseada em quotas, além de permitir aos provedores de recursos construir armazenamento em nuvem múltipla, pois o *Cumulus* foi projetado para escalabilidade.

De acordo com Pillai e Swasthimathi (2012), o Nimbus fornece também a ferramenta Nimbus *Context Broker*, usada para cria uma configuração em comum com o contexto de segurança através de recursos provisionados para possíveis nuvens múltiplas. Além de fornecer ferramentas chamadas de *Sky Computing Tools*, que são ferramentas de escala que permitem aos usuários aumentar automaticamente mesmo estando em meio a diversos provedores distribuídos.

Essas ferramentas costumam atuar em ambientes multinuvem que reúnem recursos de nuvem privada e pública (mais conhecido como nuvens híbridas).

Segundo Pillai e Swasthimathi (2012), para proporcionar o objetivo de permitir aos desenvolvedores melhorar, testar e personalizar IaaS, o Nimbus proporciona alta qualidade através de um ambiente totalmente configurável e permite a implementação extensível de código aberto. Possui diversas opções como o *Workspace Service* que pode ser configurado para suportar diferentes implementações de virtualização, apresentando opções de gerenciamento de recursos e interfaces compatíveis. Tem seus componentes frequentemente testados, o que permite aos desenvolvedores atualiza-los mais facilmente.

2.7.10.1 Arquitetura

Segundo Keahey e Freeman (2008), a arquitetura do Nimbus consiste nos seguintes componentes:

- *Workspace* de serviços, permite ao cliente implantar e gerenciar grupos definidos de VMs de maneira flexível e remotamente.

- *Workspace* gerenciador de recursos, realiza a implantação de contratos de locação de VM.

- *Workspace pilot*, se estende a gestores de recursos locais existentes (LRMs) para implantar as máquinas virtuais, utilizando a virtualização RPsto sem alterar significativamente a configuração local.

- IaaS *gateway*, permite que um cliente apresentando uma credencial PKI, utilize outra infraestrutura IaaS que possua credenciais diferentes.

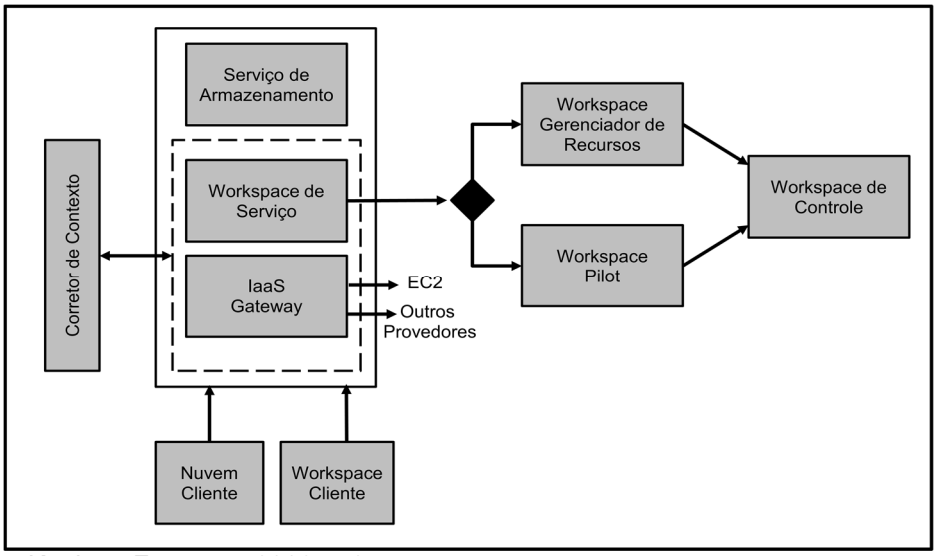
- Corretor de contexto, possibilita ao cliente implantar um "*one-click*" ao funcionamento de *cluster* virtual em oposição a um conjunto de máquinas virtuais “estranhas”, assim como permite personalizar VMs.

- *Workspace* cliente, fornece funcionalidade total do serviço de acesso para *workspace*, mas é mais complexo de utilizar.

- Nuvem cliente, permite acesso somente a uma *SelectSet* de funções, mas é fácil de utilizar e popular como uma ferramenta para usuário final.

- Serviço de armazenamento Nimbus, fornece gerenciamento seguro de espaço em disco da nuvem, dando a cada usuário um repositório que possibilita ver imagens de VMs deles próprios e as imagens que eles podem lançar.

A Figura 18 demonstra estes componentes e a relação entre eles.



Adaptado de: Keahey, Freeman, 2008, p.2.
Figura 18: Arquitetura Nimbus.

2.7.10.2 Características

Para Pillai e Swasthimathi (2012), as principais características do Nimbus são IaaS *Open Source*, armazenamento *Cloud Service*, implementação remota de gerenciamento do ciclo de vida de VMs, compatibilidade com protocolos de rede Amazons, nuvem cliente fácil de utilizar, desenvolvimento rápido, suporte a vários protocolos, grupo de gestão flexível, uso de rastreamento por cliente,

armazenamento de quota por usuário, solicitação flexível de autenticação e autorização, fácil gerenciamento de usuários e *workspace client*.

"A plataforma Nimbus é integrada com um conjunto de ferramentas que entregam força e versatilidade a infraestrutura de nuvem para os usuários. Também permite combinar Nimbus com Openstack, Amazon e outras nuvens" (NIMBUS PROJECT, 2013).

CAPÍTULO 3: RESULTADOS OBTIDOS

O capítulo 3 apresenta seção de trabalhos relacionados, comparação das ferramentas, planejamento de cenário de testes, instalação das ferramentas, testes realizados e análise dos resultados.

3.1 TRABALHOS RELACIONADOS

A pesquisa por trabalhos que possuem alguma relação ao presente trabalho foi de grande valia, pois estes auxiliaram a aprimorar os conhecimentos sobre Computação em Nuvem e como está o cenário de pesquisa nesta área. Além disso, foi uma forma de conhecer algumas das ferramentas utilizadas e verificar aonde cada uma delas é utilizada.

Toda área de estudo necessita a busca por trabalhos relacionados para que se possa ter uma noção do que já foi feito e o que ainda precisa ser explorado. Além disso, é possível observar as referências que são mais utilizadas, trazendo uma boa base para iniciar a pesquisa. Com isso, o trabalho ganha maior originalidade, consistência e possibilita que experimentos não sejam realizados repetitivamente sem grande evolução nos resultados.

3.1.1 Performance Evaluation of the Illinois Cloud Computing Testbed

Conforme Khurshid, Al-Nayeem e Gupta (2009), nesse projeto são feitos testes utilizando a ferramenta Open Cirrus, é feita uma avaliação do desempenho deste tipo de nuvem oferecendo o serviço pra clientes localizados em diferentes lugares, para analisar o desempenho enquanto são realizadas transferências dos

dados. Para fazer os testes foram utilizados PlanetLab e Emulab para que assim fosse possível simular a utilização por usuários distribuídos e utilizando aplicativos em nuvem. O PlanetLab é utilizado para poder emular uma diversidade de usuários. Já o Emulab é para poder emular uma instalação em nuvem.

Os pesquisadores perceberam que o desempenho da transferência de dados em uma nuvem pode variar dependendo de quantos usuários diferentes estão utilizando o mesmo serviço, e que muitas vezes, as variações são atribuídas as características de rede entre a nuvem e seus usuários. Sendo que com isso a distância entre a nuvem e os usuários é de grande determinação para a quantidade de desempenho. Esse projeto tem o objetivo de dar maiores contribuições para pesquisas da tecnologia em nuvem, pois os pesquisadores realizaram uma extensa avaliação de armazenamento e desempenho da ferramenta Open Cirrus oferecendo serviços para diversos usuários distribuídos.

3.1.2 A Survey on Open-source Cloud Computing Solutions

Conforme Endo et al. (2010), neste trabalho é feita uma pesquisa sobre ferramentas *open source* para computação em nuvem. Nele são apresentadas algumas ferramentas com suas características e arquiteturas que geralmente são utilizadas. Não é criado nenhum ambiente de teste com as ferramentas. As ferramentas são: Xen *Cloud Platform* (XCP), Nimbus, OpenNebula, Eucalyptus, TPlataform, Apache *Virtual Computing Lab* (VCL*)*, e Enomaly *Platform Computing Elastic*. A comparação destas ferramentas é feita pelos autores através de uma tabela onde é descrita a ferramenta, o modelo de serviço que utiliza suas principais características e exemplos de quem as utiliza.

Os autores alcançaram como conclusão desta pesquisa a necessidade de padronização das plataformas atuais, no que se trata de interface, negociação, acesso por meio de *Web Services*. Pois atualmente as nuvens utilizam muitas vezes de níveis de abstração diferentes. Mas já se está encaminhando para isso com a criação do *Open Cloud Manifesto* que foi aceito por muitas empresas.

3.1.3 Cloud Testing Tools

Segundo Bai et al. (2011) a computação em nuvem proporciona uma grande comodidade para muitas empresas devido ao seu modo de negócio. Mas todas as nuvens precisam ser testadas de maneira eficiente para saber se um determinado *software* funcionará da maneira correta ou atenderá a demanda necessária para que cumpram as SLA's.

O trabalho apresenta uma gama de ferramentas que podem ser utilizadas para a realização de testes dos mais variados tipos, para que possam englobar toda a estrutura de uma nuvem e encontrar eventuais problemas. São por exemplo abordadas ferramentas de *benchmark* (YCSB, Enhanced TPC-W, Terasort, Cloudstone, Malstone), como também ferramentas para testes comerciais (SOASTA, iTKO LISA, *Cloud Testing*). E como resultado é apresentado uma tabela que mostram diversas características que podem ser testadas, e demonstra o que cada ferramenta apresentada de fato realiza.

3.1.4 Comparação de Ferramentas de Software Livre para Administração de Nuvem Privada

Conforme Machado (2011), o trabalho tinha por objetivo comparar e elencar qual a melhor ferramenta a ser utilizada para gerenciar uma nuvem privada. Para estudo foram utilizadas as ferramentas OpenQRM e Eucalyptus. Para a realização dos testes foi utilizado um ambiente isolado de 6 computadores *desktop*, com o sistema operacional Ubuntu 10.04 de 32 bits, e uma rede com acesso à *Internet*.

Como testes, foram realizadas tarefas em cada um dos componentes de cada ferramenta individualmente, onde se executava alguma tarefa (envio de pacotes ICMP e transferência de arquivo) e era realizada alguma falha proposital para ver o resultado. E como resultado foi visto que dentre os três componentes de cada ferramenta, o OpenQRM acabou tendo melhores resultados em dois deles, sendo assim a melhor ferramenta.

3.1.5 Evaluating Open-Source Cloud Computing Solutions

Segundo Voras et al. (2011) existe uma grande quantidade de critérios que devem ser levados em consideração para avaliar a computação em nuvem. São apresentados os modelos de computação e exemplos que existem e que são utilizados diariamente por milhões de usuários.

É apresentada uma série de produtos de código aberto que são considerados ter um futuro viável para aplicações em ambientes corporativos, como por exemplo, OpenNebula, Eucalyptus, Ubuntu *Enterprise Cloud*, OpenQRM, Abiquo, Red Hat *Cloud Foundations*, *Edition One*, OpenStack, Nimbus, mOSAIC. Foi planejado um conjunto de 95 critérios, para avaliar se a ferramenta é interessante para a implantação na empresa. Os critérios são agrupados em seis categorias principais: armazenamento, virtualização, gestão, rede, segurança e apoio. Como conclusão deste trabalho é que os critérios podem ser utilizados para escolher a ferramenta mais apropriada, que se adeque as necessidades da organização.

3.1.6 Comparison of Multiple Cloud Frameworks

Conforme Laszewski et al.(2012), neste trabalho é realizada a comparação de *frameworks*, tendo como principal objetivo contrastar os diferentes *frameworks* IaaS, realizando análises do tipo qualitativa. Onde foi preciso descobrir se os usuários precisavam de mais *frameworks* de acesso e se sim, qual *framework* deveria ser utilizado. Para isso, os autores analisaram a comunidade de usuários da empresa FutureGrid, para assim fazerem seus registros. Consta no trabalho um gráfico onde estão descritas as ferramentas escolhidas como parte do processo de aplicação do projeto em FutureGrid. Onde é possível perceber que as ferramentas para *cloud* mais escolhidas são Nimbus e Eucalyptus. Foi feita uma comparação das características dos IaaS de forma qualitativa, tendo algumas das características definidas em uma tabela.

É realizado um resumo dos *frameworks* de nuvem IaaS, onde são discutidos os *frameworks* disponíveis que são diversos entre eles Eucalyptus, Nimbus, OpenNebula, OpenStack, sendo apresentados uma breve explicação sobre cada

uma destas ferramentas e definindo algumas de suas principais diferenças qualitativas. Foi feito um estudo da infraestrutura de escalabilidade para cada uma das ferramentas. Como conclusão deste trabalho, os autores relatam que são fornecidas evidências de que existe a oferta de muitos *frameworks* IaaS, e que é necessário permitir ao usuário responder à questão de qual *framework* é melhor para ele.

3.1.7 Considerações sobre Trabalhos Relacionados

O presente trabalho busca fazer uma análise e comparação de ferramentas *open source* para computação em nuvem privada. A intenção é de que este possa servir como base para pesquisadores que possuem dificuldades em encontrar uma análise e comparação com ambiente de teste entre tantas ferramentas existentes no mercado. Devido a algumas circunstâncias (tempo e infraestrutura) não foi possível em primeiro momento analisar, comparar e testar todas as ferramentas *open source* existentes para a computação em nuvem.

Para que se possam aprofundar os conhecimentos foram buscados trabalhos que tivessem alguma relação com este. Poucos foram os que fazem uma análise em um ambiente de teste para efetuar as comparações e na maioria deles, apenas são comparadas quatro ferramentas. Neste trabalho apresenta-se uma introdução a várias ferramentas e ao final são selecionas algumas para teste em ambiente de estações de trabalho.

Os trabalhos relacionados, em sua grande parte não apresentam testes realizados. De acordo com o Quadro 3, eles mostram apenas comparações entre ferramentas a partir de informações retiradas de pesquisas e não de testes realizados em um ambiente próprio como no trabalho *“A Survey on Open Source Cloud Computing Solutions”.* Alguns apenas se limitam a citar diversas ferramentas e apresentar funcionalidades, características e onde são utilizados. Também demonstram ferramentas que podem ser utilizadas para realização de testes em nuvem e critérios para se avaliar uma ferramenta, assim é o caso do trabalho “*Evaluating Open-Source Cloud Computing Solutions”* e “*Cloud Testing Tool”.*

Trabalho (Seção)	Ferramentas Estudadas	Avaliação – Quant.	Ambiente
3.1.1 (KHURSHID, AL-NAYEEM, GUPTA, 2009)	Open Cirrus	Prática- 1	128 computadores HP DL160 com dois processadores quad core, 16GB de RAM e 2TB HD.
3.1.2 (ENDO, GONÇALVES, KELNER, SADOK, 2010)	Xen Cloud Platform (XCP), Nimbus, OpenNebula, Eucalyptus, TPlataform, Apache virtual Computing Lab(VCL*)*, e Enomaly Platform Computing Elastic	Qualitativa - 7	-
3.1.3 (BAI, LI, CHEN, TSAI E GAO 2011)	Ferramentas para testar nuvens (D-Cloud, PreFail, CloudSim, ETHZ, YCSB, Benchmark, CloudStone, Testbed, YETI, SOASTA, ITKO, CloudTesting)	Qualitativa	-
3.1.4 (MACHADO, 2011)	OpenQRM e Eucalyptus	Prática- 2	6 computadores *desktop*, com o sistema operacional Ubuntu 10.04 de 32 bits,
3.1.5 (VORAS et al. 2011)	OpenNebula, Eucalyptus, Ubuntu *Enterprise Cloud*, OpenQRM, Abiquo, Red Hat *Cloud Foundations*, Edition One, OpenStack, Nimbus, mOSAIC.	Qualitativa- 10	-
3.1.6 (LASZEWSKI, DIAZM WANG, FOX, 2012)	Eucalyptus, Nimbus, OpenNebula, OpenStack	Qualitativa- 4-	-
TCC (HENTGES, THOMÉ, 2013)	Eucalyptus, OpenNebula OpenStack, CloudStack OpenQRM, UEC, Abiquo Convirt, Nimbus, Apache VCL	Qualitativa-10 Prática- 2 (OpenNebula e OpenStack)	6 computadores *desktop*, com o sistema operacional Ubuntu Server 12.04 de 32 bits,

Fonte: Hentges, Thomé, Griebler, 2013.

Quadro 3: Trabalhos relacionados.

Trabalhos como “*Performance Evaluation of the Illinois Cloud Computing Testbed”* que utiliza computadores de alto desempenho e “Comparação de Ferramentas de *Software* Livre para Administração de Nuvem Privada” que utiliza de recursos de *hardware* limitado, mostram uma abordagem semelhante ao que foi proposto, pois elaboram um ambiente experimental e executam alguns testes para obter resultados sobre as ferramentas.

Trabalho como “*Cloud Testing Tools*” é de grande valia, pois apresenta diversas dicas do que deve ser avaliado em uma nuvem, além de apresentar uma boa quantidade de ferramentas que podem ser utilizadas para testar diversas características importantes em uma nuvem.

O trabalho *Comparison of Multiple Cloud Frameworks* faz uma comparação entre ferramentas IaaS, utilizado um ambiente para verificar o comportamento de cada uma delas. São demonstrados gráficos e tabelas de comparação com as características de cada ferramenta. Em relação ao presente trabalho, percebe-se que o intuito foi fazer uma tabela de comparação para poder verificar qual ferramenta é mais apropriada em determinados tipos de tarefas. Assim, serviu de auxilio pra melhorar a visualização das semelhanças e diferenças entre as ferramentas.

Neste trabalho fez-se um estudo detalhado das principais ferramentas *open source* e compará-las qualitativamente conforme a literatura descreve. Após a comparação qualitativa, foram testadas e avaliadas algumas ferramentas em um ambiente computacional de nuvem privada, utilizando exclusivamente estações de trabalho. Sendo isso, um dos principais diferenciais deste trabalho, além de oferecer um estudo amplo das características e funcionalidades de cada uma delas.

Mesmo que alguns trabalhos façam a comparação qualitativa de ferramentas que são estudadas neste trabalho, a contribuição é em relação à atualização e melhoria da comparação e descrição das características. Já na experimentação prática, neste trabalho testaram-se as ferramentas de administração de computação em nuvem que ainda não foram testadas, como por exemplo, o OpenNebula e o OpenStack, que até são citadas mas não passam por testes práticos.

Além disso, esse trabalho apresenta uma maior gama de características comparadas e explicadas com mais detalhes, que não são vistas nos outros trabalhos, como é o caso dos tipos de interface, gerenciamento de energia (se apresentam ou não), se possuem balanceamento de carga, escalabilidade e tolerância a falhas e quais são os tipos de rede suportados, armazenamento e métodos de autenticação que utilizam.

3.2 COMPARAÇÃO DAS FERRAMENTAS

Neste item foi realizada a comparação entre as ferramentas, utilizando-se de quadros (Quadro 4, 5 e 6) com os seguintes campos: Interface, Gerenciamento de energia, Balanceamento de carga, Rede, Armazenamento, Monitoramento, Integração, Virtualização, Segurança, Escalabilidade e Tolerância a falhas. Para a melhor exposição das diferenças e semelhanças. Abaixo a descrição de cada campo:

- Interface: O campo interface descreve a forma de acesso a ferramenta, sendo que a maioria das ferramentas possibilita acesso por linha de comando ou Interface *Web*.

- Gerenciamento de energia: Este campo demonstra se a ferramenta em questão possibilita o gerenciamento de energia e como isso pode ser feito, para isso são utilizadas ferramentas como o Sistema *Cluster Energy Saving*, *Power Management* ou até mesmo de sistema próprio da ferramenta.

- Balanceamento de carga: Determina se a ferramenta possui balanceamento de carga e como o realiza. Por exemplo, a partir de sistemas como *Elastic Load Balancer* e *Citrix NetScaler.*

- Rede: Neste campo são demonstradas os tipos de conexão que a ferramenta pode oferecer para as máquinas virtuais. A maioria das ferramentas possibilita a conexão por VLAN ou *Bridge*, ainda há algumas que possibilita utilizar *Open* Vswitch.

- Armazenamento: Demonstra os sistemas de armazenamento que podem ser utilizados para o armazenamento utilizados pela nuvem. Na sua maioria utilizam de AoE (*ATA Over Ethernet*), iSCSI e NFS (*Network File System*), LVM (*Logical Volume Management*).

- Monitoramento: Expõe se a ferramenta possibilita fazer o monitoramento e se permite analisar itens como: CPU, Memória, *Disk*, Máquinas Virtuais, Rede, Swap, disponibilidade dos recursos ou disponibilidade dos componentes da infraestrutura. Podendo ter o auxílio de ferramentas como o Nagios ou componentes próprios da ferramenta para realizar este monitoramento.

- Integração: Este campo descreve com quais plataformas é possível à integração da ferramenta. Sendo possível perceber Integrações com Amazon *Elastic Compute Cloud* (EC2), Amazon *Elastic Block Storage* (EBS), Amazon *Machine Image* (AMI), Amazon *Simple Storage Service* (S3), CloudBridge Amazon EC2, Ubuntu *Enterprise Cloud*, Eucalyptus, Cisco UCS, Cumulus (Amazon S3) e Amazon Identity and Access Management (IAM).

- Virtualização: Verifica quais são os virtualizadores suportados. Podendo ser identificado que os virtualizadores que tem compatibilidade são: Xen, KVM e Vmware ESXi, Hyper-V, Vmware Server, Vmware vSphere e XenServer.

- Segurança: é o tipo de segurança que a ferramenta dispõe. Sendo que na maioria, as ferramentas possibilitam autenticação e controle por usuários e grupos, algumas permitem integração com Active Directory e outras permitem usar autenticação LDAP (*Lightweight Directory Access Protocol*).

- Escalabilidade: verifica se a ferramenta apresenta algum mecanismo que permite escalabilidade.

- Tolerância a falhas: verifica quais ferramentas possuem tolerância a falhas.

Ferramenta	Interface	Gerenciamento de Energia	Balanceamento de carga
Eucalyptus	SSH e WEB	Possui	*Elastic Load Balancer*
OpenNebula	SSH e WEB (Sustone GUI, OCCI)	CLUES	Possui
OpenStack	SSH e WEB (Horizon)	*Power Management*	*Possui*
Cloud Stack	SSH e WEB	Possui	Citrix *NetScaler*
OpenQRM	SSH e WEB	Possui	Possui
UEC	SSH e WEB	UEC *Power Management*	Não
Abiquo	SSH e WEB	Não	Possui
Convirt	SSH e WEB	Não	Não
Nimbus	SSH e WEB	Possui	Não
Apache VCL	SSH e WEB	Possui	Não

Fonte: Hentges, Thomé, Griebler, 2013.

Quadro 4: Comparação das ferramentas I.

Em relação à interface, as ferramentas apresentam duas maneiras de administrar à nuvem, sendo a primeira através de SSH que é uma conexão segura entre o cliente e o servidor (efetuada por linha de comando). A outra forma de acesso é através do *browser* por HTTP, onde é feita a conexão ao *host* através de uma página *web* que possibilita administrar a nuvem. As ferramentas Eucalyptus, OpenNebula, OpenStack, CloudStack, OpenQRM, Ubuntu *Enterprise Cloud*, Abiquo, Convirt, Nimbus e Apache *Virtual Computing Lab* apresentam a opção de se conectar através do *browser*, sendo isso uma vantagem, pois a interface *web* torna mais fácil a administração da nuvem, operações podem ser realizadas através de uma interface gráfica. Além disso, todos apresentam conexão via SSH, que também se torna importante, por ser uma maneira segura e simples de se conectar a nuvem

Gerenciamento de energia é algo importante em uma nuvem, pois possibilita diminuir os custos em energia elétrica. O Eucalyptus possui um gerenciamento de energia que é feito através da suspensão das máquinas que não estão em uso. O OpenNebula utiliza-se do sistema CLUES (*Cluster Energy Saving*) e também pode utilizar o *Green Cloud Scheduler*, que desliga os nós quando não estão em uso. Já o OpenStack, oferece gerenciamento de energia através de extensões, que somente funcionam junto a processadores Intel Xeon, onde são monitoradas as cargas de trabalho, que possibilita reduzir ao máximo o consumo quando o processador está ocioso. No CloudStack, o gerenciamento de energia depende da demanda atual em um *cluster*, pois coloca os *hosts* em modo *standby* para economizar energia.

O OpenQRM possibilita gerenciamento de energia utilizando o sistema administrador para encontrar recursos não utilizados ou em baixo uso para ou desliga-los ou migra-los para outros servidores. O Ubuntu *Enterprise Cloud* possui o UEC *Power Management*, onde são desligados nós quando os mesmos não estão em uso. Nimbus apresenta gerenciamento de energia, movendo máquinas virtuais em outros servidores para economizar energia. O Apache Virtual Computing Lab realiza gerenciamento de energia colocando em *standby* recursos que não são utilizados momentaneamente. Já o Abiquo e o Convirt são ferramentas que não apresentam nenhum tipo de gerenciamento de energia.

Balanceamento de carga é importante em uma nuvem, pois possibilita a divisão das tarefas para melhor aproveitar os recursos computacionais disponíveis. O Eucalyptus possui o *Elastic Load Balancer* que distribui o tráfego de entrada das aplicações entre os nós do *Cluster*. No OpenNebula a execução das máquinas virtuais instaladas são divididas entre os diversos *hosts*. Já o OpenStack divide as máquinas virtuais entre os nós de acordo com a carga já exercida em cada nó.

O CloudStack utiliza-se do Citrix NetScaler para dividir as tarefas entre os nós que executam os aplicativos. O OpenQRM faz o balanceamento de carga dos recursos como memória e processamento do *cluster* para execução dos processos. Abiquo utiliza o balanceamento de carga para dividir entre os nós que rodam a mesma aplicação o número de conexões. As ferramentas UEC, Convirt, Nimbus e Apache VCL não apresentam nenhum tipo de balanceamento de carga.

Ferramenta	Rede	Armazenamento	Monitoramento
Eucalyptus	Bridge e VLAN	AoE, iSCSI e NFS	Nagios
OpenNebula	Bridge, VLAN e *Open* Vswitch	NFS, ISCSI, LVM	OpenNebula Sunstone
OpenStack	VLAN e *Open* Vswitch	AoE, iSCSI e NFS	OpenStack Clanavi
Cloud Stack	VLAN	ISCSI e NFS	*Traffic Sentinel*
OpenQRM	Bridge e VLAN	NFS, iSCSI, AoE e LVM	openqrm-monitord
UEC	Bridge e VLAN	iSCSI e AoE	UEC *Monitor*
Abiquo	VLAN	NFS, iSCSI, LVM	Abiquo *Monitor*
Convirt	VLAN	NFS, iSCSI e LVM	Convirt *Monitor*
Nimbus	VLAN	AOE, iSCSI e NFS	Nagios
Apache VCL	VLAN	iSCSI	Não

Fonte: Hentges, Thomé, Griebler, 2013.

Quadro 5: Comparação das ferramentas II.

As ferramentas utilizam-se de alguns métodos para fornecer conexão as máquinas virtuais. Dentre eles podem ser citadas a VLAN que é uma forma de dividir uma rede física em diversas redes lógicas, possibilitando a criação de domínios onde *hosts* são divididos para comunicarem em segurança entre si. Dentre as ferramentas, todas elas apresentam esta opção. O *Bridge* permite conectar duas ou mais redes distintas, criando uma rede agregada para que estas se comuniquem, dentre as ferramentas, somente OpenNebula, OpenQRM e Ubuntu Enterprise Cloud possibilitam este método. Já o Open vSwitch é um *software* que cria um *switch* virtual que tem a função de encaminhar o tráfego de máquinas virtuais, dentro de um mesmo *host* ou até mesmo entre a máquina virtual e a rede física. As ferramentas que apresentam esta característica são o OpenNebula e o OpenStack.

O *Storage* ou armazenamento, pode ser realizado utilizando de algumas maneiras, como iSCSI (*Internet Small Computer System Interface*) que é um protocolo de transporte de comandos SCSI, ele é usado em uma rede de armazenamento onde dados são armazenados em diversos *hosts* de uma rede, que é utilizado por todas as ferramentas. O AoE (ATA *Over Ethernet*) é um protocolo de rede para dar alta performance e acesso a dispositivos de armazenamento SATA através da rede interna e está presente nas ferramentas Eucalyptus, OpenStack, OpenQRM, Ubuntu *Enterprise Cloud* e Nimbus.

O NFS (*Network File System*) é um sistema de arquivos em que diretórios são compartilhados entre os computadores em uma rede, permitindo que todos os computadores tenham acesso local aos dados (ferramentas que possibilitam este protocolo são Eucalyptus, OpenNebula, OpenStack, CloudStack, OpenQRM e Nimbus). O LVM (*Logical Volume Management*) é usado para criar um grande disco virtual que pode conter mais de um dispositivo de armazenamento e possibilitar a divisão em partições virtuais (disponível nas ferramentas OpenNebula, OpenQRM, Abiquo e Convirt).

Eucalyptus e Nimbus utilizam como sistema de monitoramento o Nagios para monitorar recursos como CPU, memória, HD e VMs. O OpenNebula utiliza do sistema próprio OpenNebula Sunstone que faz o gerenciamento e monitoramento da nuvem através de uma interface *web*. Da mesma forma o OpenStack utiliza do

sistema próprio Clanavi e o CloudStack utiliza o sistema Traffic Sentinel da inMon. O OpenQRM também possui um sistema próprio chamado de openqrm-monitord para monitorar os recursos computacionais.

Ferramenta	Integração	Virtualização	Segurança
Eucalyptus	EC2, EBS, AMI, S3, IAM	Xen, KVM e Vmware ESXi	Autenticação, CUG e *Active Directory*
OpenNebula	EC2	Xen, KVM e Vmware	Autenticação e CUG
OpenStack	EC2 e S3	XenServer, KVM e Hyper-V	*Keystone*, LDAP, e métodos externos
Cloud Stack	CloudBridge e EC2	Xen, KVM e Vmware ESXi	Autenticação e CUG
OpenQRM	UEC, EC2 e Eucalyptus	Vmware ESX, Xen, KVM e XenServer	Autenticação, CUG e LDAP
UEC	EC2	KVM	Autenticação e CUG
Abiquo	Cisco UCS	VMware ESXi, Hyper-V, XenServer, Xen, KVM	Autenticação, CUG e LDAP
Convirt	EC2	Xen e KVM	Não
Nimbus	EC2, S3, Cumulus	Xen e KVM	Autenticação e CUG
Apache VCL	Não	Vmware, KVM	Autenticação LDAP

Fonte: Hentges, Thomé, Griebler, 2013.

Quadro 6: Comparação das ferramentas III.

No Quadro 6 são descritas se as ferramentas possuem integração com algum outro serviço. A ferramenta Eucalyptus permite integração com os seguintes serviços da Amazon EC2, EBS, AMI, S3, IAM. Isso permite um alto nível de compatibilidade entre as suas API's e dessa forma todos os componentes funcionam sem a necessidade de serem modificados em uma nuvem privada baseada no Eucalyptus e na nuvem pública da Amazon.

O OpenNebula permite integração com os serviços EC2 e S3 da Amazon, pois além de suportar nuvens privadas, também suporta nuvens híbridas, o que permite fazer a combinação de uma infraestrutura privada com uma nuvem pública da Amazon, possibilitando níveis ainda maiores de escalabilidade. Já o OpenStack permite integração com os serviços EC2, S3 da Amazon. Isso porque esses serviços permitem maior compatibilidade entre as API's e também devido a grande quantidade de códigos existentes.

O CloudStack por ser expansível e oferecer suporte multi-*hypervisor* permite integração com o serviço EC2 da Amazon, interagindo de forma continua com os

parceiros da Amazon. Já o CloudStack permite integração com o serviço CloudBridge da Citrix que proporciona uma plataforma integrada, que conecta aplicativos e melhora a utilização da largura de banda em nuvem pública e redes privadas. O OpenQRM também permite a integração com serviço EC2 da Amazon para configurações de nuvens hibridas. Além disso, permite integração com as ferramentas UEC e Eucalyptus.

UEC permite integração com EC2 da Amazon, pois esse serviço possui maior compatibilidade com as API's e fornece melhor desempenho. O Abiquo permite integração com Cisco UCS, que facilita a mudança para a computação em nuvem e *Infrastructure-as-a-Service*. A ferramenta Convirt permite integração com EC2 da Amazon, pois este serviço fornece um ambiente de execução estável, seguro e de alto desempenho para os aplicativos.

A ferramenta Nimbus permite integração com EC2 e S3, com isso ele aloca e gerencia recursos remotos fornecidos pela Amazon EC2 e por haver dependência com o EC2, ele suporta Xen e KVM/Linux. Nimbus, também permite integração com Cumulus, sendo um sistema *open source* de armazenamento em nuvem que implementa interfaces compatíveis com o S3 REST API da Amazon. Para a ferramenta Apache VCL não foram encontrados dados que indiquem que essa ferramenta faz integração com outra/s.

O Quadro 6 traz também informações sobre a virtualização, as ferramentas Eucalyptus, OpenNebula, CloudStack, OpenQRM, Abiquo e Nimbus permitem virtualização com Xen. Todas as ferramentas permitem virtualização utilizando KVM. Eucalyptus e CloudStack permitem virtualização com VMware ESXi. OpenNebula e Apache VCL permitem virtualização com VMware. OpenStack, OpenQRM e Abiquo permitem com Xen Server. OpenStack e abiquo podem utilizar também Hyper-V. E Abiquo e o Apache VCL podem utilizar Virtualbox.

A tecnologia Xen permite a execução de vários sistemas operacionais, simultaneamente, sobre um mesmo *hardware*, ou seja, pode ser atrelado diretamente ao *hardware*. O KVM é uma infraestrutura de virtualização, integrada ao Linux que permite executar várias máquinas virtuais sem a necessidade de um

sistema operacional. E a tecnologia VMware ESXi não necessita de sistema operacional e pode ser integrado diretamente aos servidores. A tecnologia VMware é um *software* para máquina virtual que possibilita a instalação de um sistema operacional dentro de outro, com ele é possível executar um ou mais sistemas operacionais em ambientes isolados e de maneira simultânea.

A tecnologia Xen *Server* é utilizada para virtualização de máquinas. Com ela é possível colocar várias máquinas virtuais em uma máquina física, reduzindo o número de servidores físicos. O Hyper-V é uma tecnologia presente no Windows Server 2012, fornece infraestrutura de *software* e ferramentas para gerenciar ambientes de virtualização de servidores. E o Virtualbox permite a criação de ambientes para instalar sistemas operacionais diferentes, também a instalação de um sistema operacional dentro de outro, mas compartilham fisicamente o mesmo *hardware*.

Por fim, o Quadro 6 apresenta informações sobre segurança em relação as ferramentas. As ferramentas Eucalyptus, OpenNebula, CloudStack, OPenQRM, UEC, Abiquo e Nimbus no que se trata de segurança permitem autenticação e controle por usuários e grupos (CUG). A autenticação serve para confirmar a veracidade e origem das informações. O controle por usuários e grupos é utilizado para que se possa saber se o usuário ou grupo possui acesso a determinadas informações auxiliando na confiabilidade e integridade das mesmas.

A ferramenta Eucalyptus permite integração com *Active Directory*, pois ele permite organizar, gerenciar e localizar recursos na rede e realiza a autenticação dos usuários. O OpenStack, OpenQRM, Abiquo e o Apache VCL permitem usar autenticação por LDAP, que assim como o *Active Directory* realiza autenticação dos usuários, assim as aplicações se conectam e consultam um servidor de diretórios que possam acessar. A ferramenta OpenStack ainda pode utilizar do serviço *Keystone*, que é próprio dela e é um serviço de identificação utilizado para autenticação e autorização dos usuários.

A escalabilidade é uma característica importante presente em uma nuvem, pois tem a responsabilidade de atender a demanda crescente de requisições de

hardware, processamento, serviço ou de um aplicativo, agregando nós ou recursos que antes podiam estar sendo utilizados por outros serviços para suprir esta necessidade imediata. Todas as ferramentas apresentam esta característica, que é necessário por muitos que desejam implementar uma nuvem, portanto se tornando uma característica praticamente indispensável.

A tolerância a falhas é essencial aos ambientes de nuvens e se torna cada vez mais importante, pois esses ambientes precisam fornecer mecanismos de checagem de erros de *software* e de *hardware*, além de prover recuperação em caso de falhas, sendo assim todas as ferramentas possuem tolerância a falhas.

Com relação ao gerenciamento de energia, as ferramentas Abiquo e Convirt não são indicadas, pois as mesmas não realizam tal ação. Se a necessidade for balanceamento de carga, dentre as ferramentas avaliadas somente o Eucalyptus, OpenNebula, OpenStack, CloudStack, OpenQRM e Abiquo realizam esta tarefa, logo, estas são mais indicadas a quem tem essa necessidade. Em relação à rede, se houver a necessidade de conectar diferentes redes, por exemplo, realizar a comunicação entre *clusters*, são indicadas as ferramentas Eucalyptus, OpenNebula, OpenQRM e Ubuntu *Enterprise Cloud*, por possuírem o método de conexão *bridge*, além do método VLAN presentes em todas as ferramentas. Se for preciso uma maior variedade de métodos de armazenamento são indicados o Eucalyptus, OpenNebula, OpenStack, OpenQRM, Abiquo, Convirt e Nimbus, por apresentarem três ou quatro métodos.

Quando é indispensável o monitoramento tanto do *hardware* que compõem a nuvem, quanto das máquinas virtuais que operam nela, não é indicada a utilização do Apache VCL, pois este não apresenta nenhuma alternativa de monitoramento, diferente das outras ferramentas. Embora todas as ferramentas citadas na tabela ofereçam integração, exceto o Apache VLC, ao se deparar com um cenário em que se tem como prioridade a integração com diferentes nuvens, as ferramentas Eucalyptus, Nimbus e OpenQRM parecem ser mais apropriadas, por possuírem mais opções de integração.

Ao se deparar com um ambiente em que existe grande heterogeneidade arquitetural, ter diversas ferramentas de virtualização torna-se uma vantagem. Nestes casos, é mais indicado a utilização das ferramentas OpenQRM e Abiquo, por possibilitarem a utilização de uma grande quantidade de virtualizadores. Se for de desejo utilizar uma base de dados externa de autenticação, são indicados o Eucalyptus, OpenStack, OpenQRM, Abiquo e Apache VCL, por possibilitarem autenticação LDAP ou integração com o *Active Directory*.

Considerando todas as características avaliadas, a ferramenta que acaba oferecendo uma grande gama de opções, além de possuir todas as características é o OpenQRM, que como mostrado na comparação anterior, é o mais citado, ou seja é mais completo em relação aos outros. Outras ferramentas que também se destacaram foram o OpenNebula e o OpenStack.

3.3 PLANEJAMENTO DO EXPERIMENTO DE TESTES

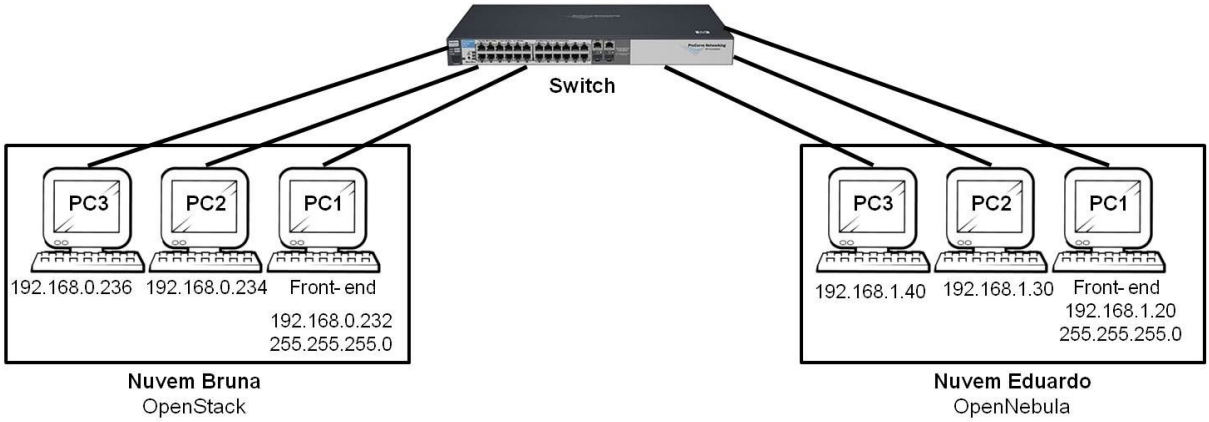
Esta seção demonstra o planejamento dos experimentos, ou seja, como foram efetuados os testes para avaliação das ferramentas em um ambiente computacional, exclusivamente com estações de trabalho.

3.3.1 Infraestrutura

A infraestrutura física utilizada como ambiente de testes foi constituída de 6 máquinas com as seguintes características: processador Intel Core I5 650 de 3.20 Ghz, memória 4Gb DDR3 1333Mhz, HD 500GB SII e rede *onboard* 10/100.

Tendo um *Desktop* utilizado como servidor possuindo: processador Intel Core I5 650 de 3.20 Ghz, memória 4Gb DDR3 1333Mhz, HD 500GB SII. E um *switch* de 24 portas.

A Figura 19 demonstra a rede física, sendo que cada nuvem (OpenNebula e OpenStack) possuia uma rede isolada mesmo conectadas ao mesmo *switch*, cada nuvem foi formada por três máquinas sendo um *frontend* e dois nós.



Fonte: Hentges, Thomé, Griebler, 2013.
Figura 19: Arquitetura rede.

3.3.2 Software

Para realização dos testes, as estações de trabalho que compõem a nuvem utilizaram o sistema operacional o Ubuntu *Server* 12.04 por se tratar de um sistema operacional estável e compatível com as ferramentas que foram instaladas. E as ferramentas utilizadas para implementar a nuvem privada nos testes com estações de trabalho foram: o OpenNebula e o OpenStack, que saíram-se bem nos testes, além de serem ferramentas que não foram testadas na prática em nenhum dos trabalhos relacionados.

3.3.3 Cenário de Teste

Para o cenário de teste as seguintes características foram avaliadas:

- Interface: Para testar a interface, foi verificado se tudo que está presente naquela interface está funcional. A interface necessita ser o mais intuitiva possível, pois isso irá auxiliar no acesso e agilizar o trabalho.

- Monitoramento: Nos testes de monitoramento, foi verificado os dados fornecidos pelo monitoramento, além de ser visto qual das ferramentas testadas apresenta maior número de informações

monitoradas e se revela informações mais detalhadas sobre cada item monitorado.

- Tolerância a falhas: neste teste a nuvem foi submetida ao desligamento de um nó, e para provar a sua tolerância a nuvem terá que se recuperar sozinha deste acontecimento.

- Gerenciamento de energia: neste caso foi explicado a maneira que ocorre o gerenciamento de energia em cada nuvem.

- Balanceamento de carga: neste caso, foram criadas algumas máquinas virtuais na nuvem, e será visto se essas VM's serão divididas entre os nós responsáveis por suas execuções.

- Escalabilidade: neste teste foi adicionado um nó a nuvem, e será visto o comportamento da nuvem.

- Segurança: neste teste foi visto como funciona o sistema de autenticação.

- Rede: neste caso foram explicadas as diferentes maneiras que podem ser utilizadas nas máquinas virtuais.

- Armazenamento: neste caso foram expostos os métodos de armazenamento e como cada um pode ser aplicado.

- Integração: foi explicado como pode ser realizada a integração com outras nuvens.

- Virtualização: foi explicado como pode ser utilizado os diferentes métodos de virtualização.

3.4 INSTALAÇÃO DAS FERRAMENTAS

Nesta seção é descrita de forma rápida os procedimentos para realizar as instalações, o que foi necessário instalar e configurar para deixar a nuvem funcionando.

3.4.1 Instalação do OpenNebula

Foi feita a instalação da versão do OpenNebula 4.2 (lançada em julho de 2013). No Apêndice A pode ser verificado a instalação e configuração da nuvem utilizando o OpenNebula. Primeiramente foi realizada a instalação no *frontend,* ou seja o computador onde é instalado de fato o OpenNebula, onde foram armazenadas as imagens, este nó é responsável pelo gerenciamento da nuvem, pela adição e remoção de nós, criação de máquinas virtuais e onde cada máquina virtual irá executar, nele primeiramente foi feita a configuração da rede e a criação do usuário que gerencia a nuvem (Seção A.1), posteriormente foi instalado o NFS-*kernel-server e* gerada a chave SSH para o usuário oneadmin *(Seção A.2),* após isto foi feita a instalação das dependências do OpenNebula e a instalação da ferramenta em si, além de sua configuração (Seção A.3) e por último foi feita a instalação das dependências e do serviço Sunstone-*server* (Seção A.4).

Nos computadores que cumpriram a função de nós (*worker nodes*) tiveram que ser instalados e configurados primeiramente a rede (Seção A.5.1), então foi feita a instalação e configuração do NFS-*kernel-server* (Seção A.5.2), posteriormente foi realizada a instalação dos *softwares* de virtualização (Qemu-KVM, Libvirt e VM-ubuntu-builder) (Seção A.5.3).

Após a instalação e configuração básica tanto do *frontend* como dos nós, a nuvem pode ser iniciada, e a partir disso foram adicionados os nós e foram criadas as *Virtual Networks,* adicionadas as imagens ao *datastore*, e puderam ser criadas as máquinas virtuais.

Primeiramente houve um erro onde não era possível monitorar os nós, e que após pesquisa foi descoberto que era necessário uma alteração nos arquivos

qemu.conf e libvirtd.conf, após isto, os nós estavam de forma correta adicionados a nuvem.

No decorrer do processo apareceram alguns problemas relacionados à execução das máquinas virtuais na nuvem, que através da avaliação dos *logs* dos erros (que são apresentados pela interface Sunstone), onde informava que era impossível executar a máquina virtual em determinado nó por falta de permissão. Para realizar a correção foi preciso apenas dar permissão para o usuário oneadmin sobre o KVM, o qual era o responsável pela execução das máquinas virtuais nos nós. Assim, as máquinas virtuais puderam executar livremente nos nós. Desta forma a nuvem estava instalada de maneira básica e capaz de executar máquinas virtuais.

3.4.2 Instalação do OpenStack

Foi instalada a versão do OpenStack conhecida como Folsom (lançada em outubro de 2012). No Apêndice B pode ser verificado a instalação e configuração da nuvem OpenStack. Inicialmente foi configurada uma máquina principal como *Controller Node*, nesta são instalados e configurados os seguintes componentes: *Keystone*, *Glance*, *Cinder*, os componetes Nova *Services* e o Horizon. Sendo o *Keystone* o componente de Identificação, o *Glance* gerencia as imagens das máquinas virtuais, o *Cinder* responsável pelo armazenamento, o Nova faz o gerenciamento da infraestrutura do OpenStack e o Horizon componente responsável pelo *Dashboard* (Seção B.1.1). No *Controller Node* também são instalados e configurados o MySQL, NTP, RabbitMQ, VLAN, Bridge-utils (Seção B.1).

Em outra máquina foi configurado um *Compute Node*, onde foram instalados e configurados os componentes Nova *Compute*, Nova *Network* e KVM para virtualização, pois as máquinas possuem suporte a virtualização em outros casos caso as máquinas não tivessem suporte a virtualização poderia ser instalado o QEMU. Também foram instalados e configurados o NTP, VLAN, Bridge-utils e Libvirt-bin (Seção B.2).

Após as instalações e configurações básicas necessárias foi utilizado o *Dashboard* para criar Projetos, Usuários e também Instâncias (máquinas virtuais).

Depois das configurações necessárias no painel é possível o usuário verificar suas máquinas virtuais rodando e até mesmo acessa-las via SSH, utilizando de usuário e senha para *login* (Seção B.2).

Durante a instalação e configuração também ocorreram alguns imprevistos. Nas primeiras tentativas de instalação do OpenStack era possível logar no painel, porém não era possivel criar as instâncias e as imagens dos sistemas operacionais não carregavam. Depois de uma série de tentativas e erros foi possível fazer funcionar tudo como deveria, o problema estava nos IPs configurados, na autenticação dos usuários e também no tipo de imagem utilizada na criação das máquinas virtuais. Uma dica importante e que facilita bastante é utilizar imagens que são de um sistema operacional previamente instalado ou imagens que sejam voltadas para a criação em nuvens.

3.5 AVALIAÇÃO DAS FERRAMENTAS

Nesta parte serão apresentados os resultados que foram obtidos a partir dos testes realizados nas ferramentas.

3.5.1 OpenNebula

O OpenNebula é uma ferramenta *open source* que é utilizada para a criação de nuvens que forneçam o modelo de serviço IaaS. Nesta seção são descritos os testes feitos em ambiente com estações de trabalho em que foram instalados esta ferramenta.

3.5.1.1 Interface

A interface de gerenciamento do OpenNebula (Sunstone) apresenta simplicidade, sendo de fácil entendimento. Ela apresenta o necessário para que se possa administrar a nuvem, apresentando usuários, grupos de usuários, *hosts, clusters,* imagens de VM's, *templates* para criação de VM's*,* unidades de armazenamento e máquinas virtuais criadas. Nesta interface é possível criar novas VM's, novas *templates,* novos usuários, etc. Praticamente tudo que pode ser feito via

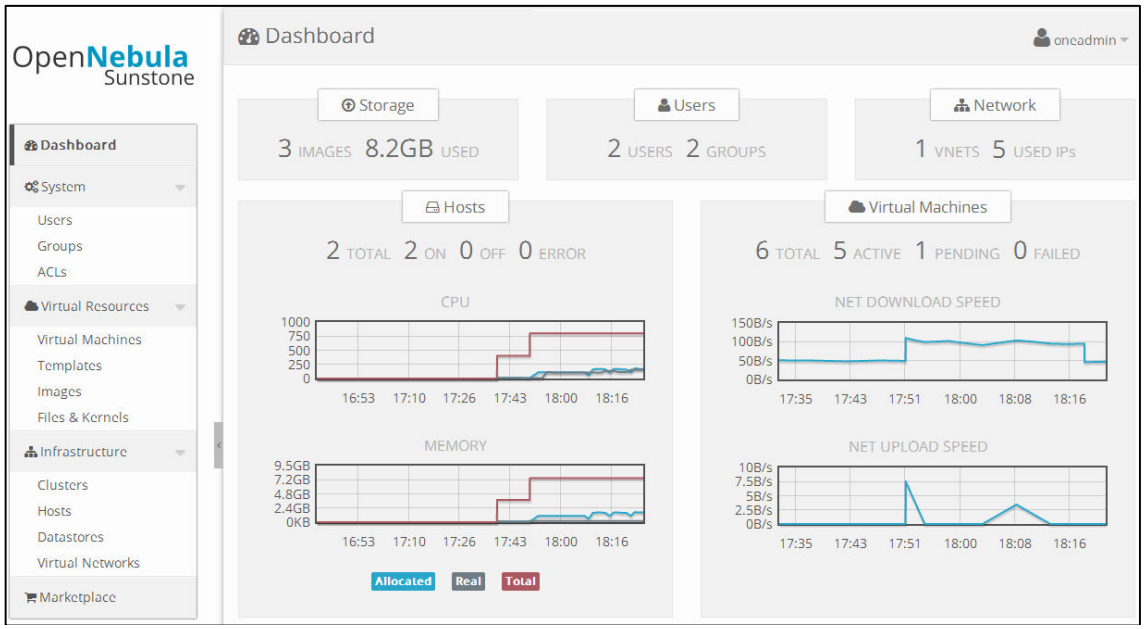
linha de comando é possível executar através da interface *web*. Apresenta uma interface clara que mostra dados importantes relacionados a monitoramento, tanto dos *hosts*, como das VM's que estão em execução no momento. Além disso, apresenta diversos idiomas dentre eles o Português-BR.

Em relação à interface do usuário, pode ser utilizada o Sunstone, porém nela são apresentados apenas elementos importantes para os usuários, como as VM's, as imagens disponíveis para o usuário e a rede disponível para o usuário. Algumas opções como gerenciar e criar, podem ser removidas do usuário através de ACL's criadas pelo administrador. O usuário também tem a opção em escolher entre diversos idiomas dentre eles o Português-BR.

A Figura 20 mostra a tela de *login* da interface Sunstone.

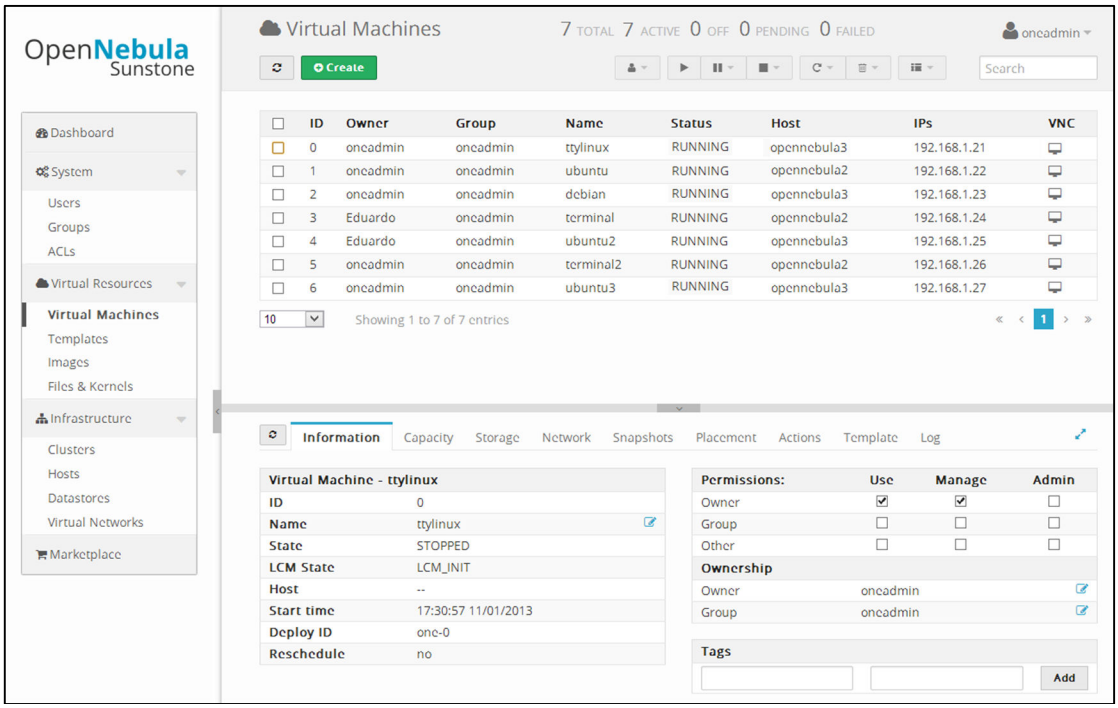
Fonte: Hentges, Thomé, Griebler, 2013.
Figura 20: Tela de login do OpenNebula

A Figura 21 apresenta a tela inicial do OpenNebula, onde é mostrado o número de *hosts* que compõem a nuvem e o estado dos mesmos, o número de máquinas virtuais criadas e o estado das VM's. Além disso, mostra graficamente o uso de memória total da nuvem, o uso total de processamento e a taxa de download e upload das máquinas virtuais.



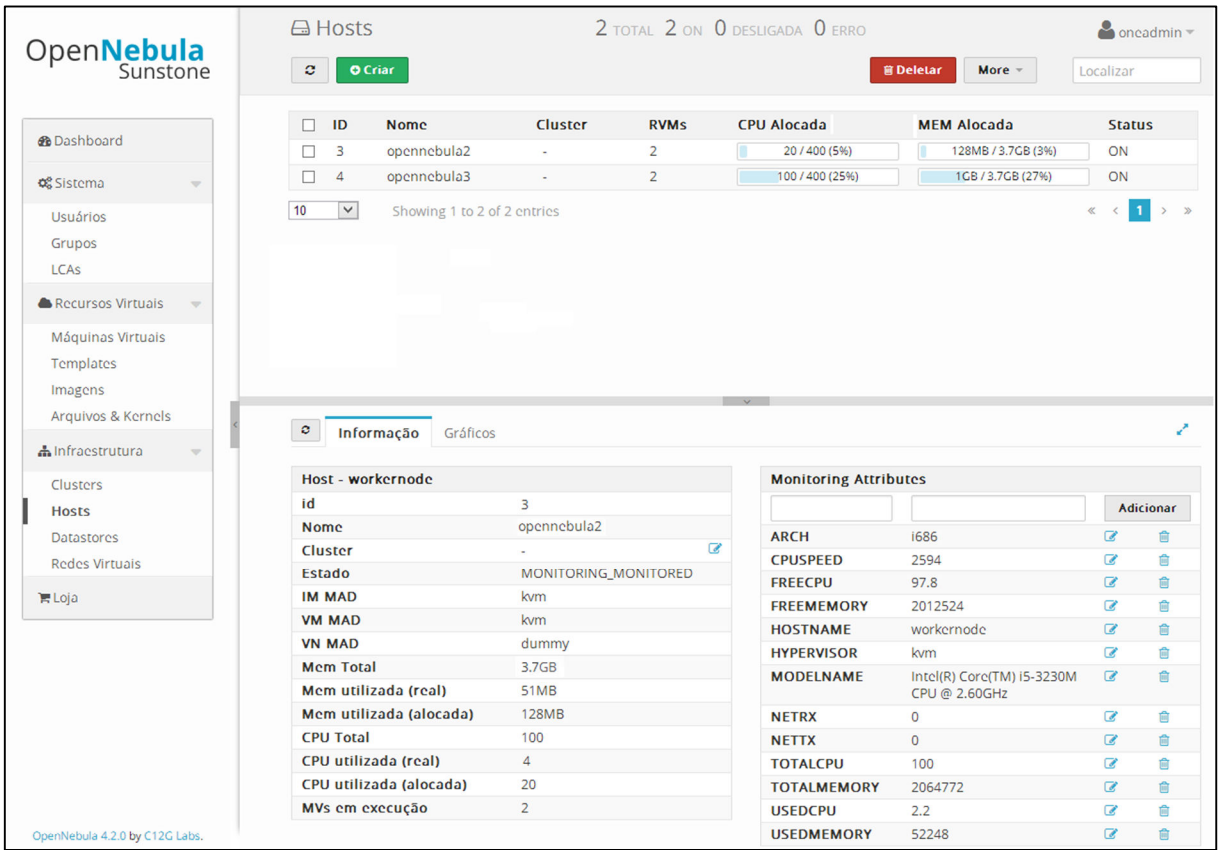
Fonte: Hentges, Thomé, Griebler, 2013.
Figura 21: Tela principal OpenNebula

A Figura 22 mostra as máquinas virtuais existentes na nuvem, onde é apresentado a ID do proprietário, seu nome, o *status*, nó em que está executando e IP. Nesta página é possível criar uma nova máquina virtual, deletar alguma já existente e executar alguma ação sobre as máquinas virtuais, como por exemplo: desligar, ligar, reiniciar, migrar.



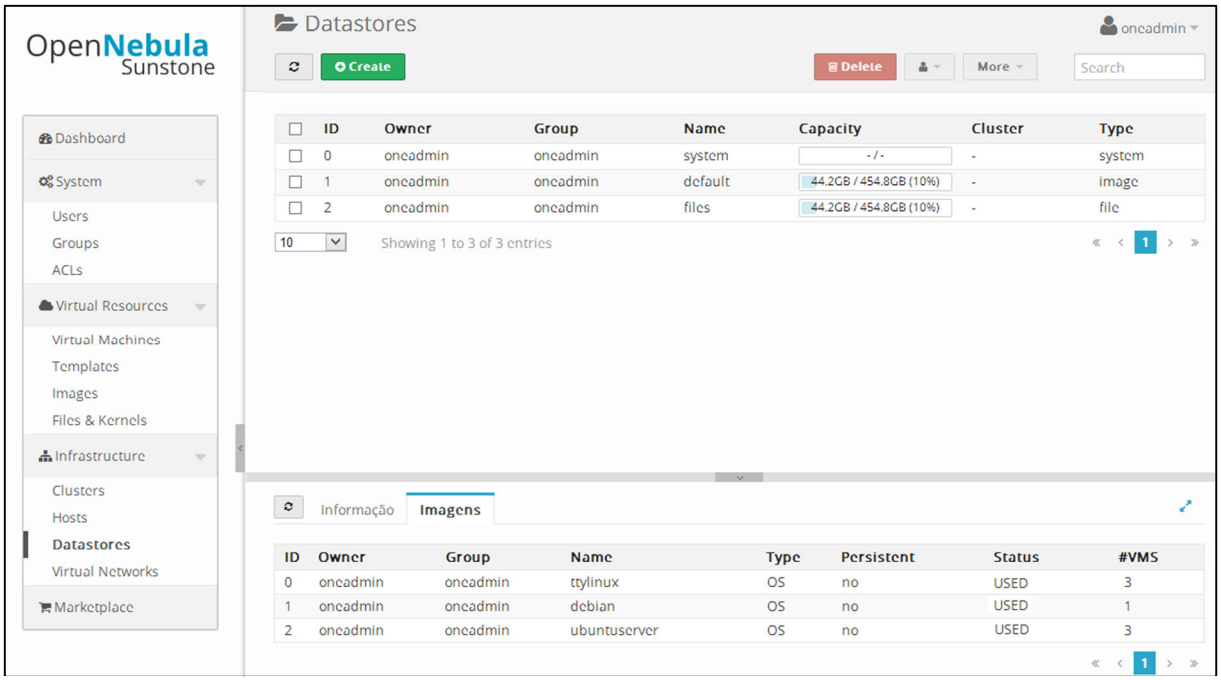
Fonte: Hentges, Thomé, Griebler, 2013.
Figura 22: Máquinas virtuais

A Figura 23 apresenta os nós que compõem a nuvem, nele são apresentados o ID, o nome do *host*, quantas máquinas virtuais estão executando em cada nó, o uso de processamento, o uso de memória e o status do nó, que pode ser *ON* (está ativo e funcional), *UPDATE* (está atualizando as informações de status), *OFF* (está desabilitado, não funcional), *RETRY* (tentar novamente obter as informações sobre o *host*) e *ERROR* (não consegue obter informação sobre o nó).



Fonte: Hentges, Thomé, Griebler, 2013.
Figura 23: Nós

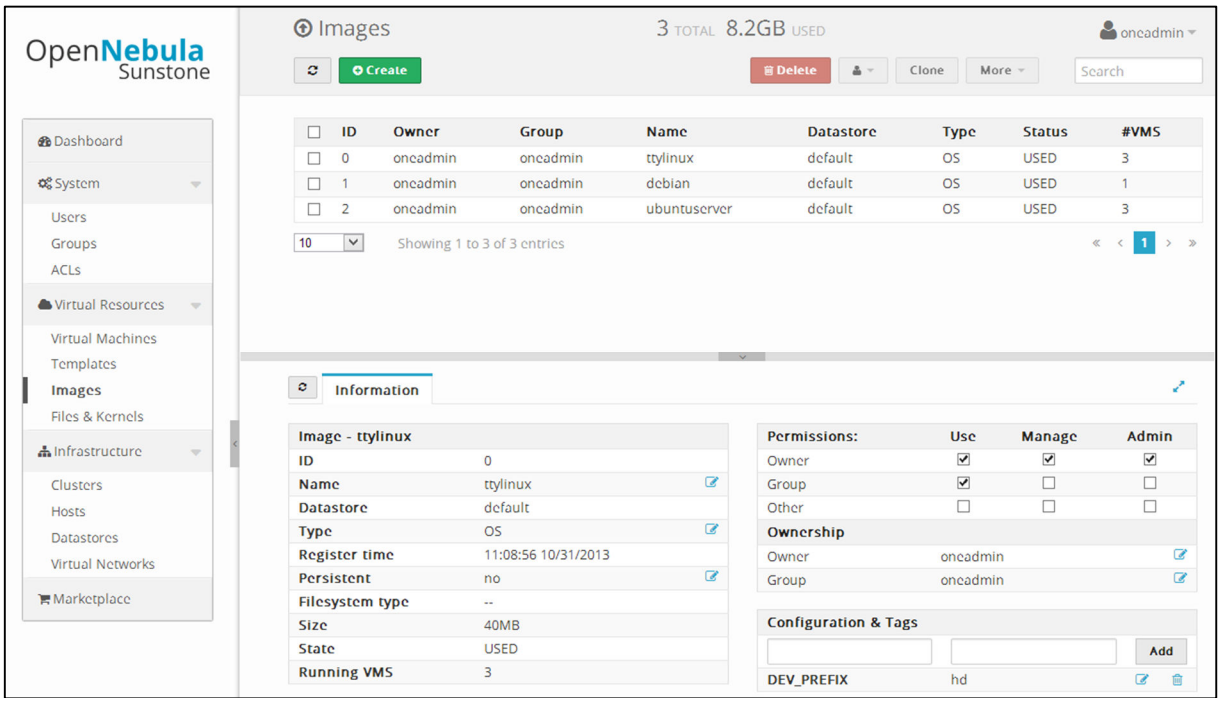
A Figura 24 mostra os *datastores* (lugares em que são armazenadas as imagens, e também arquivos que podem ser passados para as máquinas virtuais) presentes na nuvem, apresenta a capacidade e quanto foi utilizado de cada *datastore*. Além disso, podem ser adicionados mais *datastores* conforme a necessidade.



Fonte: Hentges, Thomé, Griebler, 2013.

Figura 24: *Datastores*

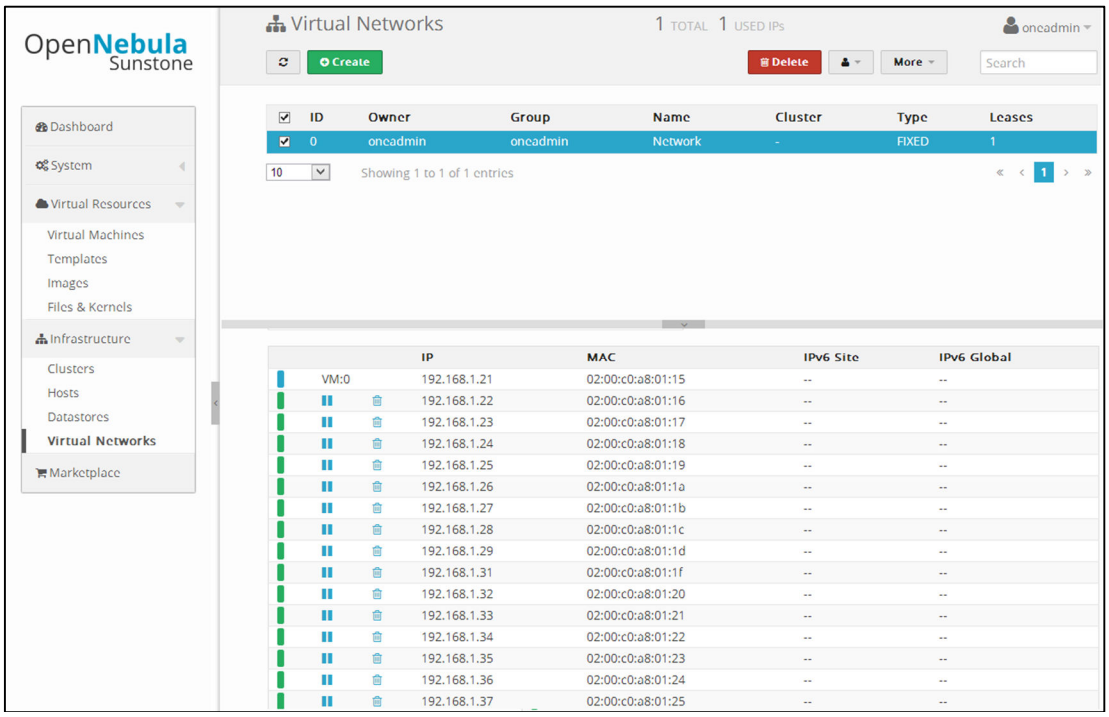
A Figura 25 exibe a tela das imagens, nela são mostradas todas as imagens das máquinas virtuais que são utilizadas para dar origem a novas máquinas virtuais.



Fonte: Hentges, Thomé, Griebler, 2013.

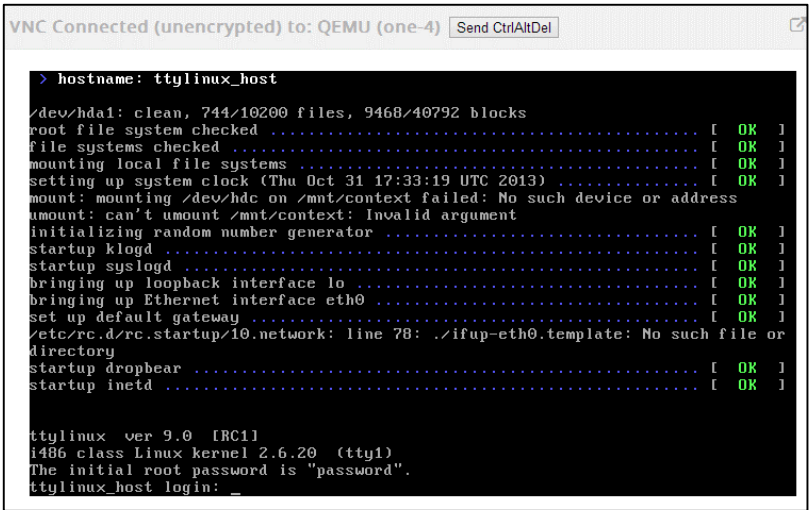
Figura 25: Imagens

A Figura 26 mostra a tela onde podem ser criadas as redes virtuais, redes essas que são as que definem os IP's para as máquinas virtuais que são criadas. É possível criar novas redes e apagar as já existentes.



Fonte: Hentges, Thomé, Griebler, 2013.
Figura 26: Rede

A Figura 27 mostra o acesso por meio do noVNC que é integrado juntamente ao OpenNebula, e que possibilita visualizar a máquina virtual em execução.

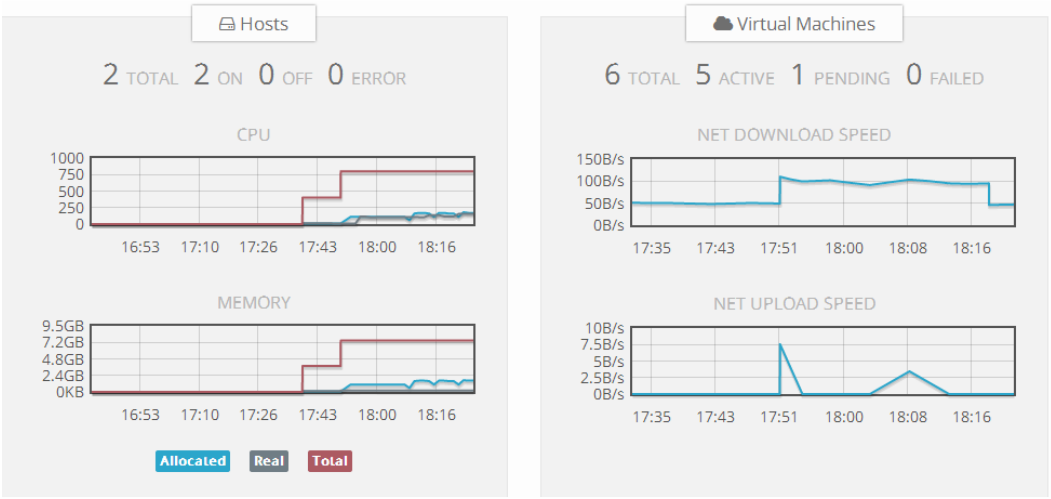


Fonte: Hentges, Thomé, Griebler, 2013.
Figura 27: noVNC

3.5.1.2 Sistema de Monitoramento

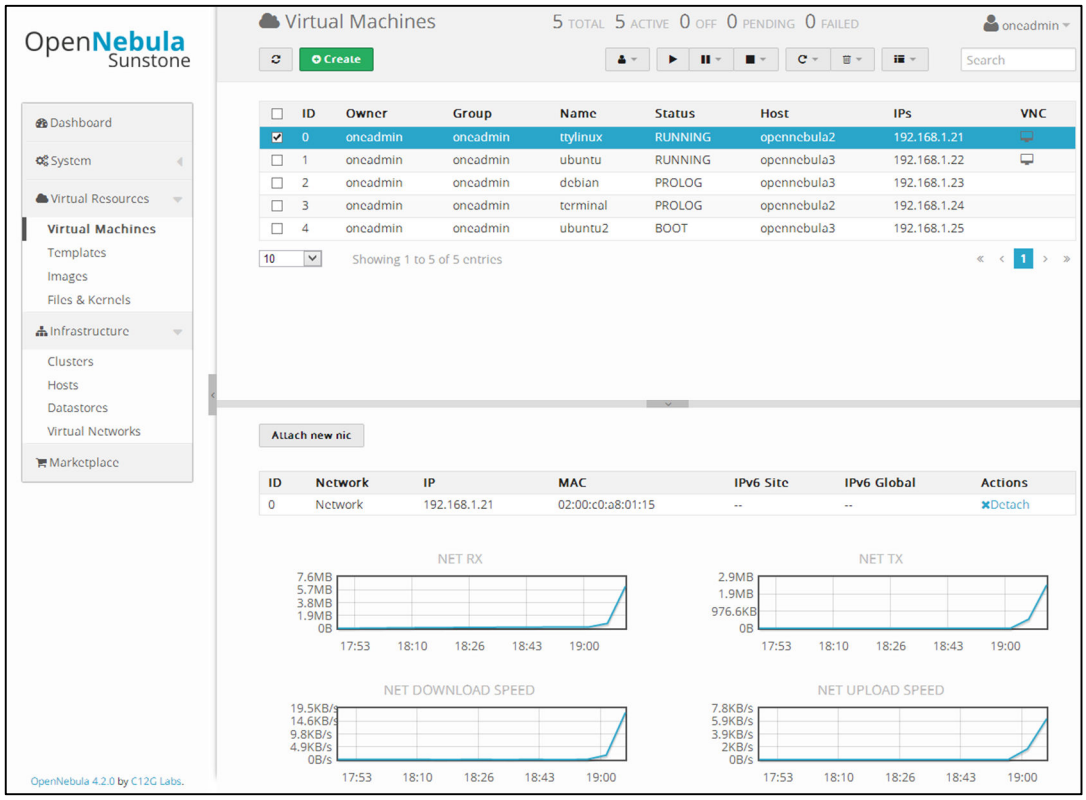
O OpenNebula apresenta informações úteis em relação ao monitoramento, tanto dos *hosts*, quanto das VM's, é possível verificar a taxa de transferência (*upload* e *download*), quantidade de memória utilizada e a percentagem (%) de processamento utilizada, tudo isso junto a interface Sunstone.

A Figura 28 mostra a tela inicial do OpenNebula, nela são apresentados gráficos que mostram o uso e o total de memória da nuvem e o processamento, podendo ser visualizado o total, o usado e o alocado. É mostrado também a taxa de *download* e *upload* por segundo de todas as máquinas virtuais.



Fonte: Hentges, Thomé, Griebler, 2013.
Figura 28: Gráficos da tela inicial

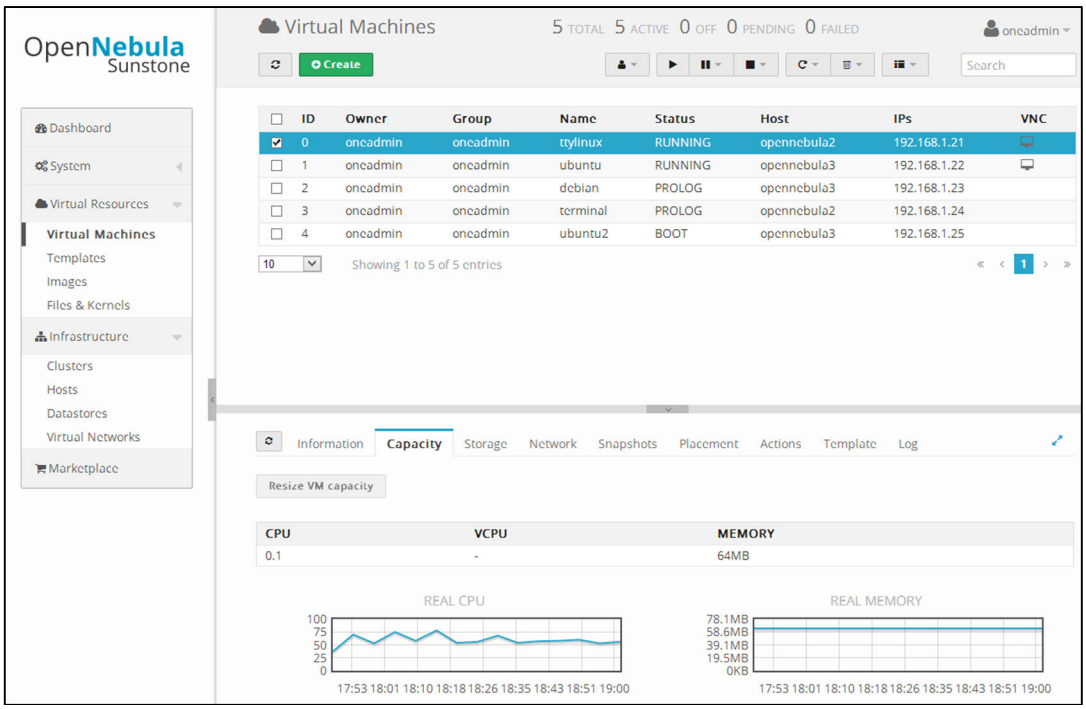
A Figura 29 mostra os gráficos que podem ser vistos em cada máquina virtual individualmente, mostrando a quantidade de *download* e *upload*, além da taxa de *download* e *upload* por segundo.



Fonte: Hentges, Thomé, Griebler, 2013.

Figura 29: Gráficos de rede das máquinas virtuais

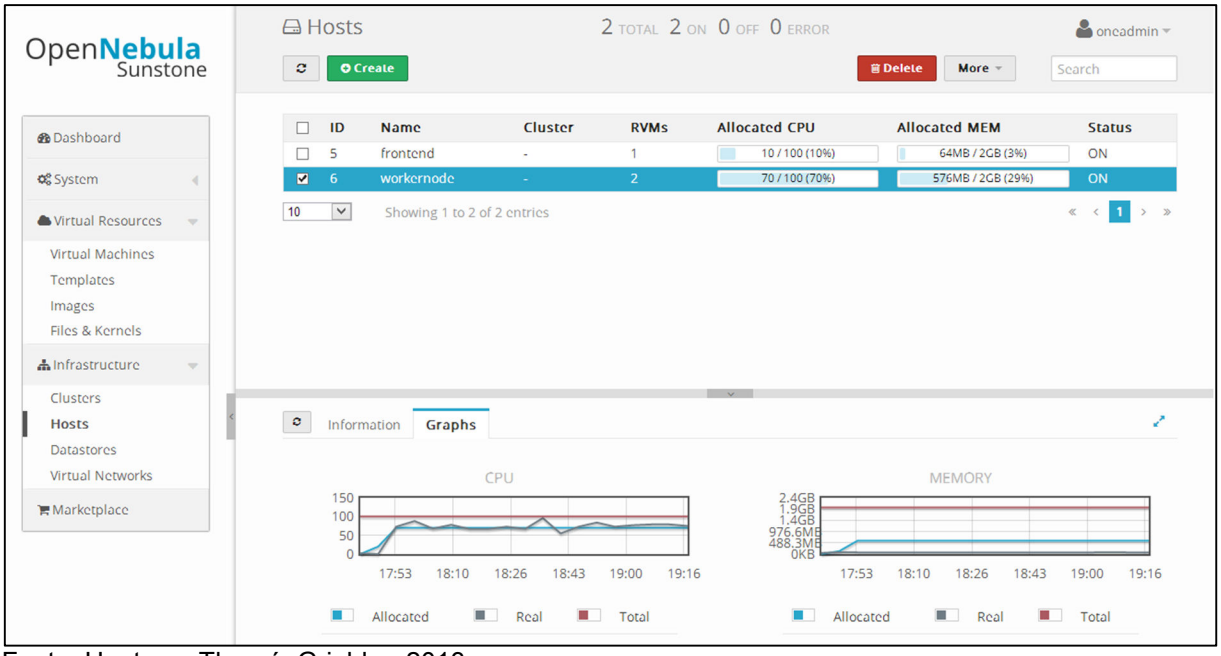
A figura 30 mostra os gráficos que podem ser vistos em cada máquina virtual, nele podem ser vistos o uso de memória e o processamento.



Fonte: Hentges, Thomé, Griebler, 2013.

Figura 30: Gráficos de memória e processamento da máquina virtual

A Figura 31 mostra os gráficos que podem ser vistos para cada *host* que compõem a nuvem. Nestes gráficos pode ser verificado o uso de memória, o limite de memória e o quanto há de memória alocada. O gráfico que exibe o processamento também mostra o limite de processamento, o uso e o quanto há alocado.



Fonte: Hentges, Thomé, Griebler, 2013.

Figura 31: Gráficos de memória e processamento dos nós

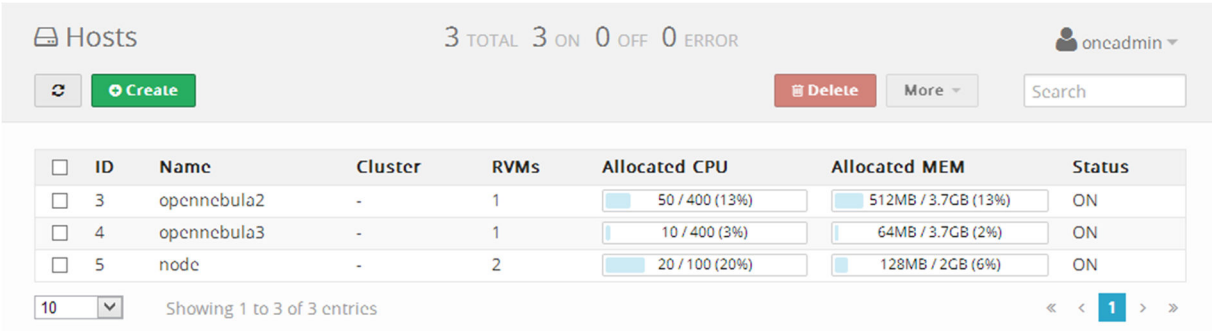
3.5.1.3 Mecanismo de Tolerância a Falhas

Para o teste de tolerância a falhas no OpenNebula haviam algumas máquinas virtuais que estavam em funcionamento nos nós. Então, foi realizado o desligamento forçado de uma VM, simulando uma possível falha naquele nó, após isso, o *frontend* que monitora os nós a cada 60s para saber seu estado, recebeu o estado de erro do nó em que foi submetido o desligamento. Depois disso foi ativada uma *trigger* onde é executado um *script* que resubmete as máquinas virtuais que estavam executando naquele nó, fazendo com que as mesmas executassem em algum outro nó que estava funcionando (está exemplificado em imagens a seguir).

Um fator problema neste caso é que o responsável por esta tarefa de resubmeter as VM's é o *frontend,* se o mesmo sofrer problemas, toda a nuvem

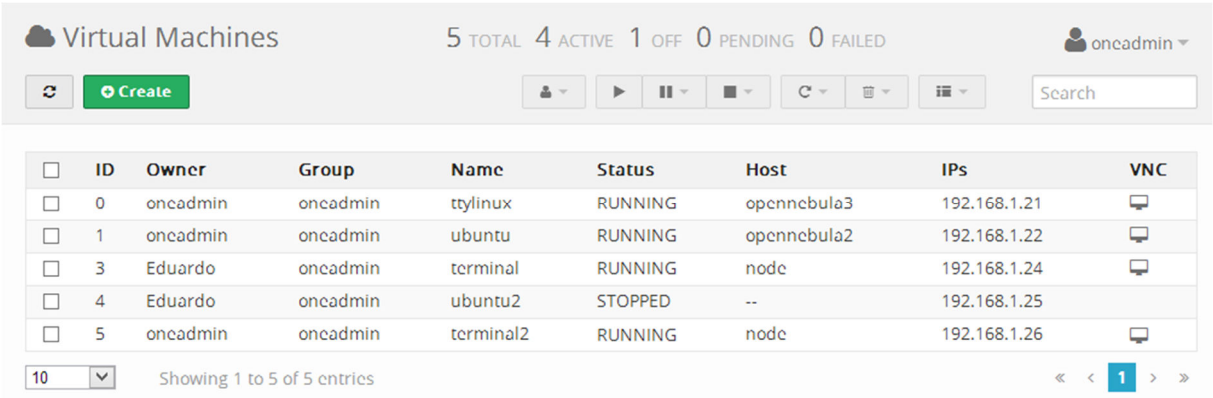
acaba sendo comprometida. Por isso, é indicado que se tenha redundância do computador que exerce esta função para que se possa manter ininterrupto o serviço, tendo um servidor *frontend* de *backup*.

A figura 32 mostra todos os nós em funcionamento, neste momento todos eles estão ligados e em execução.



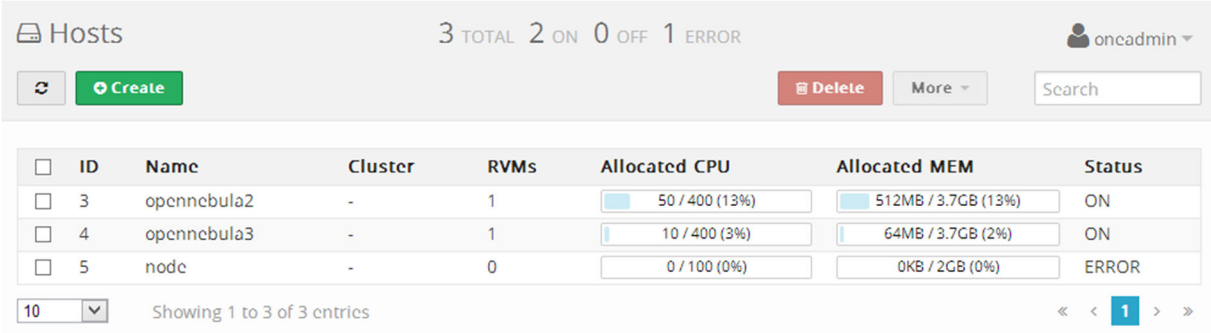
Fonte: Hentges, Thomé, Griebler, 2013.
Figura 32: Todos os nós funcionando

A figura 33 mostra as máquinas virtuais em execução, sendo duas delas no *host* chamado “node”, que será desligado.



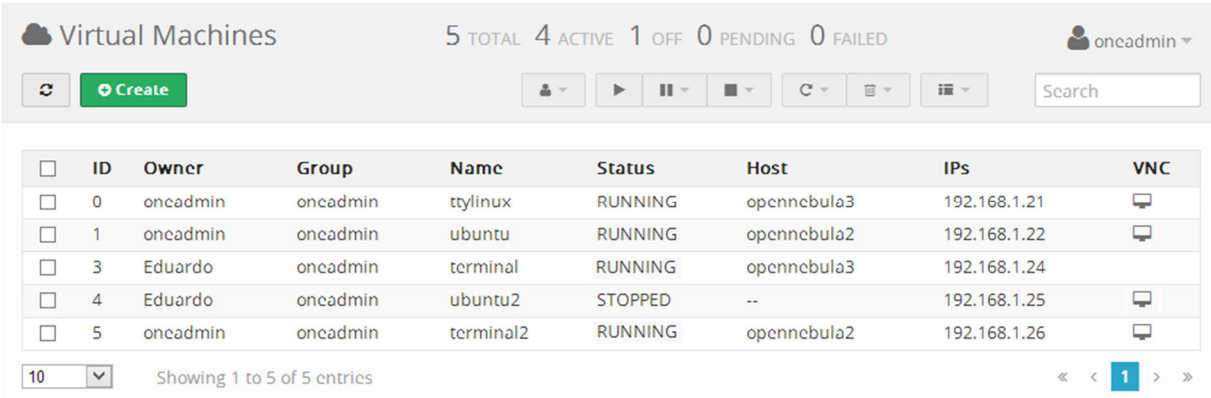
Fonte: Hentges, Thomé, Griebler, 2013.
Figura 33: Máquinas virtuais em funcionamento

A figura 34 mostra então o nó identificado por “node” foi desligado de forma abrupta, aparecendo então à mensagem de erro, ao qual deu inicio a execução da *trigger*.



Fonte: Hentges, Thomé, Griebler, 2013.
Figura 34: Identificado o erro na máquina virtual

A figura 35 mostra que após a recuperação, as máquinas virtuais estão novamente em execução, porém em outros *hosts*.



Fonte: Hentges, Thomé, Griebler, 2013.
Figura 35: Máquinas virtuais executando em outro nó

3.5.1.4 Sistema de Gerenciamento de Energia

O gerenciamento de energia do OpenNebula não pode ser implementado pois o material referente as ferramentas responsáveis pelo gerenciamento de energia são escassos, se limitando apenas ao material oferecido pelos desenvolvedores das ferramentas, e mesmo assim eram de pouco ajuda, devido ao baixo nível de detalhes. Foram encontradas duas citações de extensões que funcionam junto ao OpenNebula. A primeira é o Clues (*Cluster Energy Saving*) e a segunda o *Green Cloud Scheduler*.

Com a leitura sobre o funcionamento destas ferramentas pode se obter a noção de como funcionariam, ambas utilizariam o mesmo método para fazer o

desligamento e ligamento dos nós, utilizariam *shutdown* para desligar e o WakeOnLan para ligar as máquinas.

O Clues realiza o monitoramento dos *status* dos computadores, e só realizaria desligamento caso algum estivesse sem uso e o ligamento de nós caso estivesse perto do limite o uso de algum nó. Além disso, ele apresenta um relatório que calcula o consumo de cada nó e mostra em porcentagem o quanto de tempo cada nó esteve em cada estado (Desligado, Ligado Sem uso, Uso parcial, Uso em carga máxima e Falha).

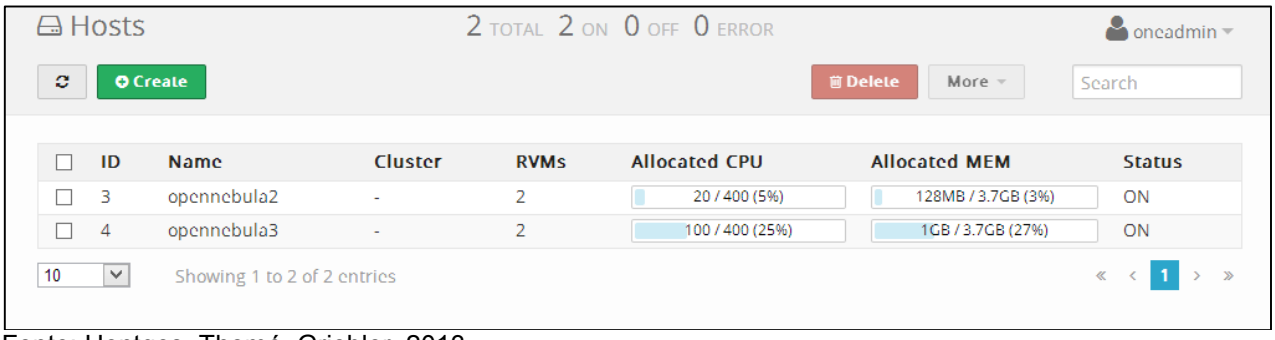
O *Green Cloud Scheduler* substitui o *Scheduler* do OpenNebula e a partir disso controla aonde cada máquina virtual deve executar. Para isso, ele obtém informações de uma ferramenta chamada Cpufrequtils, que mede o uso de processamento de cada computador e utiliza ele para saber o que fazer com cada nó. Se um estiver com pouco uso e o outro estiver com 50% livre, ele irá migrar a VM do nó com pouco uso para o outro com metade do uso livre, para então poder desligar aquele nó que agora está sem uso.

3.5.1.5 Mecanismo de Balanceamento de Carga

Em relação ao balanceamento de carga do OpenNebula, pode observar-se que durante a criação de máquinas virtuais, o *frontend*, responsável pelo controle da nuvem, divide estas máquinas virtuais por todos os *hosts* que estão ativos na nuvem. O mesmo acontece ao desligar alguma VM e religar em seguida, as máquinas virtuais são divididas de forma igual entre os nós responsáveis pela virtualização destas VM’s.

Como fator contra, pode ser citado que ela só faz isso enquanto estiver iniciando uma nova máquina virtual, pois por exemplo de houverem três nós onde um deles (node1) executam três máquinas virtuais e o outros dois nós (node2 e node3) execute somente duas, ao desligar alguma máquina virtual no node2 ou no node3, este nó só terá uma única máquina virtual para executar, ele não fará com que alguma outra máquina virtual que esteja executando no node1 passe a executar no nó que está apenas executando uma máquina virtual.

Na Figura 36, é exemplificada a divisão das máquinas virtuais entre os nós na nuvem, onde pode ser observado um número igual de VM's em cada um dos *hosts,* porém é perceptível que ele apenas realiza a divisão tendo como base o número de máquinas virtuais, e não em relação aos recursos utilizados pelas máquinas virtuais, não sendo encontrado nenhum meio de parametrizar para que se possa definir como as máquinas seriam divididas.

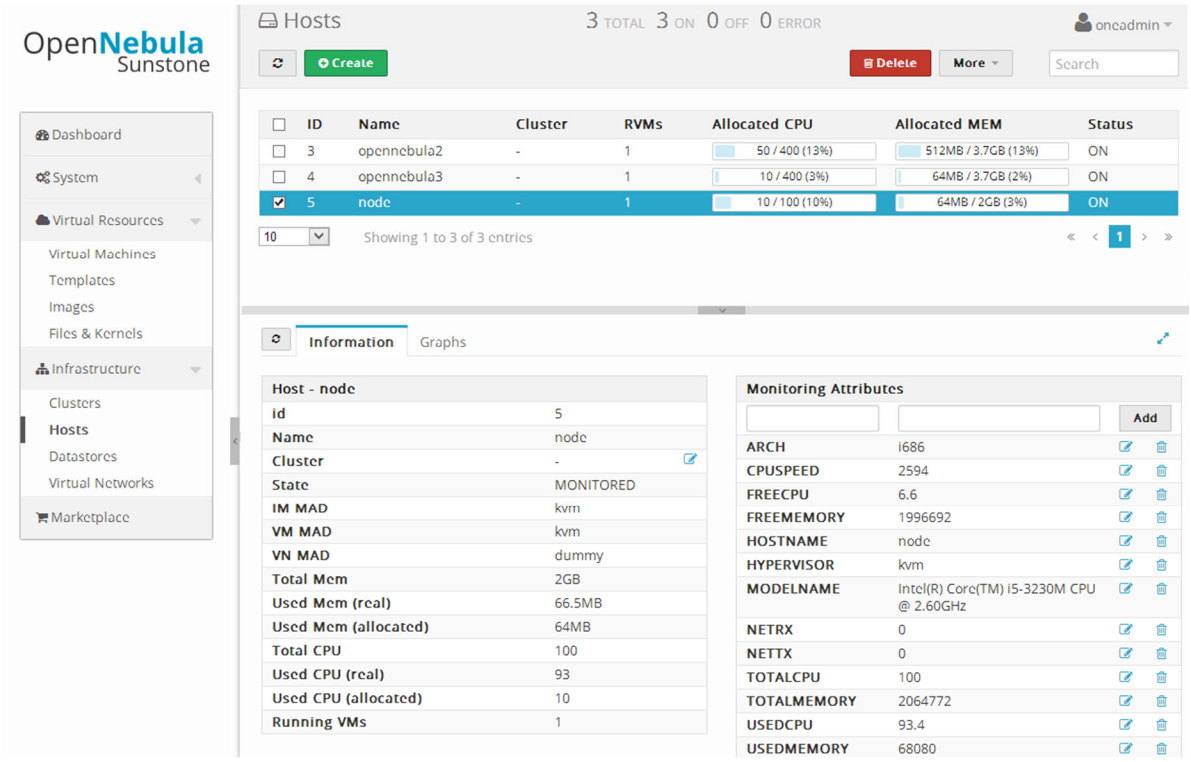


Fonte: Hentges, Thomé, Griebler, 2013.
Figura 36: Balanceamento de carga

3.5.1.6 Escalabilidade

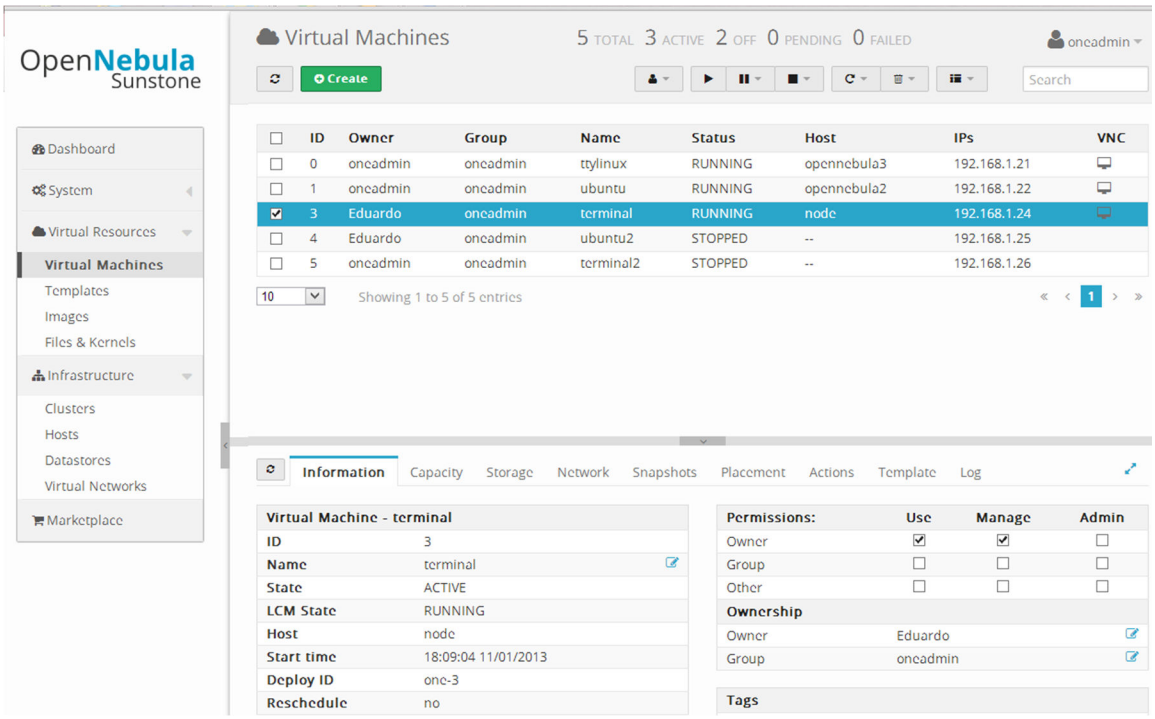
Neste teste, foi configurado mais um computador para que exercesse a função de nó na nuvem (criação do usuário oneadmin instalando os pacotes responsáveis pela virtualização, configurar SSH sem senha, dar permissão para usuário oneadmin para os *softwares* de virtualização, alterar a configuração do Qemu-KVM e Libvirt, montagem de pasta compartilhada). Após isso, com a nuvem em funcionamento o mesmo foi adicionado, onde pode exercer a função de nó, executando máquinas virtuais, sem interromper as tarefas que estavam executando no momento.

Após a configuração do nó extra, e o mesmo ser adicionado, pode-se ver na figura 37 o nó com estado ON, portanto foi reconhecido pela nuvem e está sendo monitorado pela mesma.



Fonte: Hentges, Thomé, Griebler, 2013.
Figura 37: Escalabilidade OpenNebula

A figura 38 mostra que o nó esta cumprindo com sua tarefa, sendo que o mesmo esta executando uma máquinas virtual.

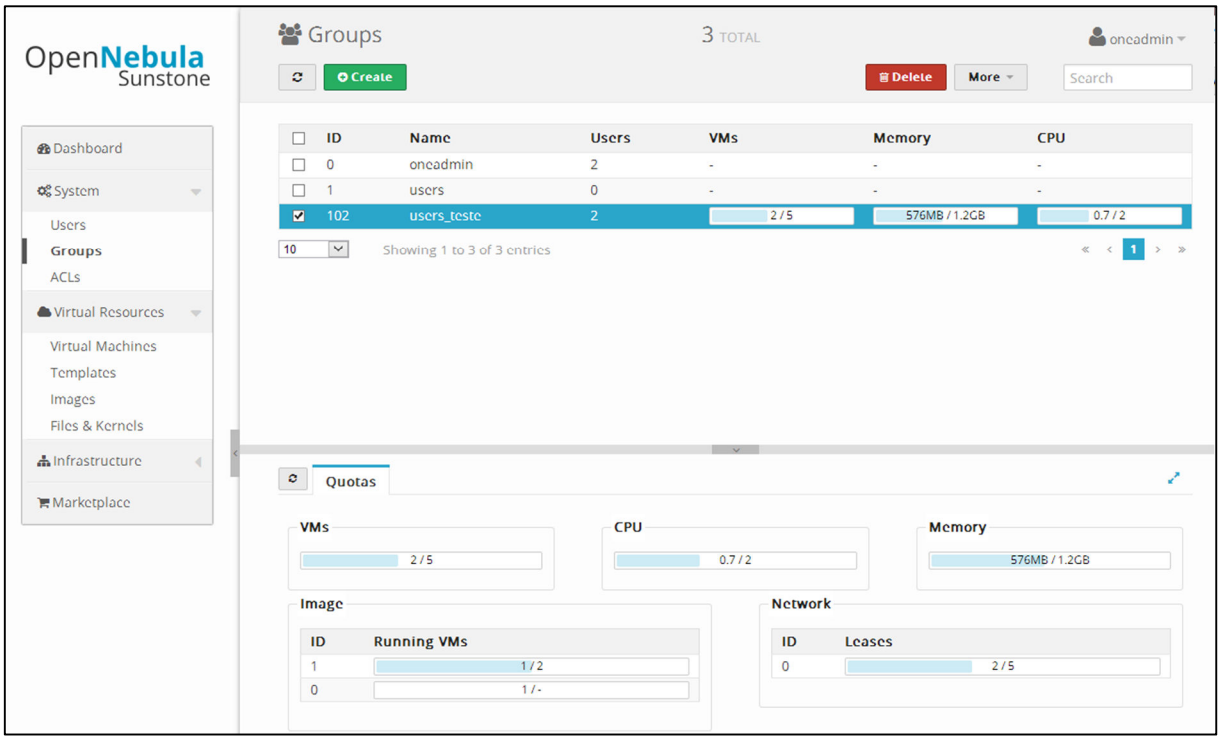


Fonte: Hentges, Thomé, Griebler, 2013.
Figura 38: Máquina virtual em execução no novo nó

3.5.1.7 Sistema de Segurança

O método de autenticação disponível para o OpenNebula se limita ao fornecido pela própria ferramenta, mas mesmo assim, ele consegue fornecer um bom controle sobre os usuários. É possível criar grupos e usuários. Além disso, o OpenNebula fornece a opção de criar ACL's (Listas de controle de acesso), onde podem ser criadas regras para o que cada grupo de usuários ou um usuário em específico tem acesso, e o que ele pode fazer (usar, criar, gerenciar). Assim como é possível limitar o número de recursos e de VM´s que cada grupo pode utilizar e isso pode ser atribuído também diretamente a cada usuário.

A Figura 39 mostra os grupos de usuários. Nesta tela é possível criar e apagar grupos, além de definir quotas para os grupos. Pode-se observar a quantidade de máquinas virtuais que estão em execução por grupo, além do uso de memória e de processamento.

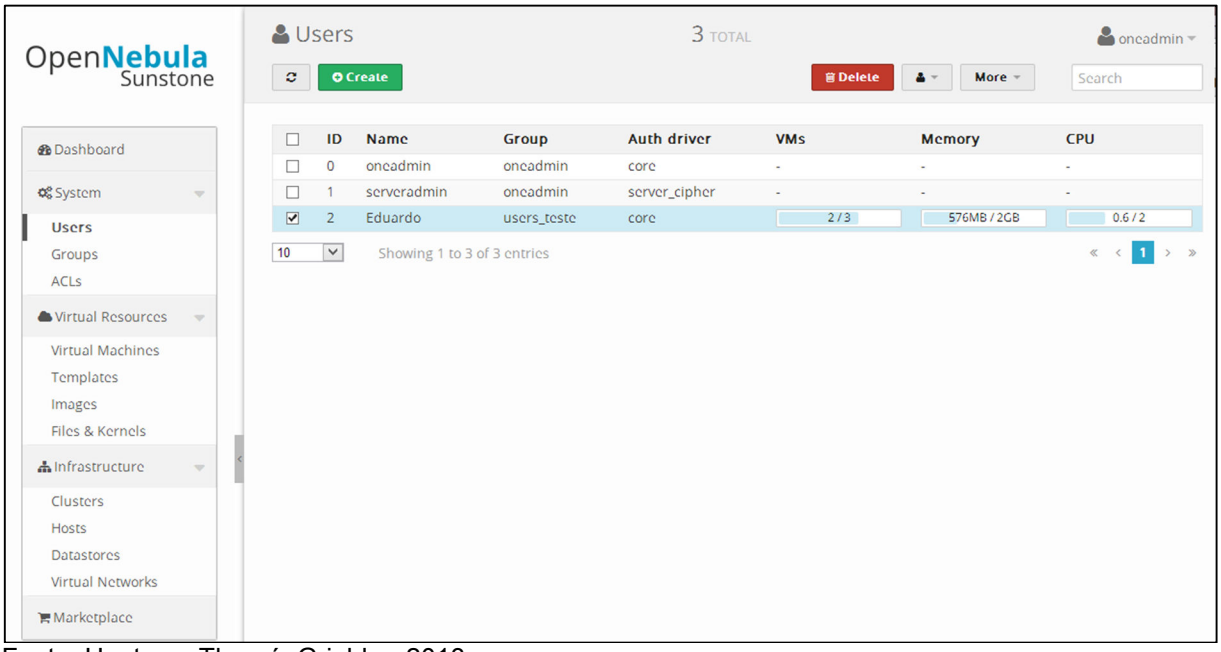


Fonte: Hentges, Thomé, Griebler, 2013.

Figura 39: Grupos de usuários

A Figura 40 mostra a tela de usuários, nela é possível criar e apagar usuários, definir um grupo para um usuário, além de definir as quotas disponíveis

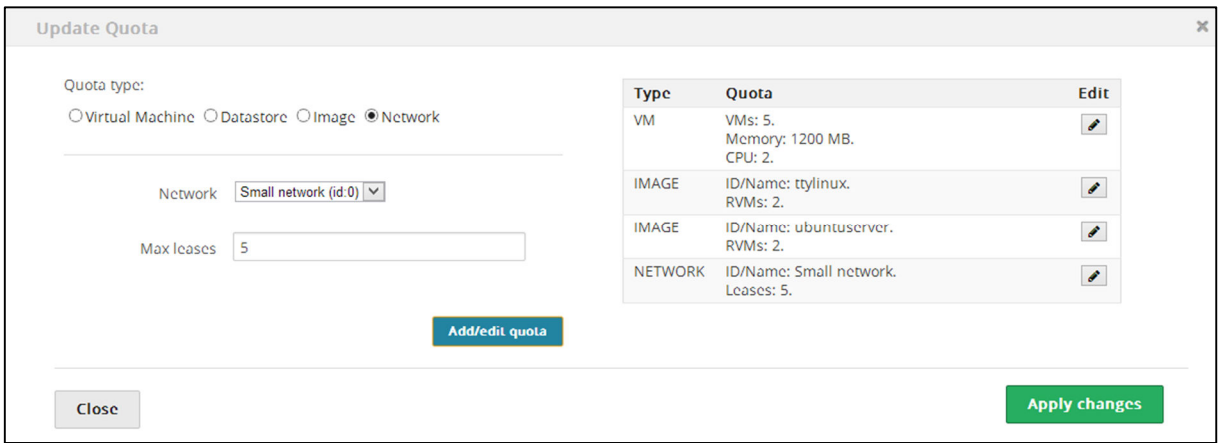
para cada usuário, nela também pode ser visto a quantidade de máquinas virtuais em execução, o uso de memória e o uso de processamento.



Fonte: Hentges, Thomé, Griebler, 2013.

Figura 40: Usuários

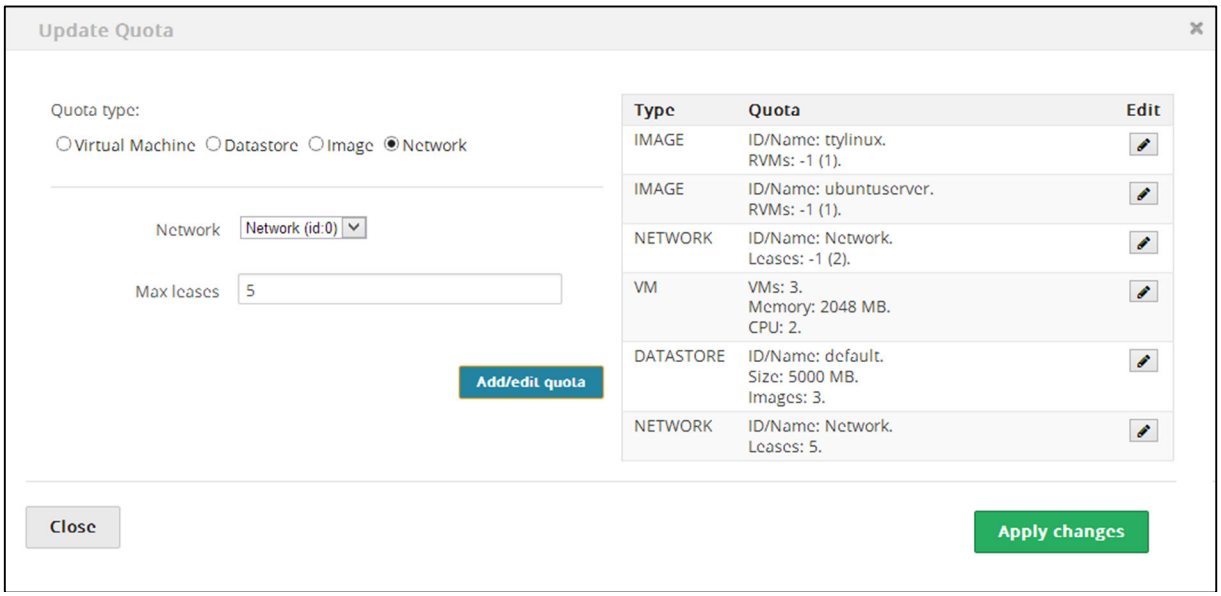
A Figura 41 mostra a criação de uma quota para o grupo, as quotas são responsáveis por definir ao que cada grupo tem acesso, e quanto ele pode utilizar dos recursos. Pode ser definido o número de máquinas virtuais, definir se tem acesso ao datastore, onde ele pode adicionar imagens ou arquivos, quantos GB ele pode armazenar, à quais imagens ele tem acesso e a rede que um grupo pode utilizar, definindo também uma quantidade de IP's que ele poderá utilizar.



Fonte: Hentges, Thomé, Griebler, 2013.

Figura 41: Quotas de grupos

A Figura 42 mostra a criação de uma quota para um usuário, nela podem ser atribuídos assim como as quotas para o grupo o número de máquinas virtuais, definir se tem acesso ao *datastore*, onde ele pode adicionar imagens ou arquivos, e a quantos GB ele pode armazenar, a quais imagens ele tem acesso e a rede que um usuário pode utilizar, definindo também uma quantidade de IP's que ele poderá utilizar.

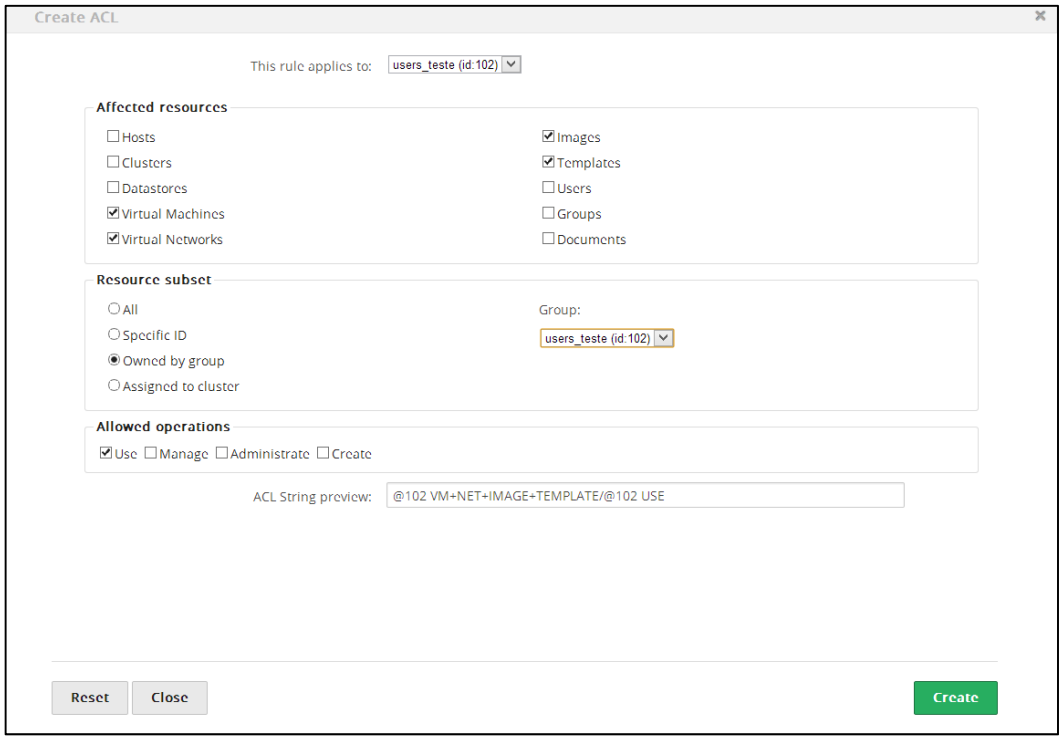


Fonte: Hentges, Thomé, Griebler, 2013.

Figura 42: Quotas de usuários

A Figura 43 mostra a criação de uma ACL, as ACL's são responsáveis por criar regras para um grupo ou usuários, nela pode-se definir ao que os usuários terão acesso, e o que ele será capaz de realizar, como usar, administrar, gerenciar e criar.

As ACL's apresentam uma grande quantidade de opções em relação a adição de permissões aos usuários, sendo possível realizar um excelente controle sobre ao que cada usuário tem acesso, dependendo das necessidades do administrador.

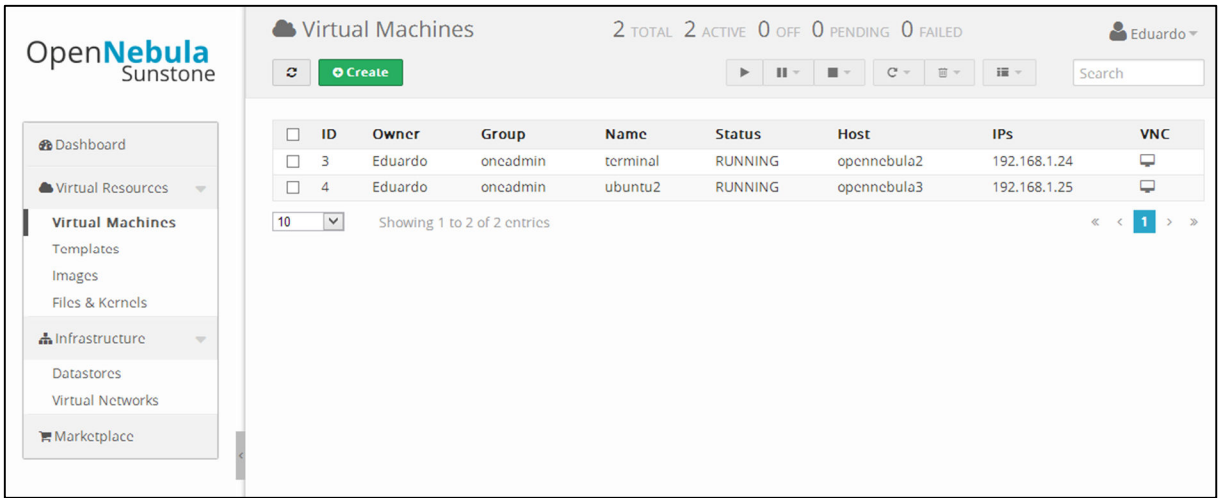


Fonte: Hentges, Thomé, Griebler, 2013.
Figura 43: Criação de ACL

A Figura 44 mostra a lista de ACL's, nesta página é possível criar e apagar as ACL's, que são responsáveis pelo controle de acesso dos usuários.

Fonte: Hentges, Thomé, Griebler, 2013.
Figura 44: Lista de ACL

A Figura 45 mostra o acesso de um usuário, onde percebe-se que no painel a esquerda é reduzido o acesso do usuário. Além disso, é possível observar que o usuário só tem acesso as máquinas virtuais ao qual ele é proprietário.



Fonte: Hentges, Thomé, Griebler, 2013.

Figura 45: Acesso do usuário

3.5.1.8 Opções de Rede

Foi configurado para que os nós dispusessem de uma interface *bridge,* para que assim as máquinas virtuais fossem configuradas para se comunicarem com esta interface, desta maneira fornecendo acesso à rede as mesmas. Há a possibilidade de utilizar-se de VLAN, para isolar *hosts* em redes separadas, para isso seria preciso de um *switch* que pudesse criar *trunks* VLAN. E também é possível a utilização do OpenvSwitch, que se trata de um *switch* virtual que tem a responsabilidade de encaminhar o tráfego em uma rede de máquinas virtuais, ele também possibilita criar VLAN para que se possa isolar o tráfego.

3.5.1.9 Opções de Armazenamento

Nesta nuvem foi configurado para que o sistema de armazenamento fosse compartilhado do *frontend* para os nós utilizando-se do NFS, sendo realizado de maneira fácil, precisando apenas da instalação do NFS-*kernel-server* e necessitando apenas configurar o diretório de compartilhamento no *frontend* e a origem do compartilhamento e o ponto de montagem nos nós. Mas também possibilita o uso de LVM e iSCSI, ambos podem ser utilizados através de partições exclusivas para o armazenamento das imagens e das VM's e também podem ser utilizados computadores somente para esta função, além disso, podem realizar a transmissão dos dados utilizando-se de SSH.

3.5.1.10 Sistema de Integração

O OpenNebula permite a integração de sua nuvem com o ambiente em nuvem da Amazon EC2, para que ambos se comuniquem é preciso que ambos tenham chaves e certificados que permitem a comunicação. E, que sejam instalados os *softwares* que farão a comunicação e também é necessária a instalação das dependências e da configuração dos arquivos referentes ao EC2. Só assim é possível se conectar a sua conta da Amazon EC2, e o ambiente em nuvem do EC2 pode ser adicionado como um nó, aparecendo na lista de *hosts* para virtualização. Após isso, é possível executar máquinas virtuais na nuvem da Amazon como se fosse no próprio ambiente da nuvem. Isto não foi implementado devido a nuvem da Amazon ser paga e, além disso, para que funcione de maneira correta é necessário uma *Internet* com boa velocidade de *upload*, devido a necessidade de transmissão das imagens para a nuvem da Amazon.

3.5.1.11 Opções de Virtualização

Para realizar a virtualização foi utilizado o virtualizador KVM, sendo ele uma ferramenta muito utilizada. Para que os computadores pudessem executar as máquinas virtuais, foi preciso ativar o suporte a virtualização (Intel-Vt) nos dois nós que tem a tarefa de executar as VM´s.

Para realizar a implementação do KVM foi necessário apenas à instalação dos pacotes qemu-kvm, libvirt-bin e ubuntu-vm-builder. E realizar pequenas alterações nos arquivos de configuração do KVM e do Libvirt para que possibilitassem acesso do usuário oneadmin a fim de monitorar o *host* e para permitir a criação e execução de máquinas virtuais (mais detalhes pode ser visto no Apêndice A.5.3).

Pode-se utilizar também o Xen, onde ao invés de instalar o KVM nos nós, deve ser instalado o Xend *daemon.* No *frontend* seria preciso apenas configurar para que o OpenNebula soubesse que deveria utilizar o Xen, precisando apenas descomentar no arquivo oned.conf as linhas referentes a ele e comentar as que se referem ao KVM.

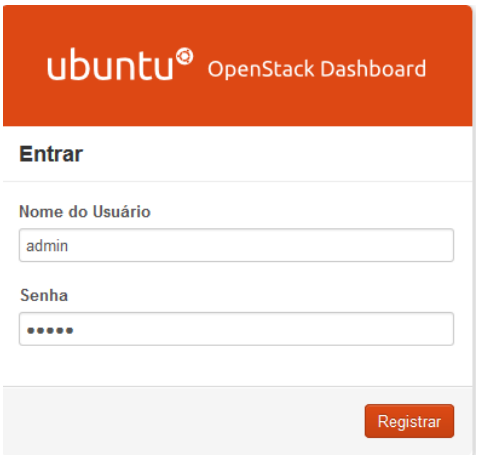
O OpenNebula também permite utilizar o VMware, onde é preciso instalar *softwares* no *frontend,* além da dos *drivers*. Após isso, é feita a configuração do arquivo oned.conf para que utilize o VMware, sendo que nos nós apenas é preciso instalar o *software* responsável pela virtualização.

3.5.2 OpenStack

O OpenStack é uma ferramenta *open source* podendo ser utilizada para a configuração e gerenciamento da infraestrutura de computação e o armazenamento em nuvem. Nesta seção são descritos os testes feitos em ambiente com estações de trabalho utilizando desta ferramenta.

3.5.2.1 Interface com Usuário

O acesso por interface *Web* no OpenStack requer previa instalação do componente Horizon, responsável por permitir esse acesso e montar a interface gráfica para o usuário. Utiliza-se do IP da máquina *frontend* (*Controller Node*) ou se configurado um domínio para acessar a interface, também é necessário o uso de usuário e senha para autenticação. (Figura 46)



Fonte: Hentges, Thomé, Griebler, 2013.
Figura 46: Tela *login* Horizon

Quanto à utilização da interface *Web*, notou-se que a interface solicita do usuário um conhecimento prévio de seus itens (encontrados no lado esquerdo da tela), pois não é muito intuitiva. Ela é formada pelos itens (Figura 47):

Instâncias**:** as instâncias são as máquinas virtuais individuais que estão rodando em nós de computação físicos, diversas instâncias podem ser executadas a partir da mesma imagem sendo utilizada uma cópia dessa imagem, para cada instância criada. O componente do OpenStack que administra as instâncias é chamado Nova.

Volumes**:** é gerido pelo serviço nova-volume, que permite que se adicione maior armazenamento para as instâncias.

Na aba **Serviços,** são listados os componentes do OpenStack, juntamente com o serviços que eles prestam e se estão habilitados.

Os *Flavors* são os modelos de *hardware* virtual disponível, eles definem, por exemplo, o tamanho da memória RAM e do disco. Os *flavors* só podem ser configurados pelo administrador.

Imagens**:** no item imagens se encontram as imagens de sistemas operacionais que são modelos para máquinas virtuais. O componente do OpenStack que gerencia e armazena essas imagens é chamado *Glance*.

Os Projetos (*Tenants*) são ambientes de recursos isolados (VLAN, volumes, instâncias, imagens, chaves e usuários) formando a estrutura principal do Nova.

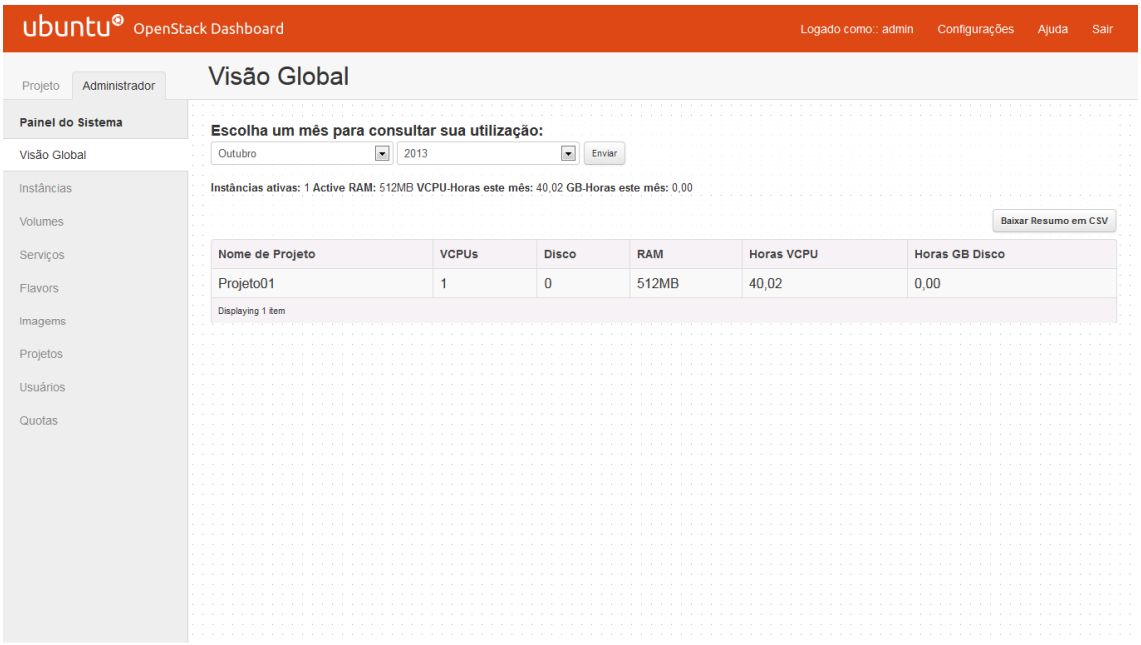
Na aba Usuários encontram-se todos os usuários cadastrados, ao criar um novo usuário se específica o projeto ao qual ele ira pertencer e a função desse usuário no projeto. Os usuários são controlados e gerenciados pelo componente Keystone.

Na aba Cotas o OpenStack fornece um número de cotas aplicadas a nível de projeto. As cotas podem especificar, por exemplo, o número de núcleos permitidos por projeto, o número de IPs flutuantes por projeto ou até mesmo o número de arquivos inseridos permitidos.

Fonte: Hentges, Thomé, Griebler, 2013.
Figura 47: Itens do painel.

Através da interface *Web*, quando logado com administrador (Figura 48) é possível ter acesso a todos os itens do painel, ter uma visão geral do projeto e seu tempo em execução da CPU virtual (vCPU). Também é possível verificar as instâncias, a que IP pertencem, o tamanho da *flavor* designada e se está em condição ativa.

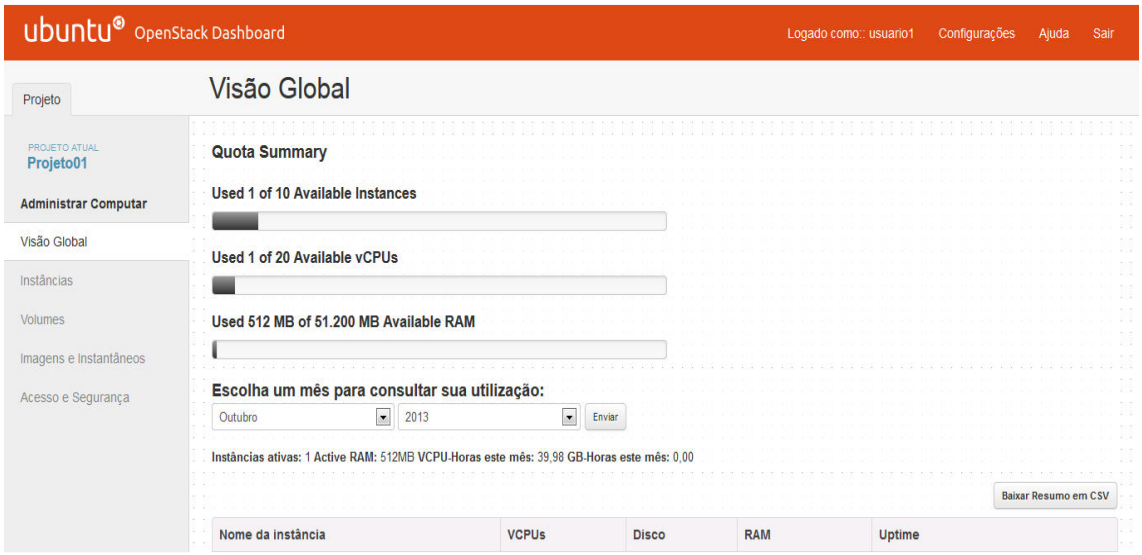
No painel também se pode verificar os volumes, serviços disponíveis, as *flavors* existentes, as imagens disponíveis para a criação de máquinas virtuais, além disso, verificar os projetos existentes e os usuários, e observar as cotas existentes. O administrador também possui acesso às mesmas informações sobre um projeto tal como um usuário comum possuem.



Fonte: Hentges, Thomé, Griebler, 2013.
Figura 48: *Login* como administrador.

Porém quando logado ao painel como usuário comum (Figura 49) apenas terá uma visão geral sobre a cota no sumário de cotas onde se verifica o uso de instâncias disponíveis, vCPUs e memória disponíveis. Pode-se também verificar a quanto tempo a sua instância está ativa.

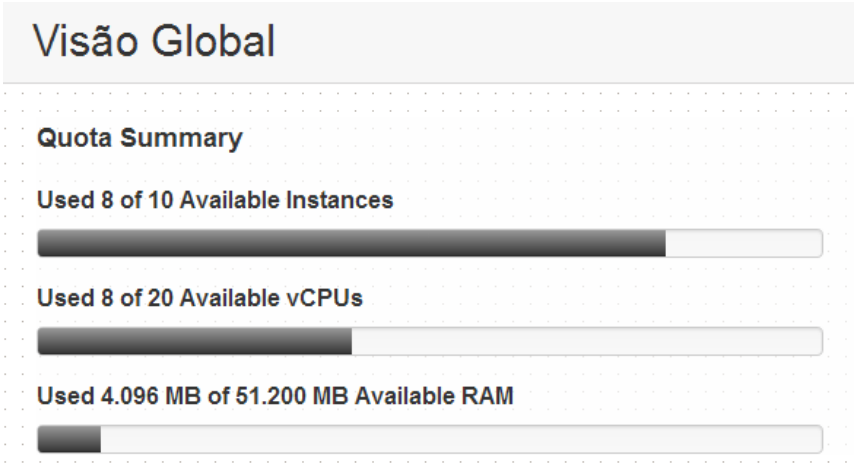
O usuário pode também observar maiores informações sobre a sua instância e lançar novas instâncias. Criar volumes e imagens, utilizar as imagens existentes em novas instâncias. Há a possibilidade de criar *snapshot* de instâncias em execução o que permite criar uma nova imagem baseada no estado atual do disco. Além disso, o usuário tem uma aba de acesso e segurança onde pode ser alocado IP para o projeto, criar grupo de segurança ou criar um par de chaves de segurança especifica.



Fonte: Hentges, Thomé, Griebler, 2013.
Figura 49: *Login* usuário comum.

3.5.2.2 Sistema de Monitoramento

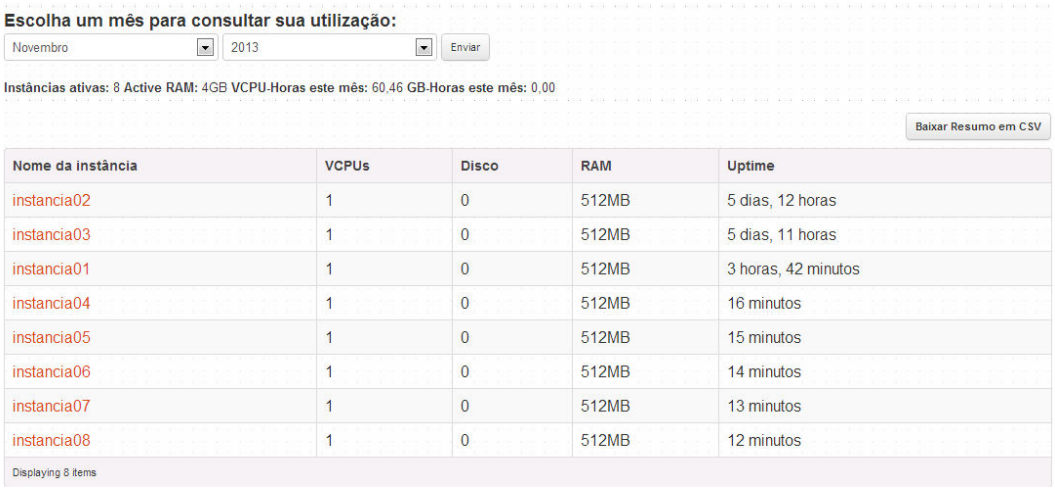
Quanto ao monitoramento, o *dashboard* do OpenStack, quando acessado como usuário demonstra a quantidade de instâncias e de vCPUs (CPUs virtuais) utilizadas e também o quanto de memória está em uso (Figura 50).



Fonte: Hentges, Thomé, Griebler, 2013.
Figura 50: Tela dados do monitoramento.

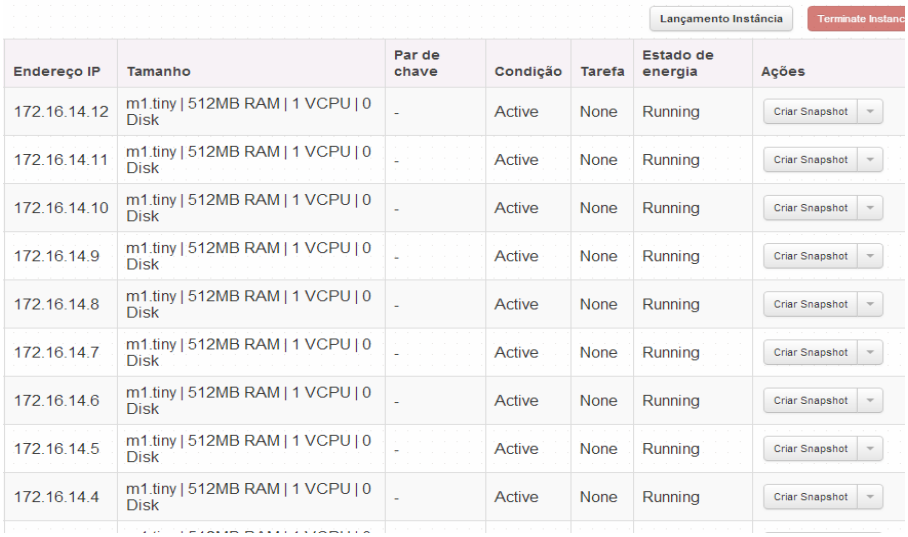
Ainda é possível visualizar informações consultando por mês, caso desejar, nome das instâncias existentes, quantidade de vCPUs de cada uma delas, disco, quantidade de memória alocada e o tempo que está em execução. Assim como,

quantas estão em execução e é possível também fazer *download* de um arquivo cvs que contém esses dados (Figura 51).



Fonte: Hentges, Thomé, Griebler, 2013.
Figura 51: Tela dados do monitoramento(2).

O painel também apresenta no item Instâncias, o endereço IP da instância, o tamanho que foi configurado, demonstra se a condição está ativa, inativa ou com erro. Também é possível ver o estado de energia para ver se a instância está rodando (Figura 52).



Fonte: Hentges, Thomé, Griebler, 2013.
Figura 52:Monitoramento das instâncias.

3.5.2.3 Mecanismo de Tolerância a Falhas

Na nuvem implantada não há possibilidade de tolerância a falhas, pois as instâncias criadas ficam armazenadas nos *Compute Node* e não no *Controller Node*, assim como o KVM apenas foi implementado nos *Compute Node*, sendo assim quando ocorrer alguma falha em algum dos *Compute Node* essas instâncias não irão ser migradas para outro *Compute Node*. O componente que permite centralizar o armazenamento das instâncias se configurado é o *Nova-volume.* Assim, em caso de falhas em um dos *Compute Nodes* é possível à migração da instância para outro *Node*.

A tolerância a falhas não foi possível, pois não foi instalado o componente nova-volume, para que pudesse ser instalado esse componente seria necessário mudanças no esquema de compartilhamento de rede para que se conecte na unidade de disco a ser criada, assim ele informa para os nós onde está o repositório. Porém o ambiente de rede utilizado não suportaria esse tipo de configuração, o ambiente para testes deveria ser todo modificado para assim ser instalada e configurada a tolerância a falhas.

3.5.2.4 Sistema de Gerenciamento de Energia

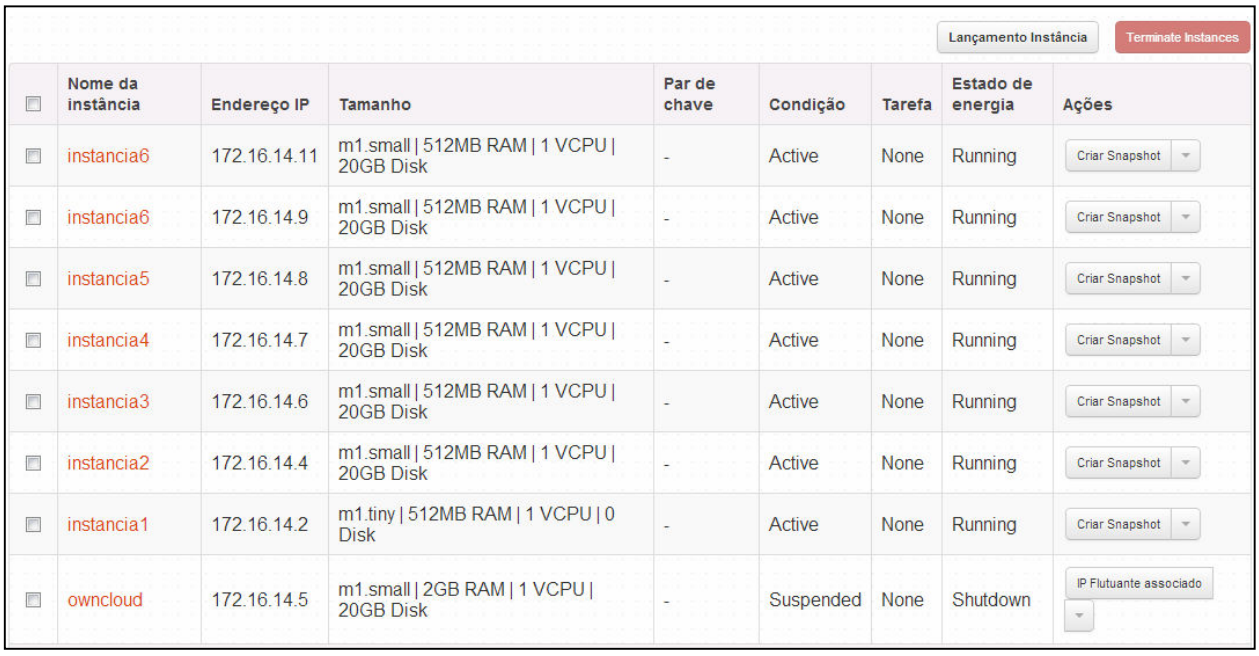
A ferramenta OpenStack permite o gerenciamento de energia através de extensões que apenas funcionam em processadores Intel Xeon (Intel, 2013). Levando em conta essa informação para o caso da nuvem implantada, está não permite gerenciamento de energia, pois ela possui um processador Intel Core I5.

3.5.2.5 Mecanismo de Balanceamento de Carga

O OpenStack pode realizar balanceamento de carga de forma simples com o auxilio do componente nova-*scheduler* que é utilizado como um padrão em arquiteturas não compartilhadas, pois implementa um programa que tenta encontrar um *host* que esteja menos sobrecarregado implementando um algoritmo de propagação.

Basta instalar o nova-*scheduler* e configurar no arquivo nova.conf a linha *scheduler driver*=nova.*scheduler.simple.SimpleScheduler,* essa linha descreve um dos algoritmos básicos que o *Scheduler* utiliza como método *Simple*, com ela as instâncias podem passar a ser executadas nos *hosts* que possuírem menor carga no momento em que a instância for criada.

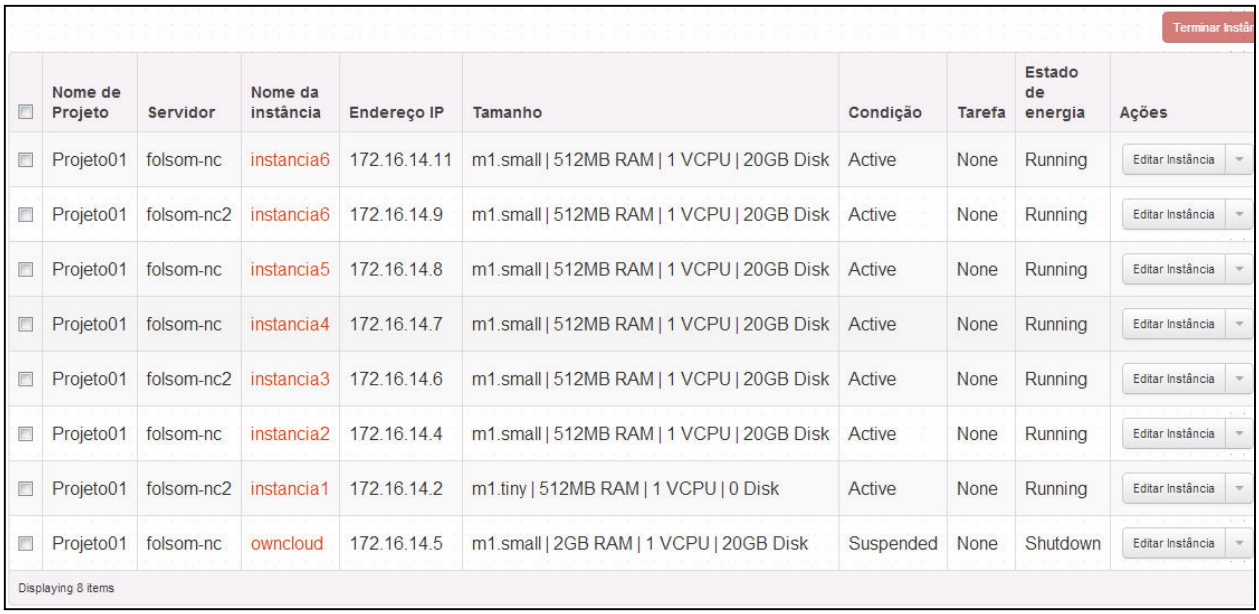
Para verificar o balanceamento de carga é preciso criar as instâncias que desejar, elas podem ser criadas no Projeto quando acessar o painel como Usuário no item Instâncias ou quando acessar o painel como Administrador na aba Projeto, item Instâncias. A Figura 53 demonstra as instâncias criadas.



Fonte: Hentges, Thomé, Griebler, 2013.

Figura 53: Instâncias criadas.

E acessando o painel como Administrador no item Instâncias é possivel verificar em qual *Compute Node* está rodando cada instância (Figura 54), está distribuição acontece pelo uso do componte nova-*scheduler*.



Nome de Projeto	Servidor	Nome da instância	Endereço IP	Tamanho	Condição	Tarefa	Estado de energia	Ações
Projeto01	folsom-nc	instancia6	172.16.14.11	m1.small | 512MB RAM | 1 VCPU | 20GB Disk	Active	None	Running	Editar Instância
Projeto01	folsom-nc2	instancia6	172.16.14.9	m1.small | 512MB RAM | 1 VCPU | 20GB Disk	Active	None	Running	Editar Instância
Projeto01	folsom-nc	instancia5	172.16.14.8	m1.small | 512MB RAM | 1 VCPU | 20GB Disk	Active	None	Running	Editar Instância
Projeto01	folsom-nc	instancia4	172.16.14.7	m1.small | 512MB RAM | 1 VCPU | 20GB Disk	Active	None	Running	Editar Instância
Projeto01	folsom-nc2	instancia3	172.16.14.6	m1.small | 512MB RAM | 1 VCPU | 20GB Disk	Active	None	Running	Editar Instância
Projeto01	folsom-nc	instancia2	172.16.14.4	m1.small | 512MB RAM | 1 VCPU | 20GB Disk	Active	None	Running	Editar Instância
Projeto01	folsom-nc2	instancia1	172.16.14.2	m1.tiny | 512MB RAM | 1 VCPU | 0 Disk	Active	None	Running	Editar Instância
Projeto01	folsom-nc	owncloud	172.16.14.5	m1.small | 2GB RAM | 1 VCPU | 20GB Disk	Suspended	None	Shutdown	Editar Instância

Displaying 8 items

Fonte: Hentges, Thomé, Griebler, 2013.
Figura 54: Balanceamento de carga.

3.5.2.6 Escalabilidade

Para testar a escalabilidade em uma nuvem OpenStack foi utilizada uma terceira máquina como *Compute Node*, nesta máquina foram instalados os mesmos componentes do *Compute Node* descrito anteriormente: os componentes Nova *Compute* e Nova *Network*, o KVM para virtualização, o NTP, VLAN, Bridge-utils e Libvirt-bin (Seção B.2). Com a instalação e configuração destes componentes, basta configurar este nó também no *Controller Node* no arquivo */etc/hosts*, sincronizar com o banco de dados, que a nuvem aceita mais um nó sem problemas. Para verificar se realmente o nó está funcionando, com o comando *nova-manage service list* deverá listar os serviços rodando no *Controller Node* (cc), *Compute Node*(nc) e os serviços rodando no novo *Compute Node* (nc2), se o resultado do comando for igual ao demonstrado na Figura 55, o novo nó foi adicionado com sucesso.

```
Binary           Host                                 Zone             Status     State Updated_At
nova-cert        folsom-cc                            nova             enabled    :-)   2013-11-04 15:00:57
nova-consoleauth folsom-cc                            nova             enabled    :-)   2013-11-04 15:00:57
nova-scheduler   folsom-cc                            nova             enabled    :-)   2013-11-04 15:01:05
nova-network     folsom-cc                            nova             enabled    :-)   2013-11-04 15:00:57
nova-compute     folsom-nc                            nova             enabled    :-)   2013-11-04 15:01:01
nova-network     folsom-nc                            nova             enabled    :-)   2013-11-04 15:00:56
nova-compute     folsom-nc2                           nova             enabled    :-)   2013-11-04 15:00:58
nova-network     folsom-nc2                           nova             enabled    :-)   2013-11-04 15:01:00
```

Fonte: Hentges, Thomé, Griebler, 2013.
Figura 55: Escalabilidade OpenStack.

3.5.2.7 Sistema de Segurança

O OpenStack utiliza-se de um componente próprio para autenticação chamado *Keystone*, este componente é instalado e configurado no *Controller Node* (seção B.1.1) e identificado no arquivo api-paste.ini do componente Nova no *Compute Node* (seção B.2). O *Keystone* fornece autenticação baseada em identidade, *token*, catálogo e políticas. A Identidade para validar as credenciais de usuário, projetos e funções, o *Token* é utilizado para autenticar os pedidos depois de verificadas as credenciais, o Catálogo fornece registro para encontrar *endpoints* (pontos de entrada ao serviço) e as Políticas de uso fornecem autorização ao uso dos projetos.

Em situação de utilização básica do *Keystone* ele mesmo gerencia os dados e tudo associado a ele, mas em casos mais específicos os dados de identidade podem ser retirados de um serviço de *backend* podendo ser o LDAP que armazena usuários e projetos separados e registra funções como entradas de projetos.

3.5.2.8 Opções de Rede

No teste de rede da nuvem OpenStack foi utilizado a conexão do tipo VLAN e *bridge*, através delas as máquinas virtuais e também os componentes da nuvem podem se comunicar. Depois de instaladas a VLAN e a *bridge* foram configuradas no arquivo nova.conf. A *bridge* além de ser configurada no mesmo arquivo também teve sua configuração feita no /etc/*network*/interfaces do *Controller Node* e dos *Compute Nodes.* Também quando se cria uma rede ela é associada a um novo projeto e a *bridge* é um dos parâmetros que são passados (seção B.2).

3.5.2.9 Opções de Armazenamento

Na questão de armazenamento na nuvem OpenStack implantada, as instâncias são armazenadas utilizando-se de um configuração *default* (qcow2), para este caso são criadas partições para armazenar as instâncias, essas partições criadas são montadas no fstab, e com permissões de acesso ao usuário e grupo Nova.

Para o armazenamento de instâncias ainda é possivel utilizar raw (formato de arquivos de imagens) ou LVM (*Logical Volume Management*, que define um padrão de gerenciamento para partições em discos do tipo IDE, SCSI ou FC) ao invés de qcow2 como foi utilizado.

3.5.2.10 Sistema de Integração

O OpenStack permite integração com os serviços EC2 (*Elastic Compute Cloud*) e S3(*Simple Storage Service*) pois possui os componentes Nova e *Glance* que auxiliam na comunicação com as API's da Amazon.

O S3 como o nome diz, é um Serviço de Armazenamento Simples, um armazenamento *online* que é fornecido pela Amazon, quando utilizado o S3 ele permite aos usuarios realizar *backup* dos dados para o Amazon S3. Porém, para utilizar do serviço Amazon S3 é necessário criar uma conta paga no site da Amazon.

Para configurar o S3 no OpenStack, no arquivo *glance*-api.conf é necessário configurar a linha *s3_store_host = URL* (importante quando se utiliza de armazenamento S3) por padrão vem configurada com ip da *localhost s3_store_host = 127.0.0.1:8080/v1.0/*. Também é importante definir a URL principal do serviço fornecido pelo S3, configurando a linha *s3_store_access_key = ACCESS_KEY*, com a chave de acesso aos serviços Amazon que se recebe ao cadastrar uma nova conta.

Definir a chave de acesso com 20 caracteres para autenticar em oposição ao *s3_store_host*, também no arquivo *glance-api.conf* na linha *s3_store_secret_key = SECRET_KEY*. Após definir chave com 40 caracteres para autenticar em oposição ao *s3_store_host* com a chave de acesso *s3_store_access_key* na linha *s3_store_bucket=BUCKET*. E definir o nome do *bucket* que vai utilizar imagens do *Glance* no S3. E por fim, a *linha s3_store_create_bucket_on_put* que por padrão é igual à *False*, se for verdade o componente *Glance* tentará criar bucket configurado em *s3_store_bucket* caso ele ainda não existir.

Já o EC2 Amazon é um serviço da Amazon que fornece a possibilidade de uma computação redimensionavel na nuvem, oferecendo um maior controle dos recursos computacionais. Utilizando o EC2 pode-se reduzir o tempo de reinicialização de novas instâncias do servidor. Porém o EC2 é um serviço do tipo pague-pelo-uso.

Para utilizar do serviço EC2 é necessário escolher um modelo de Amazon *Machine Image* (AMI) que seja pré-configurada ou criar uma com as aplicações, bibliotecas e dados necessários, configurar segurança e acesso às instâncias EC2, escolher o tipo de instancia e pagar somente pelos recursos utilizados.

Para integrar o EC2 ao OpenStack é necessário configura no arquivo nova.conf as linhas *Configuration option=Default value* (Descrição), *ec2_listen=0.0.0.0* (endereço IP para a api EC2 escutar), *ec2_listen_port=8773* (porta para a EC2 ouvir), *ec2_private_dns_show_ip=false* (retorna endereço IP como DNS privado), *keystone_ec2_url=http://localhost:5000/v2.0/ec2tokens* (URL para pedidos EC2), *lockout_attempts=5* (configura numero de auths que falham antes de bloquear), *lockout_minutes=15* (minutos para acionar o bloqueio), *lockout_window=15* (minutos para abrir janela de bloqueio).

3.5.2.11 Opções de Virtualização

Para o teste de virtualização no OpenStack foi utilizado o virtualizador KVM, o mais utilizado para virtualização no OpenStack, importante notar que as máquinas utilizadas possuíam suporte a virtualização o que também possibilitou o uso do KVM.

O KVM foi instalado e configurado apenas nos *Compute Nodes*, as configurações são feitas em arquivos específicos de componentes do OpenStack. Feito isso, já é possível contar com virtualização na nuvem criada. As imagens dos sistemas operacionais são carregadas a partir do componente Glance do OpenStack, sendo assim, o KVM necessita apenas rodar esses sistemas virtualizados para o usuário.

3.5.3 Análise Comparativa

Relacionado a interface das ferramentas, ambas as ferramentas apresentam uma interface clara e completamente funcional. Além disso, ambos apresentam o idioma português. Em relação ao uso, o OpenNebula apresenta uma interface em que tudo que está ao alcance do usuário é de fácil compreensão, diferente do Openstack, que apresenta algumas nomenclaturas que a primeira vista não ficam tão claras da sua funcionalidade é preciso que haja algum conhecimento de como a ferramenta funciona e de quais são os seus componentes para entender o funcionamento da interface.

O monitoramento presente nas ferramentas é limitado, não apresenta uma grande quantidade de informações ou a personalização para realizar tal tarefa. Em relação às informações obtidas, o OpenNebula apresenta uma gama de informações de maior relevância. Além disso, apresenta uma quantidade maior de informações em forma de gráficos, apresentando gráficos tanto sobre as máquinas virtuais como sobre os *hosts* que compõem a nuvem, diferente do OpenStack, que não apresenta gráficos e apresenta informações como o uso de processamento e o uso de memória apenas em relação a instâncias.

A tolerância a falhas é uma característica presente nas duas ferramentas, porém apenas conseguiu ser de fato funcional no OpenNebula, nele após ocorrer um erro no *host* responsável pela execução da máquina virtual, o controlador de nuvem que é aonde está armazenada a máquina virtual, poderá reexecuta-la em outro *host.* A tolerância a falhas presente no Openstack funciona da mesma maneira, porém não houve como de fato utiliza-la, pois para isso é necessário à utilização de outro componente (nova-volume) que é o responsável pelo armazenamento centralizado das máquinas virtuais, e a necessidade de reorganização da ambiente de rede utilizado, porém a complexidade de execução para estes componentes é maior e não houve a possibilidade de instalação dos mesmos.

O gerenciamento de energia não pode ser implementado em nenhuma das ferramentas. No Openstack houve a limitação de *hardware*, sendo que esta

característica somente é possível ser utilizada juntamente com processadores Intel Xeon, o qual não eram os disponibilizados nos computadores utilizados. Já o OpenNebula, apresentou outro entrave, a documentação referente à instalação das ferramentas responsáveis pelo gerenciamento e que funcionam juntamente com o OpenNebula, era praticamente inexistente. Sendo assim, não havia como sanar as duvidas referentes a problema que surgiram durante a tentativa de instalação.

O balanceamento de carga presente nas ferramentas funciona de maneira distinta, o Openstack possui um agendador que verifica os *hosts* responsáveis pela execução das máquinas virtuais para saber qual esta com menos sobrecarga para decidir em qual delas deverá executar uma nova máquina virtual. Já o OpenNebula, não verifica o uso dos computadores, ele apenas divide de maneira igual o número de máquinas virtuais para executar nos *hosts* de virtualização, não importando o quanto de carga cada máquina vai acrescentar no *host*.

Em relação ao teste de escalabilidade, nenhuma nuvem sofreu com problemas de execução ao acrescentar mais um nó na nuvem. Ambas estavam em execução e após a instalação e configuração do nó adicional, o mesmo foi adicionado à nuvem, onde comportou-se igual aos outros nós, podendo realizar suas tarefas da mesma maneira.

Em relação à segurança, em ambos os testes foi utilizada a autenticação disponibilizada pela própria ferramenta. Em ambas existe a possibilidade de controle e gerencia de grupos de usuários, onde é possível delimitar o uso de recursos para os usuários e para os grupos. Porém apenas no OpenNebula é possível a criação de ACL’s, delimitando exatamente ao o que o usuário tem acesso e o que ele pode fazer com o que é disponibilizado para ele.

A rede disponibilizada pelas nuvens tinha a finalidade de fornecer acesso as máquinas virtuais a rede, sendo que cumpriu com o objetivo. Porém também é possível em ambas às nuvens utilizar-se de VLAN para isolar grupos de máquinas virtuais, como se fossem um setor onde somente aquele grupo de computadores pudessem se comunicar. O acesso SSH funciona da mesma maneira como ao acessar outro computador físico na rede.

Ambas as nuvens utilizaram-se de armazenamento compartilhado para fornecer acesso às imagens através de um diretório compartilhado, funcionando de maneira eficiente. Porém, é possível também a utilização de outros métodos como iSCSI e LVM, que podem ser configurados em um computador exclusivo para armazenamento, fornecendo assim maior desempenho.

A integração com nuvens externas não foi possível, primeiramente por serem serviços pagos, além disso, seria necessário o uso de uma conexão de *Internet* de velocidade mais elevado, pois se trataria da transmissão de máquinas virtuais, podendo levar muito tempo a transmissão das mesmas. Mesmo assim, ambas as ferramentas possibilitam a integração com serviços disponibilizados pela Amazon, onde seria possível utilizar a nuvem disponibilizada pela Amazon como uma extensão da nuvem privada.

Em relação à virtualização para instalação de ambas as ferramentas foram utilizados o virtualizador KVM. O virtualizador funcionou sem problemas nas duas ferramentas, cumprindo sua função de executar as máquinas virtuais que lhe eram submetidas.

3.6 ANÁLISE DOS RESULTADOS

A partir dos testes realizados e da análise comparativa pode-se constatar que o OpenNebula acabou por ser a ferramenta que sobressaiu-se. Ela além de apresentar todas as características, mesmo que algumas não puderam ser de fato implementadas, conseguiu obter resultados melhores em relação ao Openstack.

O quadro 7 apresenta a análise dos resultados, onde pode ser verificado qual ferramenta leva uma vantagem em relação a outra em determinada característica, e se na prática foi possível comprovar que aquelas características existem de fato, mostrando assim se é coerente com a bibliografia.

Característica Avaliada	Desempenho	Coerente
Interface	OpenNebula	Sim
Gerenciamento de energia	OpenNebula	Indeterminado
Balanceamento de Carga	OpenStack	Sim
Rede	Empate	Parcialmente
Armazenamento	Empate	Parcialmente
Monitoramento	OpenNebula	Sim
Integração	Empate	Indeterminado
Virtualização	Empate	Parcialmente
Segurança	OpenNebula	Sim
Escalabilidade	Empate	Sim
Tolerância a falhas	OpenNebula	Parcialmente

Fonte: Hentges, Thomé, Griebler, 2013.
Quadro 7: Análise dos resultados.

Um dos pontos que o OpenNebula apresentou melhor desempenho é em relação a interface, pois percebe-se que os conceitos empregados são genéricos o suficiente a ponto de que usuários familiarizados com ambiente de redes são capazes de efetuar as configurações na nuvem, ao contrário do OpenStack, que tem sua nomenclatura própria. O desempenho em relação ao sistema de monitoramento também é visivelmente mais intuitivo e completo. No entanto, para chegar ao mesmo nível, junto a ferramenta OpenStack deveria-se integrar uma ferramenta de monitoramento de terceiros.

O desempenho em relação a tolerância a falhas também foi superior no OpenNebula, pois no Openstack haveria a necessidade da utilização de outros componentes pra obter esta função. Em relação o gerenciamento de energia, por mais que não pode ser implementado por falta de documentação, não há restrição relacionado ao *hardware*, diferente do Openstack, que necessita de processador específico para funcionar. Como o foco é o uso de estações de trabalho, seria de fato muito raro estas máquinas terem um processador utilizado em servidores.

O único ponto que o Openstack leva vantagem é o seu balanceamento mais eficiente do que o OpenNebula. Em relação à segurança, o OpenNebula apresenta

um sistema de gerencia de usuários e grupos mais eficiente, sendo possível delimitar o acesso de diversas maneiras aos recursos disponibilizados aos usuários.

Em relação às características restantes (escalabilidade, rede, armazenamento, virtualização e integração) ambos são semelhantes, sendo inapropriado dizer qual deles leva vantagem. Portanto após a pesquisa e a implantação das ferramentas o OpenNebula é a ferramenta que mais se adequa ao uso em nuvem privada com a utilização de estações de trabalho.

Analisando o que literatura nos apresenta e os resultados da prática em estações de trabalho, constatou-se que na característica de interface ambas as ferramentas apresentam uma interface gráfica, ou seja, é coerente com a literatura. Em relação ao gerenciamento de energia não pode ser comprovado, pois não foi possível a implantação em ambas as ferramentas. O balanceamento de carga pode ser comprovado, em ambos os testes a característica funcionou de acordo com a literatura.

No que se trata do teste de rede, nem todos os tipos de rede puderam ser testados, embora as ferramentas apresentassem opções para utilizar outros métodos, porém não é possível a utilização de todos de uma vez, sendo portanto parcialmente coerente com a literatura. Em relação ao armazenamento foi utilizado apenas um método de armazenamento, mas foi possível ver que os outros métodos existem, por isso esta parcialmente coerente com a literatura. O monitoramento presente nas ferramentas existe, sendo possível visualizar as informações de monitoramento coerente com a literatura.

Em relação a integração com outras nuvens, não pode ser realizado testes, pois não houve a possibilidade de investir, portanto sendo impossível saber se é coerente com a literatura. A virtualização também foi parcialmente testada, pois é possível utilizar somente um virtualizador em uma nuvem, portanto é parcialmente coerente com a literatura. No que se trata de segurança, ambos apresentam autenticação e controle por usuários e grupos, sendo coerente com a literatura.

Em relação a escalabilidade, ambas puderam ser testadas nas ferramentas, pois elas permitem a adição de nós a qualquer momento de maneira prática, portanto é coerente com a literatura. No que se trata de tolerância a falhas, pôde ser utilizado somente na ferramenta OpenNebula, portanto este teste foi parcialmente coerente com a literatura, sendo indeterminado se a tolerância funciona de fato no OpenStack.

3.7 TRABALHOS FUTUROS

Nesta seção são apresentados trabalhos que podem ser realizados dando continuidade ao trabalho atual, considerando a utilização de estações de trabalho da mesma maneira que foi realizado no presente trabalho.

- Testar as características que não puderam ser testadas (integração, gerenciamento de energia, tolerância a falhas (somente no OpenStack));

- Testar outras ferramentas pesquisadas na prática (por exemplo: Eucalyptus, CloudStack, OpenQRM) utilizando-se da mesma avaliação utilizada neste trabalho;

- Instalar uma aplicação que gere um estresse no ambiente de estações de trabalho, para avaliar o desempenho.

CONCLUSÃO

Durante o desenvolvimento deste trabalho de conclusão de curso, foram pesquisados diversos trabalhos relacionados e também muitas definições sobre computação em nuvem, ferramentas *open source*, modelos de serviço, arquitetura das ferramentas, incluindo uma pesquisa qualitativa sobre 10 ferramentas *open source* mais conhecidas no ambiente da computação em nuvem e entre essas ferramentas foram selecionadas duas para uma análise prática do funcionamento.

Os objetivos deste trabalho foram alcançados, tanto no que tange a obtenção das informações e dos conhecimentos práticos, que foram importantes para o sucesso da elaboração desta pesquisa. Além disso, foi possível instalar, comparar, analisar as ferramentas *open source* voltadas para o modelo de serviço IaaS.

Com relação ás hipóteses descritas no primeiro capítulo, verifica-se que a primeira hipótese analisa a possibilidade de que com o estudo, implantação e análise das ferramentas se possa identificar quais delas possuem melhor desempenho em uma nuvem com estações de trabalho. Foi constatado que avaliando individualmente as características elencadas é possível verificar qual das ferramentas possui melhor desempenho neste ambiente. Assim, podemos afirmar que em um ambiente de estações de trabalho a ferramenta OpenNebula apresenta um melhor desempenho comparado ao OpenStack baseando-se nas características analisadas.

Na segunda hipótese investiga-se o funcionamento das ferramentas testadas em ambiente com estações de trabalho para verificar se será coerente com

o relato da bibliografia. Após realizada a avaliação em relação aos testes para ver se condiz com a literatura, conforme o elencado no Quadro 7, pode-se dizer que é coerente somente nas características que foram implantadas de fato, pois nem todas puderam ser implantadas neste ambiente de estações de trabalho.

Além de tudo isso, foram produzidos dois documentos científicos: a publicação de um resumo no evento SAPS (Salão de Pesquisa Setrem 2013), conquistado o segundo lugar na apresentação oral na categoria Computação e este pode ser encontrado no Apêndice C; e a publicação no evento ERRC 2013 (Escola Regional de Redes de Computadores), podendo ser encontrado no Apêndice D.

Diante do que foi visto, a ferramenta que oferece um comportamento mais apropriado em estações de trabalho é o OpenNebula, podendo ser de grande ajuda para empresas que podem utilizar-se da ferramenta para realizar testes em ambientes como a nuvem.

Sabendo-se que a utilização da computação em nuvem está em ascensão. As ferramentas utilizadas evoluem depressa e cada vez com mais características para integrar as tecnologias comuns às nuvens, para assim facilitar o acesso às informações em qualquer lugar e a qualquer momento. Surge então, a necessidade constante de atualização da literatura e dos testes de utilização, para que seja possível fazer a escolha da ferramenta mais apropriada para um determinado cenário.

REFERÊNCIAS

ABIQUO [a]. **ABIQUO.** Disponível em: <http://www.abiquo.com/overview-technical/>. Acesso em 24 de julho de 2013.

ABIQUO [b]. ***Benefits*.** Disponível em < http://www.abiquo.com/enterprise/features/>. Acesso em 18 de agosto de 2013.

ANTONOPOULOS, Nick, GILLAM, Lee. ***Cloud Computing: Principles, Systems and Applications***. 1º Edição. Londres: Springer-Verlag, 2010, p.379.

APACHE VCL [a]. **Apache VCL.** Disponível em < http://vcl.apache.org/>. Acesso em 10 de agosto de 2013.

APACHE VCL [b]. ***Architecture*.** Disponível em < http://vcl.apache.org/info/architecture.html>. Acesso em 19 de agosto de 2013.

APACHE VCL [c]. ***Features*.** Disponível em < http://vcl.apache.org/info/features.html>. Acesso em 19 de agosto de 2013.

BAI, Xiaoying et al.. Cloud Testing Tools. In: International Symposium on Service Oriented System Engineering. 2011. Irvine-CA. p.1-12.

BARRETT, Daniel J., SILVERMAN Richard E., BYRNES, Robert G.. **SSH, *The Secure Shell: The Definitive Guide*.** Sebastopol, CA: O'Reilly Media, Inc, 2005, p.645.

BARRETT, Diane, KIPPER, Greg. ***Virtualization and Forensics: A Digital Forensic Investigator's Guide to Virtual Environments***. 1º Edição. Burlington: Syngress, 2010, p. 272.

BENATALLAH, Boualem, CHEN, Jinjun, RANJAN, Rajiv, WANG, Lizhe. ***Cloud Computing Methodology, Systems, and Applications***. 1º Edição. Boca Raton: Taylor & Francis Group, 2012, p. 815.

BUYYA, Rajkumar, BROBERG, James, GOSCINSKI, Andrzej M.. ***Cloud Computing: Principles and Paradigms***. 1º Edição. John Wiley and Sons, 2010, p. 660.

CAMPESATO, Oswald, NILSON, Kevin. ***Web 2.0 Fundamentals: With AJAX, Development Tools, and Mobile Platforms***. 1º Edição. Boston: Jones & Bartlett Learning, 2010, p.751.

CANONICAL. **Ubuntu *Enterprise Cloud*.** Disponível em: <http://www.canonical.com/news/ubuntu-10.04-server-edition>. Acesso em: 5 de set de 2013.

CITRIX. **Xen *Server***. Disponível em: <http://www.citrix.com.br/products/xenserver/how-it-works.html>. Acesso em: 13 de ago de 2013.

CLOUDSTACK**.** ***What is CloudStack?.*** Disponível em < http://cloudstack.apache.org/ >. Acesso em 31 de julho de 2013.

CONVIRTURE [a]. **ConVirt**. Disponível em: <http://www.convirture.com/solutions_cloudcomputing.php>. Acessado em 30 de julho de 2013.

CONVIRTURE [b]. ***Products: Key Features*.** Disponível em < http://www.convirture.com/products_features.php>. Acesso em 19 de agosto de 2013.

CONVIRTURE [c]. ***Products: Compare Products*.** Disponível em < http://www.convirture.com/products_compare.php>. Acesso em 20 de agosto de 2013.

COULOURIS, George, DOLLIMORE, Jean, KINDBERG, Tim. **Sistemas Distribuídos Conceitos e Projeto.** 4º Edição. Porto Alegre: Bookman Editora, 2007, p. 784.

CSA, Cloud Security Alliance. ***Security guidance for critical areas of focus in cloud computing V3.0.*** 3º Edição. Cloud Security Alliance, 2011, p. 176.

DOELITZSCHER, Frank, SULISTIO, Anthony, REICH, Christoph, KUIJS, Hendrik, WOLF, David. ***Private cloud for collaboration and e-Learning services: from IaaS to SaaS***. *Computing*, Nova Iorque, 1 Jan 2011. p.23-42.

ENDO, Patrícia Takako, GONÇALVES, Glauco Estácio, KELNER, Judith, SADOK, Djamel. ***A Survey on Open-source Cloud Computing Solutions.*** In: VIII *Workshop* em *Clouds*, *Grids* e Aplicações. 2010. Gramado-RS. Anais. Gramado: Sociedade Brasileira de Computação, 2010, p.3-16.

ESCALANTE, Armando. ***Handbook of Cloud Computing.*** 1º Edição. Nova Iorque: Springer-Verlag, 2010, p. 634.

FINN, Aidan, LOWNDS, Patrick. ***Mastering Hyper-V Deployment*.** 1º Edição. New Jersey: John Wiley & Sons, Inc., 2011, p.600.

FOX, Geoffrey. ***Collaboration and Community Grids***. *Proceedings of the International Symposium on Collaborative Technologies and Systems*. Washington, 2006. p.419-428.

GALLO, Michaele A., HANCOCK, William M.. **Comunicação entre Computadores e Tecnologias de Rede.** [S.l]. São Paulo: Thompson Learning, 2003, p. 673.

GIBSON, Darril. **MCTIP *Guide to* Microsoft Windows *Server* 2008, *Enterprise Administrarion*.** 1º Edição. Boston: Cengage Learning, 2010, p.448.

HALETKY, Edward. ***VMware ESX and ESXi in the Enterprise: Planning Deployment of Virtualization Servers***. 1º Edição. Boston: Pearson Education, 2011, p. 576.

HESS, Kenneth, NEWMAN, Amy. ***Practical Virtualization Solutions: Virtualization from the Trenches***. 1º Edição. Boston: Pearson Education, 2009, p. 336.

IBM. **KVM**. Disponível em: <http://www-03.ibm.com/systems/virtualization/kvm/whykvm.html>. Acesso em: 13 de agosto de 2013.

INTEL**. *Power Management, Security with OpenStack and Intel*.** Disponivel em < http://www.intel.com.br/content/www/br/pt/cloud-computing/cloud-builders-xeon-5500-5600-openstack-power-mgmt-security-guide.html>. Acesso em 04 novembro de 2013.

JESÚS, José de. **Navegando na Nuvem IBM, Parte 1: Um Manual sobre Tecnologias em Nuvem.** IBM® WebSphere® Developer Technical Journal, 20 jun 2012. p.21-36.

JOSEPH, Joshy, FELLENSTEIN, Craig. ***Grid Computing***. 1º Edição. Upper Saddle River: Prentice Hall Professional, 2004, p. 378.

JOSYULA, Venkata, ORR, Malcolm, PAGE, Greg. ***Cloud Computing Automating the Virtualized Data Center.*** 1º Edição. Indianapolis: Cisco Press, 2012, p. 377.

KEAHEY, Kate, FREEMAN, Tim. **Science Clouds: Early Experiences in Cloud Computing for Scientific Applications**. [S.l.], Nimbus Project, 2008, p.5.

KESWANI, Umeshkumar. ***High Performance Cluster And Grid Computing Solutions For Science.*** 1º Edição. Ann Arbor: ProQuest LLC, 2008, p.79.

KHURSHID, Ahmed, Al-NAYEEM, Abdullah, GUPTA, Indranil. ***Performance Evaluation of the Illinois Cloud Computing Testbed.*** [S.l], Urbana-Champaign: Illinois Digital Environment for Access to Learning and Scholarsip, 2009, p.12.

KOENIG, Gregory Allen. ***Efficient Execution of Tightly-coupled Parallel Applications in Grid Computing Environments*.** 1º Edição. Ann Arbor: ProQuest LLC, 2007, p.161.

KVM. **KVM**. Disponível em: <http://www.linux-kvm.org/page/Main_Page>. Acesso em 30 de julho de 2013.

LASZEWSKI, Gregor von, DIAZ, Javier, WANG, Fugang, FOX, Geoffrey C.. ***Comparison of Multiple Cloud Frameworks.*** 2012 IEEE Fifth International Conference on Cloud Computing. Washington. 2012. p.734-741.

LOVATO, Adalberto; EVANGELISTA, Mário Luiz Santos; GÜLLICH, Roque Ismael da Costa. **Metodologia da Pesquisa: normas para apresentação de trabalhos: redação, formatação e editoração**. 2º Edição. Três de Maio: Editora SETREM, 2007, p.151.

MACHADO, Claiton Prado. **Comparação de ferramentas de software Livre para administração de nuvem privada.** [S.l.], Canoas: Ulbra, 2011, p.18.

MAHMOOD, Zaigham, HILL, Richard. ***Cloud Computing for Enterprise Architectures.*** 1º Edição. Londres: Springer-Verlag, 2011, p. 327

MARKS, Eric A., LOZANO, Bob. ***Executive's Guide to Cloud Computing*****.** 1ºEdição. New Jersey: Published by John Wiley & Sons, Inc., 2010, p.285.

MILLER, Michael**.** ***Cloud Computing: Web-based applications that change the way you work and collaborate online.*** 1º Edição. Indianapolis: Que Publishing, 2009, p. 284.

MINOLI, Daniel. **A Networking Approach to Grid Computing.** 1º Edição. Hoboken: John Wiley & Sons, 2004, p.300.

NAGIOS. **Nagios**. Disponível em: <http://www.nagios.com/>. Acesso em 04 de setembro de 2013.

NIMBUS PROJECT. **Nimbus *Platform*.** Disponível em < http://www.nimbusproject.org/ >. Acesso em 12 de agosto de 2013.

OPENNEBULA [a]. ***About the OpenNebula.org Project.*** Disponível em < http://OpenNebula.org/about:about >. Acesso em 22 de julho de 2013.

OPENNEBULA [b]. **OpenNebula 4.2 *Detailed Features and Functionality*.** Disponível em < http://OpenNebula.org/documentation:features>. Acesso em 16 de agosto de 2013.

OPENQRM. **About OpenQRM.** Disponível em < http://www.openqrm-enterprise.com/about-openqrm/concepts.html>. Acesso em 18 de agosto de 2013.

OPENSTACK [a]. ***About OpenStack.*** Disponível em < http://www.openstack.org/software/ >. Acesso em 30 de julho de 2013.

OPENSTACK [b]. **Infraestrutura do OpenStack**. Disponível em: <http://docs.openstack.org/trunk/openstack-compute/admin/content/conceptual-architecture.html>. Acesso em 14 de ago de 2013.

OPENSTACK [c]. **OpenStack *Compute*.** Disponível em < http://www.openstack.org/software/openstack-compute/ >. Acesso em 16 de agosto de 2013.

PETERSEN, Richard. ***Fedora 10 Linux Administration, Networking, and Security***. 1º Edição. Alameda: Surfing Turtle Press, 2009, p. 820.

PILLAI, Anita S., SWASTHIMATHI, L.S.. ***A Study on Open Source Cloud Computing Plataforms*.** EXCEL International Journal of Multidisciplinary Management Studies. Zenit, 7 jul 2012. p.31-40.

RANSOME, James F., RITTINGHOUSE, John W.. ***Cloud Computing: Implementation, Management and Security***. 1º Edição. Boca Raton: CRC Press, 2009, p. 407.

RECHENBURG, Matthias. ***The openQRM Datacent Management and Cloud Computing Plataform.*** [S.l.] Koln: openQRM Enterprise, 2010, p.68.

RIBAS, Marcelo. **Nuvem Privada: Uma proposta de aplicação prática.** [S.l.]. Novo Hamburgo: Feevale, 2012, p.50.

RIMAL, Bhaskar Prasad, CHOI, Eunmi, LUMB, Ian. ***A Taxonomy and Survey of Cloud Computing Systems*.** In: NCM '09. Fifth International Joint Conference on INC, IMS, and IDC. 2009. Seoul – KR. Anais: Seoul: IEEE Computer Society, 2009. p.44-51.

RITTINGHOUSE, John W., RANSOME, James F. . ***Cloud Computing: Implementation, Management, and Security.*** 1º Edição. Estados Unidos: CRC Press, 2010, p.301.

RUSCHEL, Henrique, ZANOTTO, Mariana Susan, MOTA, Wélton Costa da. **Computação em Nuvem.** [S.l.], Curitiba: Pontifícia Universidade Católica do Paraná, 2010, p.15.

SABHARWAL, Navin, SHANKAR, Ravi. ***Apache Cloudstack Cloud Computing.*** 1º Edição. Reino Unido: Packt Publishing Ltd., 2013, p.294.

SEMPOLINSKI, Peter, THAIN, Douglas. ***A Comparison and Critique of Eucalyptus, OpenNebula and Nimbus*.** In: 2010 IEEE Second International Conference on Cloud Computing Technology and Science. 2010. Indianapolis-USA. Processo. Washington-USA: IEEE Computer Society, 2010. p.417-426.

SMOOT, Stephen R., TAN, Nam-Kee. ***Private Cloud Computing: Consolidation, Virtualization, and Service-oriented Infrastructure*.** 1º Edição. Waltham: Elsevier Inc, 2012, p.399.

SMYTH, Mary-Jane.**.** Disponível em < http://abicloud.org/display/ABI24/Abiquo+Architecture+Overview>. Acesso em 20 de agosto de 2013.

SOTO, Julio Alba**. OpenNebula: implantação de uma nuvem privada e orquestração das máquinas virtuais no paradigma da Computação em Nuvem.** [S.l]. Fortaleza: Universidade Federal do Ceará, 2011, p.61.

SOUSA, Flávio R. C., MOREIRA, Leonardo O., MACHADO, Javam C. **Computação em Nuvem: Conceitos, Tecnologias, Aplicações e Desafios.** In: III Escola Regional de Computação Ceará, Maranhão e Piauí. ERCEMAPI. 2009. Parnaíba-PI. Livro de resumos. Teresina: Editora da Universidade Federal do Piauí, 2009. p.150-175.

SUN MICROSYSTEMS. ***Introduction to Cloud Computing architecture.*** 1º Edição, Santa Clara: Sun Microsystems Inc, 2009, p.36.

TANENBAUM, Andrew S. **Redes de Computadores.** 4º Edição. Rio de Janeiro: Editora Campus Ltda, 2003, p.945.

TANENBAUM, Andrew S., STEEN, Maarten Van. **Sistemas Distribuídos princípios e paradigmas.** 2º Edição. São Paulo: Pearson Prentice Hall, 2007, p.402.

TOMSHO, Gregory. ***Guide to Networking Essentials***. 6º Edição. Boston. Cengage Learning, 2011, p. 688.

TORALDO, Giovanni. **OpenNebula 3 *Cloud Computing*.** 1º Edição. Reino Unido: Packt Publishing Ltd., 2012, p.314.

TORRES, Gabriel. **Redes de Computadores Curso Completo.** 1º Edição. Rio de Janeiro: Axcel Books do Brasil Editora Ltda, 2001, p.664.

TOSSATO, Daniele. ***Citrix Xenserver 6.0 Administration Essential Guide***. 1º Edição. Birmingham: Packt Publishing Ltd, 2012, p. 364.

VACCA, John R. ***Computer and Information Security***. 2º Edição. Waltham: Elsevier Inc, 2013, p. 1171.

VECCHIOLA, Christian, CHU, Xingchen, BUYYA, Rajkumar. ***Aneka: A Software Platform for .NET-based Cloud Computing.*** High Speed and Large Scale Scientific Computing. Amsterdam, Jul 2009. p.267-295.

VELTE, Anthony T. et al. . ***Cloud Computing: A practical Approach.*** 1º Edição. [S.I.] The McGraw-Hill Companies, 2010, p. 334.

VERAS, Manoel, TOZER, Robert. ***Cloud Computing:*** **Nova Arquitetura da TI.** 1º Edição. Rio de Janeiro: Brasport Livros e Multimídia Ltda, 2012, p.240.

VERAS, Manoel. **Virtualização: Componente Central do Datacenter**. 1º Edição. Rio de Janeiro: Brasport Livros e Multimidia Ltda, 2011, p. 364.

VMWARE. **VMware**. Disponível em: <http://www.vmware.com/br/virtualization/virtualization-basics/what-is-virtualization.html>. Acesso em: 13 de agosto de 2013.

VMWARE. ***White Paper: Understanding Full Virtualization, Paravirtualization, and Hardware Assist*.** [S.l., S.n.], VMware, 2007, p.17.

VORAS, I. MIHALJEVIC, B. ; ORLIC, M. ; PLETIKOSA, M. ; ZAGAR, M. ; PAVIC, T. ; ZIMMER, K. ; CAVRAK, I. ; PAUNOVIC, V. ; BOSNIC, I. ; TOMIC, S. ***Evaluating Open-Source Cloud Computing Solutions*.** In: MIPRO, 2011 Proceedings of the 34th International Convention. 2011. Opatija-HR. Anais. Washington: IEEE Computer Society, 2011. p.209-214.

VUGT, Sander van. ***Beginning Ubuntu LTS Server Administration: From Novice to Professional***. 2º Edição. Berkeley: Apress, 2008, p. 424.VACCA, John R. Computer.

WARDLEY, Simon, GOYER, Etienne, BARCET, Nick. ***Ubuntu Enterprise Cloud Architecture.*** [S.l.], Man Island: Canonical, 2009, p.18.

WILKINSON, Barry. ***Grid Computing: Techniques and Applications.*** 1º Edição. Boca Raton: CRC Press, 2010, p.387.

WINKLER, Vic. ***Securing the Cloud: Cloud Computer Security Techniques and Tactics*.** 2º Edição. Waltham: Elsevier, 2011, p.314.

APÊNDICES

Apêndice A: Instalação do OpenNebula

Apêndice B: Instalação do Openstack

Apêndice C: Resumo para o SAPS

Apêndice D: Artigo ERRC 2013

APÊNDICE A: INSTALAÇÃO DO OPENNEBULA

Esta seção mostra passo a passo como foi feita a instalação do *frontend* e dos nós no OpenNebula.

A.1 Instalação e configuração da rede

Primeiramente foi instalado o pacote *bridge-utils* para que assim possa ser criada uma interface *bridge* que será utilizada para a comunicação com as máquinas virtuais.

Comece instalando o bridge-utils:

```
$ apt-get install bridge-utils
```

Após isso, foi configurado o arquivo /etc/network/interfaces, para que fosse configurada a interface *bridge*. Deixando-o desta maneira.

```
auto eth
iface eth0 inet manual

auto br0
iface br0 inet static
address 192.168.1.20
netmask 255.255.255.0
network 192.168.1.0
broadcast 192.168.1.255
gateway 192.168.1.1
dns-nameservers 8.8.8.8
dns-search example.com
bridge_ports eth0
bridge_fd 9
bridge_hello 2
```

```
bridge_maxage 12
bridge_stp off
```

Após isto, foi criado o grupo e o usuário que serão usados pelo OpenNebula, além de ser definida a senha para o usuário e defini-lo como dono da pasta.

Neste comando foi criado o grupo oneadmin.

```
$ groupadd –g 10000 oneadmin
```

O seguinte comando foi dado para a criação do usuário oneadmin, sendo lhe atribuído uma pasta diferente do padrão de criação dos usuários e o definindo a um grupo.

```
$ useradd –u 10000 –m oneadmin –d /var/lib/one –s /bin/bash -g 10000 oneadmin
```

E então foi denifida a senha para o usuário oneadmin:

```
$ passwd oneadmin
```

A.2 Instalação do NFS e geração da chave SSH.

Em seguida foi feita a instalação do NFS e foi gerada uma chave SSH, o NFS foi instalado, pois ele foi o responsável por compartilhar o diretório na rede, em que estão armazenadas as máquinas virtuais para dar acesso aos nós, além da chave SSH.

Primeiramente foi instalado o NFS, através do comando:

```
$ apt-get install nfs-kernel-server
```

E configurado o arquivo /etc/exports para compartilhar uma determinada pasta na rede da seguinte maneira:

```
/var/lib/one 192.168.1.0/24(rw,sync,no_subtree_check,
no_root_squash,anonuid=10000,anongid=10000)
```

E posteriormente iniciado o serviço.

```
$ /etc/init.d/nfs-kernel-server start
```

Então foi feito o acesso com o usuário oneadmin e gerado uma chave SSH para este usuário o qual deve ter acesso aos nós sem senha.

```
$ su -l oneadmin
$ ssh-keygen
```

Após isso foi passado para as chaves autorizadas, para que fosse possível o uso da chave e não precisasse utilizar-se de senha.

```
$ cat ~/.ssh/id_rsa.pub > ~/.ssh/authorized_keys
```

E então criado um arquivo.

```
$ nano ~/.ssh/config
```

E foram adicionadas as seguintes linhas para que não adicione chaves SSH aos known_hosts a cada nova conexão.

```
Host *
StrictHostKeyChecking no
```

A.3 Instalação OpenNebula e dependências

Para isso foi feito o *download* da última versão do OpenNebula a partir deste link http://dev.OpenNebula.org/packages/OpenNebula-4.2.0/OpenNebula-4.2.0.tar.gz

Mas antes de instalar o OpenNebula foi preciso instalar suas dependências usando o comando apt-get.

```
$ apt-get install libsqlite3-dev libxmlrpc-c3-dev g++ ruby libopenssl-ruby libssl-dev ruby-dev libxml2-dev libmysqlclient-dev libmysql++-dev libsqlite3-ruby libexpat1-dev rake rubygems libxml-parser-ruby1.8 libxslt1-dev genisoimage scons
```

Antes de prosseguir com a instalação das dependências é preciso instalar uma versão mais recente do Ruby, senão não irá instalar uma das dependências (nokogiri), para isto faça:

```
$ apt-get install python-software-properties
$ apt-add-repository ppa:brightbox/ruby-ng
$ apt-get update
$ apt-get install build-essential ruby rubygems ruby-switch
$ apt-get install ruby1.9.1-full
$ ruby-switch --set ruby1.9.1
```

Agora continue com a instalação:

```
$ gem install nokogiri rake xmlparser
$ apt-get install mysql-server
```

Após a instalação do Mysql e definida a senha durante sua instalação, foram utilizados os seguintes comandos, para criar um usuário, um banco de dados e dar permissão ao banco de dados para aquele usuário criado:

Primeiro foi acessado a linha de comando do Mysql com o seguinte comando.

```
$ mysql –uroot -psenha
```

Então foi criado o usuário oneadmin para o Mysql:

```
$ CREATE USER ‘oneadmin’@’localhost’ IDENTIFIED BY ‘oneadmin’;
```

Depois foi criado o banco de dados:

```
$ CREATE DATABASE OpenNebula;
```

E por último foi dado acesso completo ao banco de dados criado para o usuário oneadmin, e então saiu-se da linha de comando do Mysql.

```
$ GRANT ALL PRIVILEGES ON OpenNebula. * TO ‘oneadmin’ IDENTIFIED BY ‘oneadmin’
quit;
```

Antes de instalar o OpenNebula, configure o suporte ao Mysql, para que o OpenNebula utilize o Mysql ao invés do sqlite.

```
$ cd ~/OpenNebula-4.2.0
$ scons sqlite=no mysql=yes
```

E então foi instalado o OpenNebula:

```
$ ./install.sh -u oneadmin -g oneadmin -d /var/lib/one
```

Foi criado um arquivo de perfil (~/.bash_profile) para definir variáveis de ambiente, onde foi definido o caminho de instalação do OpenNebula, o arquivo onde

foi armazenado o usuário e senha do usuário oneadmin e o caminho de onde se encontram as gems.

```
export ONE_LOCATION=/var/lib/one
export ONE_AUTH=$ONE_LOCATION/.one/one_auth
export ONE_XMLRPC=http://localhost:2633/RPC2
export
PATH=$ONE_LOCATION/bin:/usr/local/bin:/var/lib/gems/1.8/bin/:/var/lib
/gems/1.8/:$PATH
```

Após criado o arquivo e definidas as variáveis foi executado o arquivo.

```
$ source ~/.bash_profile
```

Então foi criado uma pasta e um arquivo para armazenar o usuário e a senha do oneadmin.

```
$ su - oneadmin
$ mkdir ~/.one
$ nano ~/.one/one_auth
oneadmin:senha
```

Após isso, foi aberto o arquivo /var/lib/one/etc/oned.conf e comentada a seguinte linha para que o OpenNebula não utilize o Sqlite.

```
#DB = [ backend = "sqlite" ]
```

E descomentadas as seguintes linhas, para que o OpenNebula passe a utilizar o Mysql.

```
DB = [ backend = "mysql",
server = "localhost",
port = 0,
```

```
user = "oneadmin",
passwd = "oneadmin",
db_name = "OpenNebula" ]
```

Para ativar a tolerância a falhas, teve-se que entrar no arquivo /var/lib/one/etc/oned.conf, e procurar a seção *Fault Tolerance Hooks*, e abaixo disto descomentar as seguintes linhas:

```
HOST_HOOK = [
name = "error",
on = "ERROR",
command = "ft/host_error.rb",
arguments = "$ID -r",
remote = "no" ]

VM_HOOK = [
name = "on_failure_recreate",
on = "FAILED",
command = "/usr/bin/env onevm delete --recreate",
arguments = "$ID" ]
```

A.4 Instalação do Sunstone

A princípio, o Sunstone já está instalado, porém precisa de alguns pacotes que são instalados a seguir, para poder iniciar, então isto foi feito executando os seguintes comandos de instalação.

```
$ apt-get install rails thin
$ gem install json sinatra thin
```

Execute este commando, para evitar que de erro no formato de data no Ruby.

```
$ sudo sed -i 's/ 00:00:00.000000000Z//' /var/lib/gems/1.8/specifications/*
```

E então foi alterado o arquivo de configuração do Sunstone /var/lib/one/etc/sunstone-server.conf para definir o IP em que se acessa o Sunstone, mudando apenas esta opção de acordo com o IP.

```
:host: 192.168.x.x
```

A.5 Instalação no node

A.5.1Instalação do bridge-utils e criação do usuário oneadmin

Comece instalando o bridge-utils (para que assim possa ser criada uma interface *bridge* que será utilizada para a comunicação com as máquinas virtuais) com o seguite comando:

```
$ apt-get install bridge-utils
```

Após isso foi configurado o arquivo /etc/network/interfaces para que fosse configurada a interface *bridge*.

```
auto eth0
iface eth0 inet manual

auto br0
iface br0 inet static
address 192.168.1.30
netmask 255.255.255.0
network 192.168.1.0
broadcast 192.168.1.255
gateway 192.168.1.1
dns-nameservers 8.8.8.8
```

```
dns-search example.com
bridge_ports eth0
bridge_fd 9
bridge_hello 2
bridge_maxage 12
bridge_stp off
```

Após isto, foi criado o grupo e o usuário que serão usados pelo OpenNebula, além de ser definida a senha para o usuário e defini-lo como dono da pasta.

Neste comando foi criado o grupo oneadmin.

```
$ groupadd –g 10000 oneadmin
```

O seguinte comando foi dado para a criação do usuário oneadmin, sendo lhe atribuído uma pasta diferente do padrão de criação dos usuários e o definindo a um grupo.

```
$ useradd –u 10000 –m oneadmin –d /var/lib/one –s /bin/bash -g 10000 oneadmin
```

E então foi denifida a senha para o usuário oneadmin

```
$ passwd oneadmin
```

A.5.2 Instalação do NFS

Primeiramente foi instalado o NFS, para que possa ser feita a montagem do diretório que foi compartilhado pelo *frontend,* através do comando:

```
$ apt-get install nfs-kernel-server
```

Também, foi configurado o arquivo /etc/fstab para montar o diretório compartilhado na rede da seguinte maneira, onde é passado o IP do *host* que está compartilhando o diretório, além do diretório que ele compartilha, o diretório local de montagem, o que será utilizado para fazer a montagem, o tipo de montagem, se é possível interromper o processo de montagem e a automatização da montagem ao iniciar o sistema operacional, respectivamente.

```
192.168.1.20:/var/lib/one  /var/lib/one  nfs   soft,intr,auto
```

E posteriormente foi montado o diretório.

```
$ mount /var/lib/one
```

Não é preciso copiar a chave SSH para os nós, pois ao montar o diretório, a chave vai estar automaticamente dentro dele, permitindo o uso de SSH sem senha.

A.5.3 Instalação dos softwares de virtualização

Posteriormente foi feita a instalação dos pacotes responsáveis pela virtualização:

```
$ apt-get install qemu-kvm libvirt-bin ubuntu-vm-builder
```

Após terminada a instalação são adicionados o usuário root e o usuário oneadmin aos grupos libvirtd e kvm:

```
$ adduser `id -un` libvirtd
$ adduser oneadmin libvirtd
$ adduser `id -un` kvm
$ adduser oneadmin kvm
```

Para então ser alterada a configuração dos seguintes arquivos (/etc/libvirt/libvirtd.conf, /etc/libvirt/qemu.conf) de acordo com o que é demonstrado, descomente a linha, se a mesma estiver comentada.

No arquivo libvirtd.conf, a primeira linha abaixo serve para desabilitar a escuta de conexões TLS, a segunda linha linha habilita a escuta de conexões TCP, a terceira linha desabilita o mDNS, a quarta linha permite o uso de *socket* para o grupo oneadmin, e a quinta linha define o nível de acesso as máquinas virtuais, neste caso ele dá acesso total.

```
listen_tls = 0
listen_tcp = 1
mdns_adv = 0
unix_sock_group = “oneadmin”
unix_sock_rw_perms = “0777”
```

No arquivo qemu.conf, a primeira linha abaixo define em que IP’s serão escutadas conexões VNC, a segunda e a terceira linha definem respectivamente o usuário e o grupo que tem acesso para utilizar o QEMU, e a quarta linha desabilita a troca dinâmica de proprietário das máquinas virtuais.

```
vnc_listen = “0.0.0.0”
user = “oneadmin”
group = “oneadmin”
dynamic_ownership = 0
```

Após alterar os arquivos, o serviço libvirt foi reiniciado para aceitar as novas configurações.

```
$ sudo service libvirt-bin restart
```

Configure para libvirt dar acesso aos usuários do grupo oneadmin

```
$ chown :oneadmin /var/run/libvirt/libvirt-sock
```

Após finalizada a instalação do OpenNebula, tanto do *frontend* quanto do nó, o mesmo foi iniciado através do comando a seguir e também foi iniciada a interface Sunstone:

```
$ one start
$ sunstone-server start
```

Então foi adicionado o primeiro nó ao *host*, onde é utilizado o comando onehost *create* para adicionar um nó, mais o nome do nó (nomedohost) e em seguida foram passados parâmetros que definem o driver de monitoramento, o *driver* de virtualização e o tipo de conexão, respectivamente.

```
$ onehost create nomedohost --im kvm --vm kvm --net dummy
```

APÊNDICE B: INSTALAÇÃO DO OPENSTACK

B.1 Instalação e configuração do Controller Node:

É necessário 3 partições e logar como root.

No arquivo /etc/hosts configurar um domínio da seguinte forma adicionando o IP da máquina e o nome dela além de configurar a rede privada e a rede pública com IP, domínio e nome da máquina:

```
127.0.0.1 localhost
192.168.0.232 folsom-cc
#Rede Privada
192.168.0.232 folsom-cc-private.domain.local folsom-cc-private
192.168.0.234 folsom-nc-private.domain.local folsom-nc-private
#Rede Publica
192.168.0.232 folsom-cc-public.domain.local folsom-cc-public
192.168.0.234 folsom-nc-public.domain.local folsom-nc-public
```

Instalar o repositório Folsom (versão utilizada do OpenStack), para armazenar, recuperar e instalar pacotes no sistema.

```
$ apt-get install ubuntu-cloud-keyring
```

```
$ echo "deb http://ubuntu-cloud.archive.canonical.com/ubuntu precise-updates/folsom main" > /etc/apt/sources.list.d/folsom.list
```

Após essas instalações é aconselhado atualizar (*update, upgrade, dist upgrade*) e reiniciar o sistema. Então, pode-se instalar e configurar o MySQL (Banco de dados), o NTP para sincronizar os relógios dos computadores e o RabbitMQ que é um sistema chamado de *Message Broker*, traduz uma linguagem de programação em outra conhecida internacionalmente.

MySQL:

```
$ apt-get install mysql-server python-mysqldb
```

Colocar uma senha para o root e acessar o banco:

```
$ mysql -u root -p
```

Após isso, logar no mysql e executar o comando a baixo para conceder todos os privilégios no mysql para o usuário ‘root’ de qualquer máquina identificando-se através da senha ‘cloud’:

```
$ mysql> GRANT ALL ON *.* TO 'root'@'%' IDENTIFIED BY 'cloud';
$ mysql> quit;
```

Configurar para aceitar conexões em todas as interfaces para que assim possa acessar o banco de dados de qualquer uma das interfaces que tiver configurado.

```
$ sed -i 's/127.0.0.1/0.0.0.0/g' /etc/mysql/my.cnf
$ service mysql restart
```

RabbitMQ:

```
$ apt-get install rabbitmq-server
```

Criar uma conta de usuário RabbitMQ que será utilizada pelos serviços OpenStack:

```
$ rabbitmqctl add_user openstack_rabbit_user cloud
```

Crie o RabbitMQ vhost usado pelo *Quantum* (sistema que gerencia redes e endereços IPs):

```
$ rabbitmqctl add_vhost /quantum
```

Definir as permissões para a nova conta do usuário RabbitMQ, o rabbitmqctl faz o gerencimanto do banco de dados do usuário interno RabbitMQ, as permissões são uma forma de controle de acesso, e o comando a baixo define as permissões de usuário dizendo ao RabbitMQ conceder ao usuário acesso ao host virtual, sendo que todos podem escrever e ler em todos os recursos.

```
$ rabbitmqctl set_permissions -p / openstack_rabbit_user ".*" ".*" ".*"
$ rabbitmqctl set_permissions -p /quantum openstack_rabbit_user ".*" ".*" ".*"
```

NTP:

Como foi descrito anteriormente é utilizado para sincronizar os relógios dos computadores e foi instalado e iniciado com os seguintes comandos:

```
$ apt-get install ntp
$ service ntp restart
```

VLAN e Bridge-Utils:

Foi instalado VLAN (rede local virtual) que permite acrescentar no mesmo domínio broadcast, *hosts* que tem sua localização física diferente. E também instalado o pacote bridge-utils, este instala os componetes que serão necessários para a criação, configuração e gerenciamento da *bridge*.

```
$ apt-get install vlan bridge-utils
```

Habilite o IP_Forwarding, para permitir roteamento de pacotes:

```
$ sed -i 's/#net.ipv4.ip_forward=1/net.ipv4.ip_forward=1/g' /etc/sysctl.conf
```

B.1.1 Instalação dos componentes do OpenStack:

Keystone:

Primeiro componente do OpenStack a ser instalado e configurado será o *Keystone* responsável pelo serviço de autenticação de toda a nuvem:

```
$ apt-get install keystone
```

Criação da base de dados do *keystone*:

```
$ mysql -u root -p
$ mysql> CREATE DATABASE keystone;
$mysql> GRANT ALL ON keystone.* TO 'keystone'@'%' IDENTIFIED BY 'keystone';
$ mysql> quit;
```

Atualize os dados do arquivo /etc/keystone/keystone.conf alterando a conexão com o banco para mysql e difinir um *token* de autenticação (admin_token para ADMIN) que permite ao usuário se comunicar com a API para poder criar, por exemplo, os usuários e os projetos:

```
Connection=mysql://root:cloud@folsom-cc-private/keystone
admin_token = ADMIN
```

Reinicie o serviço e sincronize o banco de dados:

```
$ service keystone restart
$ keystone-manage db_sync
```

Foi utilizado os *scripts* do repositório Git do Nimbula para criar as entradas de usuários, *Tenants* e *Endpoint's*:

```
$ mkdir /home/cloud/scripts
$ cd  /home/cloud/scripts

$ wget https://raw.github.com/nimbula/OpenStack-Folsom-Install-guide/master/Keystone_Scripts/Without%20Quantum/keystone_basic.sh

$ wget https://raw.github.com/nimbula/OpenStack-Folsom-Install-guide/master/Keystone_Scripts/Without%20Quantum/keystone_endpoints_basic.sh
```

No *script* /home/cloud/scripts/keystone_basic.sh alterar a variável $DOMAIN para domínio utilizado e a variável $HOST_IP para o *hostname* do servidor. Esse *script* define variáveis (abaixo) utilizadas para quando executado preencher o banco de dados do *Keystone* com os dados de usuários, serviços, projetos etc.

```
HOST_IP=${HOST_IP:-folsom-cc-private}
DOMAIN=${DOMAIN:-domain.local}
ADMIN_PASSWORD=${ADMIN_PASSWORD:-cloud}
SERVICE_PASSWORD=${SERVICE_PASSWORD:-cloud}
export SERVICE_TOKEN="ADMIN"
export SERVICE_ENDPOINT="http://${HOST_IP}:35357/v2.0"
SERVICE_TENANT_NAME=${SERVICE_TENANT_NAME:-service}
```

No *script* /home/cloud/scripts/keystone_endpoints_basic.sh alterar as variáveis $HOST_IP e $EXT_HOST_IP para o *hostname* do servidor:

```
# Host address
HOST_IP=folsom-cc-private
EXT_HOST_IP=folsom-cc-public
```

```
# MySQL definitions
MYSQL_USER=root
MYSQL_DATABASE=keystone
MYSQL_HOST=$HOST_IP
MYSQL_PASSWORD=cloud

# Keystone definitions
KEYSTONE_REGION=RegionOne
export SERVICE_TOKEN=ADMIN
export SERVICE_ENDPOINT="http://${HOST_IP}:35357/v2.0"
```

Dar permissão de execução aos dois *scripts* (keystone_basic.sh e keystone_endepoints_basic.sh):

```
$ chmod +x keystone_basic.sh
$ chmod +x keystone_endpoints_basic.sh
```

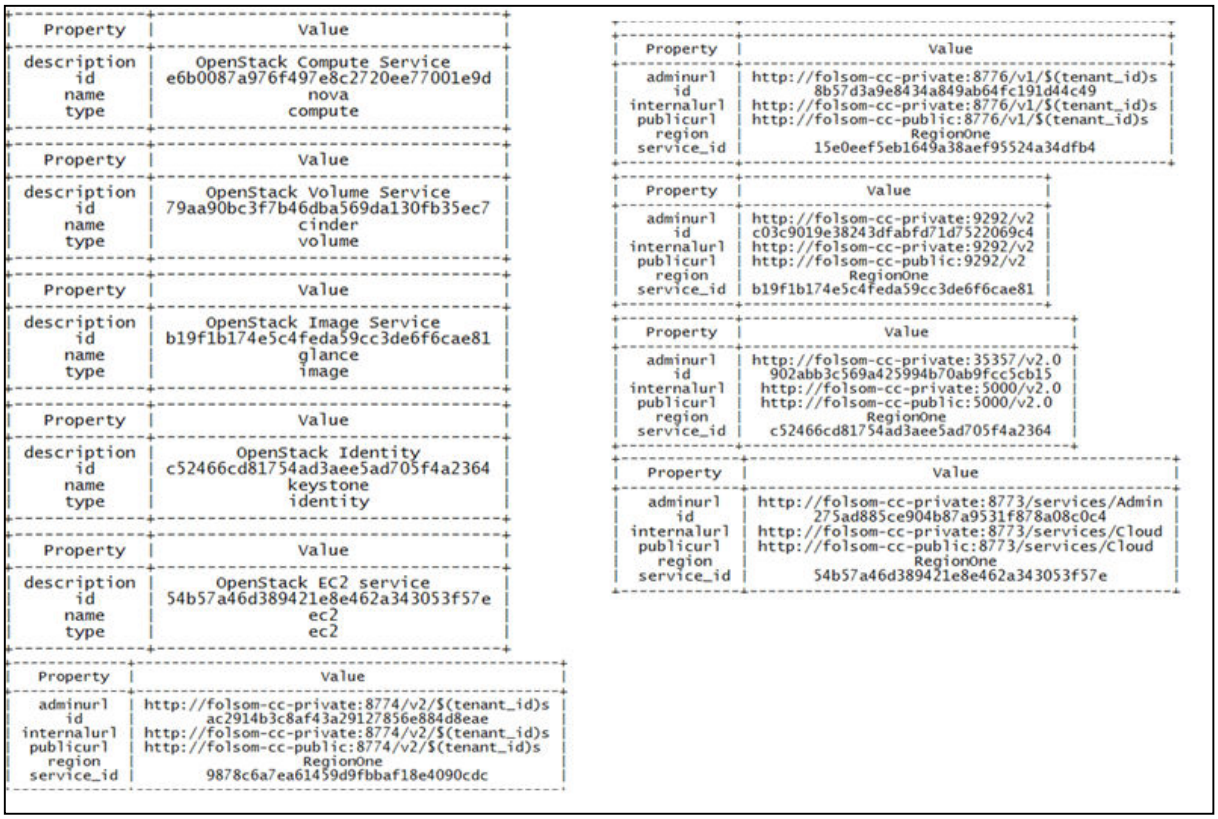
Obs.: Esses *scripts* devem sempre ser executados na sequencia correta primeiro o keystone_basic e keystone_depois o end_points.

Ao executar o *script* keystone_basic.sh não deverá retornar nada na tela:

```
$ ./keystone_basic.sh
```

Executar o *script* keystone_enpoints_basic.sh que deve retornar os valores abaixo (que é uma amostra dos dados que acabaram de ser inseridos no banco de dados do *keystone*):

```
$ ./keystone_endpoints_basic.sh
```



Criar um arquivo local (~/.creds) com as credenciais do OpenStack e carregar suas informações nas variáveis de ambiente para que assim o sistema tenha armazenado as variáveis mais importantes para o funcionamento do OpenStack:

```
export OS_NO_CACHE=1
export OS_TENANT_NAME=admin
export OS_USERNAME=admin
export OS_PASSWORD=cloud
export OS_AUTH_URL="http://folsom-cc-private:5000/v2.0/"
```

Incluir as informações nas variáveis de ambiente do usuário:

```
$ source ~/.creds
$ echo " source ~/.creds" >> ~/.bash_profile
```

Verificar se o *keystone* está rodando corretamente (maneira simples):

```
$ apt-get install curl
$ curl http://folsom-cc-private:35357/v2.0/endpoints -H 'x-auth-token: ADMIN' | python -m json.tool
```

A saída do comando deve ser algo semelhante com:

Glance:

Próximo passo instalar e configurar o *Glance* que é o componente do OpenStack responsável por armazenar e gerenciar as imagens:

```
$ apt-get install glance
```

Crie uma partição primária (para armazenar as imagens) em uma das existentes com o comando fdisk, com a opção ‘n’ estará criando uma nova partição, a opção ‘p’ é para imprimir a tabela de partições e ‘w’ para escrever a nova tabela de partições e em seguida sair deste modo:

```
$ fdisk /dev/sda2
$ fdisk> n
$ fdisk> p
```

```
$ fdisk> enter
$ fdisk> enter
$ fdisk> enter
$ fdisk> w
```

Formatar a partição como ext4:

```
$ mkfs.ext4 /dev/sda2
```

A partição deve ser incluída no /etc/fstab e depois montada:

```
$ echo "/dev/sda2 /var/lib/glance/images ext4 errors=remount-ro
0 1" >>  /etc/fstab
$ mount -a
```

Deve ser alteradas as permissões do diretório para o usuário e grupo *glance*:

```
$ chown -Rv glance.glance /var/lib/glance/images
```

Criar a base de dados para o glance para armazenar as imagens e demais dados importantes a este componente, para isso acessar o mysql, criar o banco de dados *glance* e conceder todos os privilégios do banco glance ao usuário glance, identificado pela senha glance:

```
mysql -u root -p
mysql> CREATE DATABASE glance;
mysql> GRANT ALL ON glance.* TO 'glance'@'%' IDENTIFIED BY
'glance';
mysql> quit;
```

Atualizar a seção [filter:authtoken] no arquivo /etc/glance/glance-api-paste.ini:

```
[filter:authtoken]
paste.filter_factory = keystone.middleware.auth_token:filter_factory
delay_auth_decision = true
auth_host = folsom-cc-private
auth_port = 35357
auth_protocol = http
admin_tenant_name = service
admin_user = glance
admin_password = cloud
```

Atualizar o arquivo /etc/glance/glance-registry-paste.ini:

```
[filter:authtoken]
paste.filter_factory = keystone.middleware.auth_token:filter_factory
auth_host = folsom-cc-private
auth_port = 35357
auth_protocol = http
admin_tenant_name = service
admin_user = glance
admin_password = cloud
```

Atualizar também o arquivo /etc/glance/glance-api.conf:

```
sql_connection = mysql://root:cloud@folsom-cc-private/glance

[keystone_authtoken]
auth_host = folsom-cc-private
auth_port = 35357
auth_protocol = http
admin_tenant_name = service
admin_user = glance
admin_password = cloud
```

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

Atualizar o arquivo /etc/glance/glance-registry.conf:

```
sql_connection = mysql://root:cloud@folsom-cc-private/glance

[keystone_authtoken]
auth_host = folsom-cc-private
auth_port = 35357
auth_protocol = http
admin_tenant_name = service
admin_user = glance
admin_password = cloud
[paste_deploy]
flavor = keystone
```

Reiniciar os serviços glance-api e glance-registry:

```
$ service glance-api restart
$ service glance-registry restart
```

Sincronizar a base de dados do glance:

```
$ glance-manage db_sync
```

É possível que apareça um aviso que "useexisting" é obsoleto, porém não tem problema, siga em frente. Reiniciar os serviços glance-api e glance-registry novamente:

```
$ service glance-registry restart
$ service glance-api restart
```

Para testar a instalação do *Glance* utiliza-se uma imagem que pode ser de *Cloud* Cirros (essa imagem contém um sistema linux básico rodando). Esse teste pode ser feito de duas maneiras:

- Direto da *Internet* com:

```
$ glance image-create --name Cirros --is-public true --container-format bare --disk-format qcow2 --location https://launchpad.nett/cirros/trunk/0.3.0/+download/cirros-0.3.0-x86_64-disk.img
```

Ou usando um arquivo local (para essa situação foi utilizado um arquivo local):

```
$ mkdir /home/cloud/images
$ cd /home/cloud/images
$ wget https://launchpad.net/cirros/trunk/0.3.0/+download/cirros-0.3.0-x86_64-disk.img
$ glance image-create --name Cirros --is-public true --container-format bare --disk-format qcow2 < cirros-0.3.0-i386-disk.img
```

Se ocorrer este erro:

```
Error communicating with /v1/images: [Errno 113] No route to host
```

Então, configurar o arquivo /etc/network/interfaces com o IP correto para sua necessidade:

```
auto lo
iface lo inet loopback
auto br100
iface br100 inet static
        address 192.168.0.232
        netmask 255.255.255.0
        network 192.168.0.255
```

```
broadcast 192.168.0.255
gateway 192.168.0.1
dns-nameservers 8.8.8.8
bridge_ports eth0
bridge_stp off
bridge_maxwait 0
bridge_fd 0
auto eth1
iface eth1 inet static
        address 192.168.2.232
        netmask 255.255.255.0
```

Reiniciar o serviço:

```
$ /etc/init.d/networking restart
```

Obs.: E se der um erro de time-out, mudar no /etc/hosts para:

```
# Rede Privada
192.168.0.232 folsom-cc-private.domain.local folsom-cc-private
192.168.0.234 folsom-nc-private.domain.local folsom-nc-private
# Rede Publica
192.168.0.232 folsom-cc-public.domain.local folsom-cc-public
192.168.0.234 folsom-nc-public.domain.local folsom-nc-public
```

Novamente o comando:

```
$ glance image-create --name Cirros --is-public true --container-format bare --disk-format qcow2 < cirros-0.3.0-i386-disk.img
```

A saída do comando deve ser semelhante a esta:

```
+------------------+--------------------------------------+
| Property         | Value                                |
+------------------+--------------------------------------+
| checksum         | 90169ba6f09b5906a7f0755bd00bf2c3     |
| container_format | bare                                 |
| created_at       | 2013-11-01T16:36:22                  |
| deleted          | False                                |
| deleted_at       | None                                 |
| disk_format      | qcow2                                |
| id               | 7707e41f-fc29-480e-9d48-d931c64d3616 |
| is_public        | True                                 |
| min_disk         | 0                                    |
| min_ram          | 0                                    |
| name             | Cirros                               |
| owner            | 24689aeec80e4e4a88784e779cb4ddca     |
| protected        | False                                |
| size             | 9159168                              |
| status           | active                               |
| updated_at       | 2013-11-01T16:36:23                  |
+------------------+--------------------------------------+
```

Para se certificar de que o upload da imagem ocorreu de forma correta, rode o seguinte comando:

$ glance image-list

Que deve retornar algo semelhante a isso:

```
+--------------------------------------+--------+-------------+------------------+---------+--------+
| ID                                   | Name   | Disk Format | Container Format | Size    | Status |
+--------------------------------------+--------+-------------+------------------+---------+--------+
| 7707e41f-fc29-480e-9d48-d931c64d3616 | Cirros | qcow2       | bare             | 9159168 | active |
| b6c8098b-566d-4c2d-8d19-969a079f0840 | Cirros | qcow2       | bare             | 9159168 | active |
+--------------------------------------+--------+-------------+------------------+---------+--------+
```

Verificar se a *bridge* foi criada corretamente é o próximo passo (comando abaixo):

$ brctl show

Deve retornar:

```
bridge name     bridge id               STP enabled     interfaces
br100           8000.000c29f2f262       no              eth0
```

Nova:

Próximo componente a ser instalado e configurado é o Nova, que controla a maior parte dos serviços e recursos do OpenStack:

$ apt-get install nova-api nova-cert novnc nova-consoleauth nova-scheduler nova-novncproxy nova-network

Criar a base de dados do Nova:

```
$ mysql -u root -p
$ mysql> CREATE DATABASE nova;
$ mysql> GRANT ALL ON nova.* TO 'root'@'%' IDENTIFIED BY 'cloud';
$ mysql> quit;
```

Alterar a seção [filter:authtoken] do arquivo /etc/nova/api-paste.ini, para que *host*, porta, protocolo, usuário e senha fiquem de acordo com o necessário e utilizado na nuvem:

```
[filter:authtoken]
paste.filter_factory = keystone.middleware.auth_token:filter_factory
auth_host = folsom-cc-private
auth_port = 35357
auth_protocol = http
admin_tenant_name = service
admin_user = nova
admin_password = cloud
signing_dirname = /tmp/keystone-signing-nova
```

Modificar o arquivo /etc/nova/nova.conf para:

```
[DEFAULT]
#logdir=/var/log/nova
#state_path=/var/lib/nova
#lock_path=/run/lock/nova
#verbose=True
#api_paste_config=/etc/nova/api-paste.ini
#scheduler_driver=nova.scheduler.simple.SimpleScheduler
s3_host=folsom-cc-private
ec2_host=folsom-cc-private
```

```
ec2_dmz_host=folsom-cc-private
rabbit_host=folsom-cc-private
cc_host=folsom-cc-private
metadata_host=192.168.0.232
metadata_listen=0.0.0.0
nova_url=folsom-cc-private:8774/v1.1/
sql_connection=mysql://root:cloud@folsom-cc-private/nova
ec2_url=http://folsom-cc-private:8773/services/Cloud
root_helper=sudo nova-rootwrap /etc/nova/rootwrap.conf

# Auth
use_deprecated_auth=false
auth_strategy=keystone
keystone_ec2_url=http://folsom-cc-private:5000/v2.0/ec2tokens

# Imaging service
glance_api_servers=folsom-cc-private:9292
image_service=nova.image.glance.GlanceImageService

# Vnc configuration
novnc_enabled=true
novncproxy_base_url=http://folsom-cc-public:6080/vnc_auto.html
novncproxy_port=6080
vncserver_proxyclient_address=folsom-cc-private
vncserver_listen=0.0.0.0

# Network
network_manager=nova.network.manager.FlatDHCPManager
force_dhcp_release=True
dhcpbridge_flagfile=/etc/nova/nova.conf
firewall_driver=nova.virt.libvirt.firewall.IptablesFirewallDriver
my_ip=192.168.0.232
public_interface=br100
vlan_interface=eth0
```

```
flat_network_bridge=br100
flat_interface=eth0

# Compute #
compute_driver=libvirt.LibvirtDriver

# Cinder #
volume_api_class=nova.volume.cinder.API
osapi_volume_listen_port=5900
```

Sincronizar a base de dados Nova:

```
$ nova-manage db sync
```

Reiniciar todos serviços pertencentes ao componete Nova com o comando:

```
$ cd /etc/init.d/; for i in $(ls nova-*); do service $i restart; done
```

É necessário verificar se todos os serviços do Nova estão rodando:

```
$ nova-manage service list
```

O comando deve retornar como demonstrado abaixo, onde no campo Binary estam descritos os serviços instalados do Nova, em Host a qual pertencem, importante também verificar o campo Status deve estar enabled (ativo) e o campo State que deve aparecer : -) :

```
Binary           Host                                 Zone             Status     State Updated_At
nova-network     folsom-cc                            nova             enabled    :-)   2013-11-01 16:49:01
nova-consoleauth folsom-cc                            nova             enabled    :-)   2013-11-01 16:49:00
nova-cert        folsom-cc                            nova             enabled    :-)   2013-11-01 16:49:08
nova-scheduler   folsom-cc                            nova             enabled    :-)   2013-11-01 16:49:07
```

Cinder:

Instalação e configuração do componente *Cinder* responsável pelo armazenamento de blocos utilizado para a criação das instâncias:

```
$ apt-get install cinder-api cinder-scheduler cinder-volume
```

Criar base de dados para o *Cinder*:

```
$ mysql -u root -p
$ mysql> CREATE DATABASE cinder;
$ mysql> GRANT ALL ON cinder.* TO 'cinderUser'@'%' IDENTIFIED BY 'cinderPass';
$ mysql> quit;
```

Próximo passo configurar o arquivo /etc/cinder/api-paste.ini, configurar conforme necessário os itens protocol, *host*, port, usuário, senha de acesso:

```
[filter:authtoken]
paste.filter_factory = keystone.middleware.auth_token:filter_factory
service_protocol = http
service_host = folsom-cc-private
service_port = 5000
auth_host = folsom-cc-private
auth_port = 35357
auth_protocol = http
admin_tenant_name = services
admin_user = cinder
admin_password = cloud
```

Configurar também o arquivo /etc/cinder/cinder.conf modificando a conexão com o banco de dados:

```
[DEFAULT]
rootwrap_config = /etc/cinder/rootwrap.conf
sql_connection = mysql://root:cloud@folsom-cc-
private.domain.local/cinder
api_paste_confg = /etc/cinder/api-paste.ini
iscsi_helper = tgtadm
volume_name_template = volume-%s
volume_group = cinder-volumes
verbose = True
auth_strategy = keystone
state_path = /var/lib/cinder
volumes_dir = /var/lib/cinder/volumes
```

Sincronizar a base de dados do *cinder*:

```
$ cinder-manage db sync
```

Pode aparecer à mensagem a seguir, isso é normal porque a opção Verbose está configurada como *True* e quando essa opção está configurada ele lista os *debug*:

```
2013-11-01 09:59:02 2323 DEBUG cinder.utils [-] backend <module 'cinder
.db.sqlalchemy.migration' from '/usr/lib/python2.7/dist-packages/cinder
/db/sqlalchemy/migration.pyc'> __get_backend /usr/lib/python2.7/dist-pa
ckages/cinder/utils.py:481
```

Agora é preciso criar um volume group chamado cinder-volumes que será utilizado para armazenamento:

```
$ fdisk /dev/sda3
$ fdisk> n
$ fdisk> p
$ fdisk> 1
$ fdisk> ENTER
$ fdisk> ENTER
```

```
$ fdisk> t
$ fdisk> 8e
$ fdisk> w
```

A seguir criar o *physical* volume e o volume *group* pois o *Cinder* faz armazenamento de bloco assim ele será utilizado como um volume físico:

```
$ pvcreate /dev/sda3
```

Deve aparecer a seguinte mensagem:

```
Physical volume "/dev/sda3" successfully created
```

Criar um volume *group* (um grupo é a união volumes físicos, deve ser especificado o nome do grupo e os volumes físicos que farão parte dele):

```
$ vgcreate cinder-volumes /dev/sda3
```

Então deve aparecer a seguinte mensagem:

```
Volume group "cinder-volumes" successfully created
```

Horizon/Dashboard:

Instalação e configuração do componetes Horizon que fornece a usuários e administradores uma interface gráfica para acesso a ferramenta OpenStack:

```
$ apt-get install openstack-dashboard memcached
```

Incluir a linha ServerName no arquivo /etc/apache2/ports.conf:

```
ServerName folsom-cc-public
```

Opcionalmente é possível habilitar o SSL (Protocolo de segurança para comunicação na *Internet*), RewriteEngine(*software* que modifica a aparência da URL) e o VirtualHost (capacidade de hospedar mais de um *site* na mesma máquina) do SSL, com isso redirecionar todo tráfego HTTP para HTTPS:

```
$ a2enmod ssl
$ a2enmod rewrite
$ a2ensite default-ssl
$ service apache2 reload
```

Alterar o arquivo /etc/apache2/sites-available/default para fazer o redirecionamento HTTPS:

```
<VirtualHost *:80>
  RewriteEngine on
  RewriteCond %{SERVER_PORT} ^80$
  RewriteRule ^(.*)$ https://%{SERVER_NAME}$1 [L,R]
</VirtualHost>
```

Por padrão o Horizon irá abrir na url http://folsom-cc-public.domain.local, caso desejar alterar para que abra na raiz. Alterar no arquivo /etc/apache2/conf.d/openstack-dashboard.conf:

De:

```
WSGIScriptAlias /horizon /usr/share/openstack-dashboard/openstack_dashboard/wsgi/django.wsgi
WSGIDaemonProcess horizon user=www-data group=www-data processes=3 threads=10
```

Para:

```
WSGIScriptAlias / /usr/share/openstack-dashboard/openstack_dashboard/wsgi/django.wsgi
```

```
WSGIDaemonProcess horizon user=www-data group=www-data processes=3 threads=10
```

E no arquivo /etc/openstack-dashboard/local_settings.py alterar:

De:

```
LOGIN_URL='/horizon/auth/login/'
LOGIN_REDIRECT_URL='/horizon'
```

Para:

```
LOGIN_URL='/auth/login/'
LOGIN_REDIRECT_URL='/'
```

Depois de alterar esses arquivos reiniciar o apache e o memcached:

```
$ service apache2 restart
$ service memcached restart
```

Executar *reboot* e verificar se todos os serviços do Nova estão rodando corretamente utilizando o comando:

```
$ cd /etc/init.d/; for i in $(ls nova-*); do service $i status; done
```

- Após confirmado se os serviços estão rodando corretamente passe para a instalação e configuração do Compute Node:

B.2 Instalação e configuração do Compute Node:

O *Compute Node* forma um centro de recursos, pois fornece processamento, memória, rede e armazenamento para execução das instâncias no OpenStack.

Para começar com a instalação e configuração do *Compute Node* é necessário estar logado como root no terminal e primeiramente configurar o arquivo /etc/hosts:

```
127.0.0.1 localhost
192.168.0.234 folsom-nc
#Rede Privada
192.168.0.232 folsom-cc-private.domain.local folsom-cc-private
192.168.0.234 folsom-nc-private.domain.local folsom-nc-private
#Rede Publica
192.168.0.232 folsom-cc-public.domain.local folsom-cc-public
192.168.0.234 folsom-nc-public.domain.local folsom-nc-public
```

Deve se adicionar o repositório do Folsom:

```
$ apt-get install ubuntu-cloud-keyring
$ echo "deb http://ubuntu-cloud.archive.canonical.com/ubuntu precise-updates/folsom main" > /etc/apt/sources.list.d/folsom.list
```

Em seguida é necessário efetuar a atualização e *reboot*, usando os seguintes comandos:

```
$ apt-get update
$ apt-get upgrade
$ apt-get dist-upgrade
$ reboot
```

NTP:

No *Compute Node* também é necessária ainstalação do NTP para sincronização dos relógios:

```
$ apt-get install ntp
```

```
$ service ntp restart
```

VLAN e Bridge-Utils, instalar:

Pelo mesmo motive que foi feita a instalação de VLAN e bridge-utils no *Controller Node*, instalar nó *Compute Node*:

```
$ apt-get install vlan bridge-utils
```

Habilitar o IP_Forwarding:

```
$ sed -i 's/#net.ipv4.ip_forward=1/net.ipv4.ip_forward=1/g' /etc/sysctl.conf
```

KVM/QEMU:

É necessário configurar e instalar o KVM ou o QEMU por questões de virtualização, mas primeiramente é necessário verificar se o *hardware* tem suporte ao KVM:

```
$ apt-get install cpu-checker
$kvm-ok
```

Deve aparecer a seguinte mensagem:

```
INFO: /dev/kvm exists
KVM acceleration can be used
```

Se tiver suporte, instalar o KVM e o LibVirt:

```
$ apt-get install kvm libvirt-bin
```

Se não tiver suporte deve-se usar o QEMU instalando os seguintes pacotes:

```
apt-get install qemu libvirt-bin guestmount libguestfs-tools
```

Editar o cgroup_device_acl no arquivo /etc/libvirt/qemu.conf:

```
cgroup_device_acl = [
  "/dev/null", "/dev/full", "/dev/zero",
  "/dev/random", "/dev/urandom",
  "/dev/ptmx", "/dev/kvm", "/dev/kqemu",
  "/dev/rtc","/dev/hpet","/dev/net/tun"
]
```

Deletar a bridge virtual padrão que é criada:

```
$ virsh net-destroy default
$ virsh net-undefine default
```

Live Migration:

Será configurando o *Live Migration* que é um sistema de tranferência de máquina virtual ou aplicações entre diferentes m´quinas físicas. Foi instalado pensando em necessidades futuras:

Configurar no arquivo /etc/libvirt/libvirtd.conf os itens listen_tls (possibilidade de ouvir conexões tls segura na porta de IP público), listen_tcp (ouvir conexões criptografadas na porta TCP/IP pública), auth_tcp (autenticação tcp):

```
listen_tls = 0
listen_tcp = 1
auth_tcp = "none"
```

Alterar no arquivo /etc/init/libvirt-bin.conf:

```
env libvirtd_opts="-d -l"
```

E alterar no arquivo /etc/default/libvirt-bin:

```
libvirtd_opts="-d -l"
```

Reiniciar o libvirt para que as alterações sejam aplicadas:

```
$ service libvirt-bin restart
```

Se utilizar o KVM instalar também:

```
$ apt-get install nova-api-metadata nova-compute-kvm
```

Se utilizar o QEMU instalar também:

```
$ apt-get install nova-api-metadata nova-compute-qemu
```

Criar o arquivo local com as credenciais (~/.creds):

```
export OS_NO_CACHE=1
export OS_TENANT_NAME=admin
export OS_USERNAME=admin
export OS_PASSWORD=cloud
export OS_AUTH_URL="http://folsom-cc-private:5000/v2.0/"
```

Incluir as informações nas variáveis de ambiente do usuário:

```
$ source ~/.creds
$ echo " source ~/.creds" >> ~/.bash_profile
```

Nova-Network:

Instalação e configuração do nova-network serviço de rede do Nova:

```
$ apt-get install nova-network bridge-utils
```

Configurar a rede, no arquivo /etc/network/interfaces:

```
auto lo
iface lo inet loopback
auto br100
iface br100 inet static
        address 192.168.0.234
        netmask 255.255.255.0
        network 192.168.0.255
broadcast 192.168.0.255
gateway 192.168.0.1
dns-nameservers 8.8.8.8
bridge_ports eth0
bridge_stp off
bridge_maxwait 0
bridge_fd 0
auto eth1
iface eth1 inet static
                address 192.168.2.234
                netmask 255.255.255.0
```

Reiniciar a rede:

```
$ /etc/init.d/networking restart
```

Verificar se a bridge foi criada corretamente:

```
$ brctl show
```

Deve retornar algo semelhante a isso:

```
bridge name     bridge id               STP enabled     interfaces
br100           8000.000000000010       no              eth0
                                                        eth1
```

Alterar a sessão [filter:authtoken] no arquivo /etc/nova/api-paste.ini:

```
[filter:authtoken]
paste.filter_factory = keystone.middleware.auth_token:filter_factory
auth_host = folsom-cc-private
auth_port = 35357
auth_protocol = http
```

```
admin_tenant_name = service
admin_user = nova
admin_password = cloud
signing_dirname = /tmp/keystone-signing-nova
```

Editar o arquivo /etc/nova/nova-compute.conf:

Da seguinte forma se utilizar KVM:

```
[DEFAULT]
libvirt_type=kvm
```

Ou da seguinte se utilizar o QEMU:

```
[DEFAULT]
libvirt_type=qemu
```

A seguir editar o arquivo /etc/nova/nova.conf:

```
[DEFAULT]
logdir=/var/log/nova
state_path=/var/lib/nova
lock_path=/run/lock/nova
verbose=True
api_paste_config=/etc/nova/api-paste.ini
scheduler_driver=nova.scheduler.simple.SimpleScheduler
s3_host=folsom-cc-private
ec2_host=folsom-cc-private
ec2_dmz_host=folsom-cc-private
rabbit_host=folsom-cc-private
cc_host=folsom-cc-private
metadata_host=192.168.0.234
metadata_listen=0.0.0.0
nova_url=http://folsom-cc-private:8774/v1.1/
sql_connection=mysql://root:cloud@folsom-cc-private/nova
```

```
ec2_url=http://folsom-cc-private:8773/services/Cloud
root_helper=sudo nova-rootwrap /etc/nova/rootwrap.conf

# Auth
use_deprecated_auth=false
auth_strategy=keystone
keystone_ec2_url=http://folsom-cc-private:5000/v2.0/ec2tokens

# Imaging service
glance_api_servers=folsom-cc-private:9292
image_service=nova.image.glance.GlanceImageService

# Vnc configuration
novnc_enabled=true
novncproxy_base_url=http://folsom-cc-public:6080/vnc_auto.html
novncproxy_port=6080
vncserver_proxyclient_address=folsom-nc-private
vncserver_listen=0.0.0.0

# Network
network_manager=nova.network.manager.FlatDHCPManager
force_dhcp_release=True
dhcpbridge=/usr/bin/nova-dhcpbridge
dhcpbridge_flagfile=/etc/nova/nova.conf
firewall_driver=nova.virt.libvirt.firewall.IptablesFirewallDriver
my_ip=192.168.0.234
public_interface=br100
vlan_interface=eth0
flat_network_bridge=br100
flat_interface=eth0

#Compute
compute_driver=libvirt.LibvirtDriver
```

```
#Cinder
volume_api_class=nova.volume.cinder.API
osapi_volume_listen_port=5900
```

Sincronizar a base de dados do Nova:

```
$ nova-manage db sync
```

Reiniciar os serviços nova-*:

```
$ cd /etc/init.d/; for i in $(ls nova-*); do service $i restart; done
```

Verificar se os serviços estão rodando e se o *Compute* foi incluído a lista:

```
$ nova-manage service list
```

A saída deve ser algo semelhante a isso:

```
Binary           Host                                 Zone             Status     State Updated_At
nova-cert        folsom-cc                            nova             enabled    :-)   2013-11-01 23:46:09
nova-consoleauth folsom-cc                            nova             enabled    :-)   2013-11-01 23:46:11
nova-scheduler   folsom-cc                            nova             enabled    :-)   2013-11-01 23:46:10
nova-network     folsom-cc                            nova             enabled    :-)   2013-11-01 23:46:10
nova-compute     folsom-nc                            nova             enabled    :-)   2013-11-01 23:46:07
nova-network     folsom-nc                            nova             enabled    :-)   2013-11-01 23:46:07
```

Armazenamento:

Para o armazenamento foi utilizada a configuração *default*, criando uma partição para armazenar as instâncias:

```
$ fdisk /dev/sda2
$ fdisk> n
$ fdisk> p
$ fdisk> enter
$ fdisk> enter
$ fdisk> enter
$ fdisk> w
```

Formatar em ext4:

```
$ mkfs.ext4 /dev/sda2
```

Incluir no /etc/fstab e montar a partição:

```
$ echo "/dev/sda2 /var/lib/nova/instances/ ext4 errors=remount-ro
0 1" >> /etc/fstab
$ mount -a
```

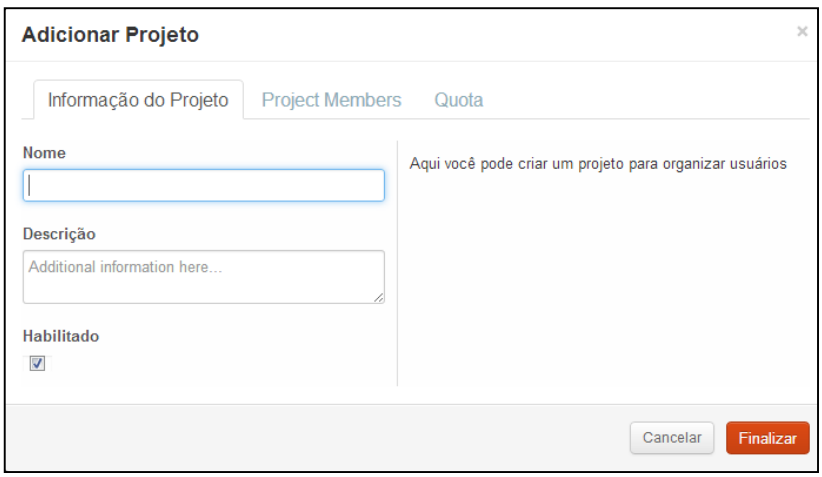
Alterar as permissões para o usuário e grupo nova:

```
$ chown -R nova.nova /var/lib/nova/instances
```

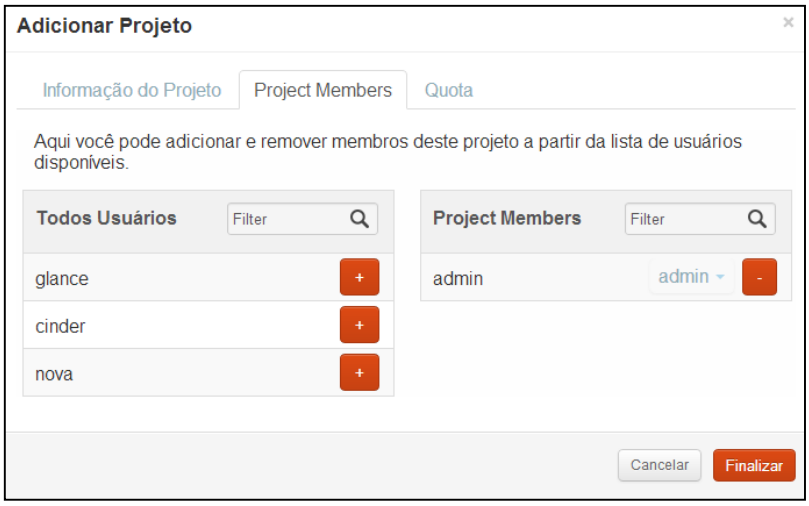
Painel Horizon:

O próximo passo é criar um projeto usando o painel Horizon. Acessar o painel a partir do IP http://192.168.0.232 e logar com usuário admin e senha.

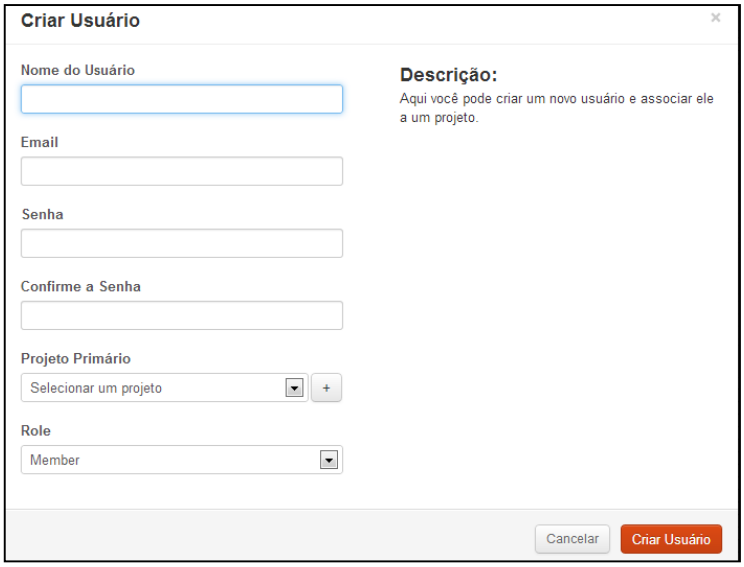
No item Projetos, Criar projeto. Na guia Informação do Projeto informar o nome do novo projeto.



Em *Project Members*, selecionar o usuário admin com permissão de admin:



Em Quota não é necessário configurar neste ponto, apenas Finalizar. Projeto Criado. Agora no item Usuários, Criar Usuário. Dando um nome ao usuário, *email*, senha e adicionando ele ao projeto anteriormente criado como Member:



A seguir é necessário criar uma rede e associa-la ao projeto criado, para isso é necessário o ID do Projeto, que pode ser conseguido através do próprio painel ou utilizando o comando:

```
$ keystone tenant-list
```

Que retornará os campos ID, nome do projeto e se está ativo:

```
+----------------------------------+-----------+---------+
|                id                |    name   | enabled |
+----------------------------------+-----------+---------+
| 24689aeec80e4e4a88784e779cb4ddca |   admin   |   True  |
| 6e43c203354042f18a0d88ef378609d2 |  service  |   True  |
| 839353ca87c54cc68116a8f4793c1cff | Projeto01 |   True  |
+----------------------------------+-----------+---------+
```

Agora pode ser criada uma Rede e associa-la ao Projeto:

$ nova-manage network create --label=Projeto01Network --fixed_range_v4=172.16.14.0/27 --fixed_cidr=172.16.14.0/27 --network_size=32 --bridge=br100 --project_id=839353ca87c54cc68116a8f4793c1cff --num_networks=1 --dns1=8.8.8.8 --dns2=8.8.4.4 --multi_host=T

Verificar se a rede foi criada corretamente:

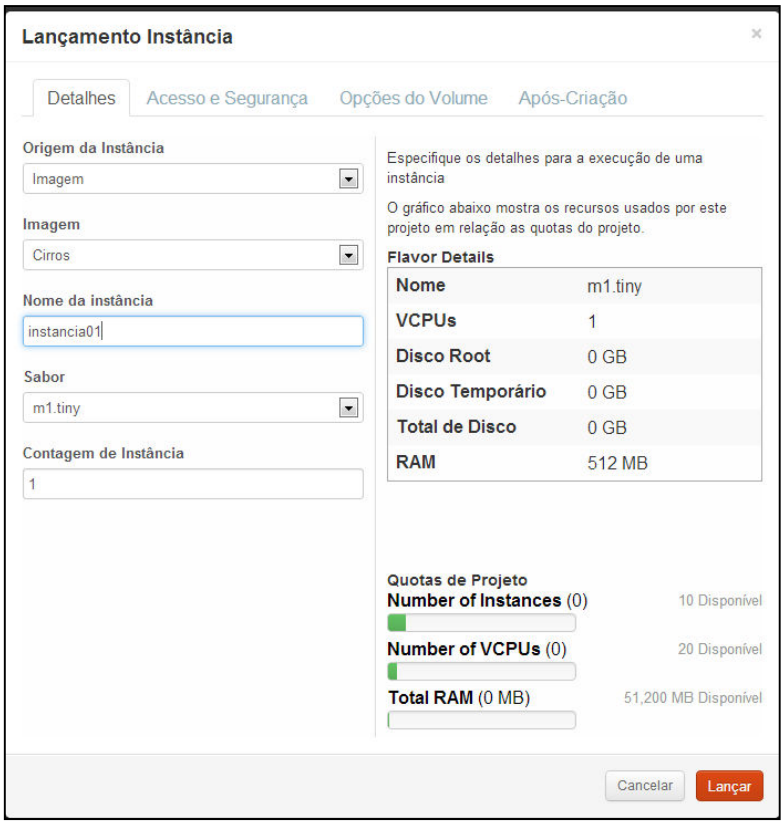
$ nova-manage network list

Que retorna valores semelhantes á esses (em linhas), essas informações dizem respeito a rede criada e associada ao projeto, assim é possível verificar se foi criada e associada corretamente ao projeto:

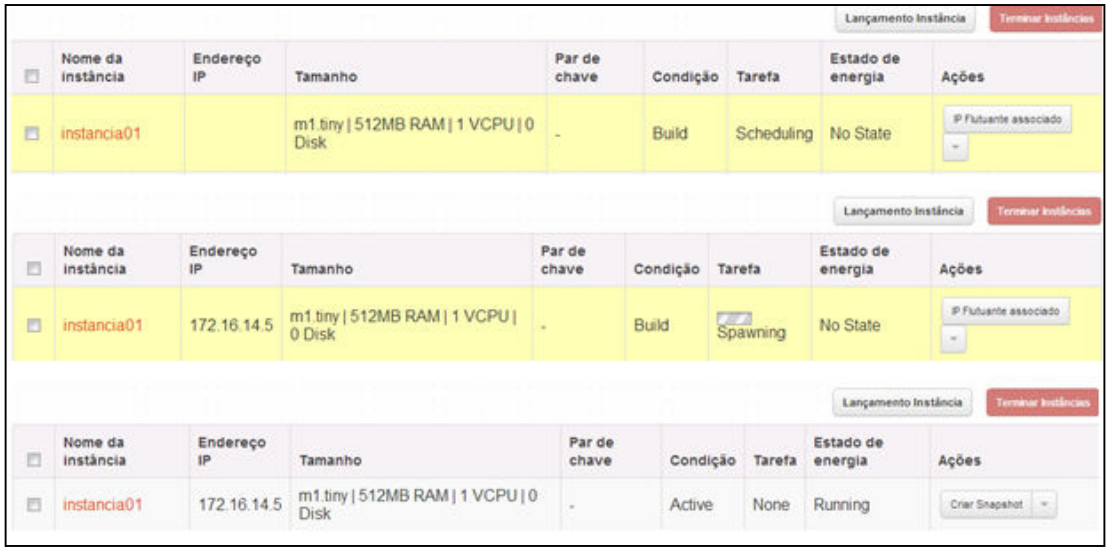
id 1
IPv4 172.16.14.0/27
IPv6 None
start address 172.16.14.2
DNS1 8.8.8.8
DNS2 8.8.4.4
VlanID None
project 839353ca87c54cc68116a8f4793c1cff
uuid 760beef2-9b70-4e0-9353-8c0142ba0065

O próximo passo é logar no painel (com o usuário anteriormente criado, ao invés de logar como admin) para criar uma instância.

Clicar no item Instâncias, Lançamento Instância. Escolher a Origem da Instância no caso Imagem, em Imagem escolher uma das que estiverem disponíveis e dar um nome a instância. No restante não têm necessidade de configurar ainda, por fim Lançar:



A instância é criada em poucos minutos, pode-se dizer em três passos, primeiro carrega a instância, obtém um IP e por fim se torna Ativa:



Depois disso já é possível acessar a máquina virtual por SSH estando no Compute Node, abaixo é possível verificar que foi realmente permitido o acesso SSH e com ifconfig verificar a IP da rede da máquina virtual:

```
root@folsom-nc:/home/cloud# ssh cirros@172.16.14.5
The authenticity of host '172.16.14.5 (172.16.14.5)' can't be established.
RSA key fingerprint is 59:9c:df:d8:14:4d:ae:b7:91:f1:4a:74:a7:fb:ec:03.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added '172.16.14.5' (RSA) to the list of known hosts.
cirros@172.16.14.5's password:
$ ifconfig
eth0      Link encap:Ethernet  HWaddr FA:16:3E:13:79:ED
          inet addr:172.16.14.5  Bcast:172.16.14.31  Mask:255.255.255.224
          inet6 addr: fe80::f816:3eff:fe13:79ed/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:143 errors:0 dropped:0 overruns:0 frame:0
          TX packets:82 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:20334 (19.8 KiB)  TX bytes:10838 (10.5 KiB)
          Interrupt:11

lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          inet6 addr: ::1/128 Scope:Host
          UP LOOPBACK RUNNING  MTU:16436  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:0 (0.0 B)  TX bytes:0 (0.0 B)

$
```

APÊNDICE C: RESUMO PARA O SAPS

ANÁLISE E COMPARAÇÃO DE FERRAMENTAS *OPEN SOURCE* DE COMPUTAÇÃO EM NUVEM PARA O MODELO DE SERVIÇO IAAS

A computação em nuvem é uma área que atrai grande interesse na atualidade, pois possibilita às empresas, independentemente do tamanho da mesma e de suas necessidades, ter acesso à tecnologia. A motivação para realização deste trabalho, é encontrar uma ferramenta que possa ter bons resultados, sendo utilizada em estações de trabalho. O trabalho tem o intuito de esclarecer ao leitor a utilidade de uma nuvem privada, e auxiliá-lo na escolha da melhor ferramenta que pode vir a ser utilizada em seu ambiente de trabalho, sem a necessidade de grandes investimentos. O objetivo é realizar a comparação de ferramentas *open source* de computação em nuvem para o modelo de serviço IaaS. Por isso, ferramentas serão testadas para que se possa escolher a ferramenta que melhor se adeque com o uso de estações de trabalho. Uma nuvem privada é exclusiva a um único usuário ou empresa, somente a empresa contratante do serviço e quem administra a nuvem pode ter acesso a mesma, tudo para garantir a segurança das informações, e o IaaS (*Infrastructure as a Service* ou Infraestrutura como um serviço) é um modelo de serviço que fornece ao usuário a infraestrutura de *hardware*, sem que o mesmo necessite gerenciar ou realizar a manutenção, somente terá responsabilidade pelas aplicações que irão executar nela. Utilizou-se como metodologia métodos como Abordagem indutiva e também qualitativa, os Procedimentos de pesquisa exploratória, bibliográfica e estudo do caso e a Técnica de observação. Entre as ferramentas existentes pode-se citar o OpenNebula, OpenStack, CloudStack, Eucalyptus, Ubuntu Entreprise Cloud, Abiquo, OpenQRM e ConVirt, sendo que serão escolhidas quatro ferramentas para implantar. Como passo importante para realização da comparação será desenvolvida uma tabela que possibilita a visualização das diferenças e semelhanças através de características pesquisadas entre todas as ferramentas. No presente trabalho será utilizado como ambiente de testes, computadores desktop, todos em uma mesma rede, e após escolhidas as ferramentas, serão instaladas quatro delas nesses computadores, além de usar outros computadores como nós para que se possam ser realizados os testes, os testes deverão ser baseados nas características a serem avaliadas nas ferramentas, para que se possa obter um resultado que mostre a eficácia dos mesmos.

Palavras-chaves: Computação em nuvem, Nuvem privada, Infraestrutura como um serviço.

Referências

ABIQUO. **ABIQUO.** Disponível em: <http://www.abiquo.com/overview-technical/>. Acesso em 24 de julho de 2013.

ANTONOPOULOS, Nick, GILLAM, Lee. 2010. **Cloud Computing: Principles, Systems and Applications**. Londres: Springer-Verlag.

CLOUDSTACK**. What is CloudStack?.** Disponível em < http://cloudstack.apache.org/ >. Acesso em 31 de julho de 2013.

CONVIRTURE. **ConVirt**. Disponível em: <http://www.convirture.com/solutions_cloudcomputing.php>. Acessado em 30 de jul de 2013.

EUCALYPTUS. **Eucalyptus**. Disponível em < http://www.eucalyptus.com/ >. Acesso em 24 de julho de 2013.

HENTGES, Eduardo Luís; THOMÉ, Bruna Roberta, GRIEBLER, Dalvan. 2013. **Análise e Comparação de Ferramentas Open Source de Computação Em Nuvem para o Modelo de Serviço Iaas** - Trabalho de Conclusão do Curso Superior de Tecnologias em Redes de Computadores. Tecnologia em Redes de Computadores. Três de Maio: SETREM.

MARKS, Eric A., LOZANO, Bob. 2010. **Executive's Guide to Cloud Computing.** New Jersey: Published by John Wiley & Sons, Inc..

OPENNEBULA. **About the OpenNebula.org Project.** Disponível em < http://OpenNebula.org/about:about >. Acesso em 22 de julho de 2013.

OPENSTACK. **About OpenStack.** Disponível em < http://www.openstack.org/software/ >. Acesso em 30 de julho de 2013.

OPENQRM. **Welcome to the openQRM Community Resources.** Disponível em < http://www.openqrm-enterprise.com/community/ >. Acesso em 24 de julho de 2013.

UBUNTU. **Ubunte Enterprise Cloud**. Disponível em < http://www.ubuntu.com/cloud >. Acesso em 24 de julho de 2013.

APÊNDICE D: ARTIGO ERRC 2013

Computação em Nuvem: Análise Comparativa de Ferramentas Open Source para IaaS

Bruna Thomé, Eduardo Hentges
Curso Superior de Tecnologia em Redes de Computadores
Faculdade Três de Maio (SETREM)
Três de Maio – RS – Brasil
{thome.bru, eduhentges}@gmail.com

Dalvan Griebler
Programa de Pós-Graduação em Ciência da Computação
Pontifícia Universidade Católica do Rio Grande do Sul
Porto Alegre – RS – Brasil
dalvan.griebler@acad.pucrs.br

***Abstract*— Este artigo tem por objetivo estudar, apresentar e comparar as principais ferramentas open source de computação em nuvem. O conceito de computação em nuvem está cada vez mais presente nas redes de computadores. A dificuldade não está apenas em implantar uma nuvem, mas também em escolher a ferramenta mais apropriada. Assim, este trabalho buscou estudar as seguintes ferramentas: Eucalyptus, OpenNebula, OpenQRM, OpenStack, CloudStack Ubuntu Enterprise Cloud, Abiquo, Convirt, Apache Virtual Lab e Nimbus. Para estas, foram consideradas as características, funcionalidades e formas de operação, evidenciando o cenário mais indicado para cada uma delas.**

Keywords*— *Computação em Nuvem, Ferramentas Open Source, Modelo IaaS;

Introdução

A Computação em nuvem (CN) possibilita acessar recursos computacionais (por exemplo, servidores, armazenamento, redes, serviços e aplicações) de maneira prática e sob demanda, rapidamente e que podem ser liberados para o usuário sem qualquer envolvimento gerencial. [1]. Isso pode ser muito importante para agilizar o desenvolvimento do trabalho, reduzir custos, facilitar o emprego de recursos de alto processamento, evitar gastos com manutenção e licenças de *software*.

As nuvens podem ser caracterizadas em diferentes tipos (pública, privada e híbrida) e diferentes modelos de serviços (IaaS - *Infrastructure as a Service*, PaaS - *Plataform as a Service* e SaaS - *Software as a Service*) [2,3]. Neste trabalho, o escopo são as ferramentas *open source* para administração de nuvem que suportam o modelo IaaS.

A dificuldade não está somente em implantar uma nuvem, mas também em escolher a ferramenta mais apropriada para o projeto de redes. Neste artigo, o objetivo écaracterizar, estudar e comparar as principais ferramentas, evidenciando o cenário mais indicado para cada uma delas. Para isso, o artigo apresenta inicialmente os trabalhos relacionados na Seção II. Na Seção III são estudadas as ferramentas de CN e na Seção IV é efetuada uma análise comparativa das ferramentas.

II. TRABALHOS RELACIONADOS

Nesta seção o objetivo é expor alguns trabalhos que apresentam uma relação a este e o que há de diferente em comparação ao que já existe na literatura. Alguns deles fazem um comparativo de características. No entanto, outros procuram implantar uma ou duas ferramentas a fim de comparar as funcionalidades em um cenário específico.

No trabalho [4], não é criado nenhum ambiente de teste e apenas são estudadas as características e a arquitetura das seguintes ferramentas: Xen *Cloud Platform*, Nimbus, OpenNebula, Eucalyptus, TPlataform, Apache *Virtual Computing Lab* e Enomaly *Platform Computing Elastic*. A análise comparativa é feita através de uma tabela, onde é descrita a ferramenta, o modelo de serviço, suas principais características e exemplos de quem as utiliza. Os autores concluíram que existe a necessidade de padronização das plataformas atuais, referente à interface, negociação, acesso por meio de *Web Services*. Isso por que as nuvens têm diferentes níveis de abstração.

O trabalho [5] trata da comparação de ferramentas, com o objetivo de descobrir se os usuários necessitavam de mais ferramentas de acesso. Caso necessário, verificar qual delas deveria ser utilizada. Para isso, os autores analisaram a comunidade de usuários da empresa FutureGrid e assim fizeram seus registros. As ferramentas escolhidas como parte do processo de aplicação do projeto foram Nimbus e Eucalyptus. Os autores concluíram que são fornecidas evidências de que existe a oferta de muitas ferramentas e que é necessário o usuário dizer qual é a melhor para ele.

Em [6], são feitos testes utilizando a ferramenta *Open* Cirrus. Para efetuar a avaliação de desempenho, foram utilizados o PlanetLab e o Emulab, para simular a utilização de usuários distribuídos e a utilização de aplicativos em nuvem. O resultado da pesquisa mostrou que o desempenho da transferência de dados em uma nuvem pode variar, dependendo de quantos usuários diferentes estão utilizando o mesmo serviço. Além disso, as variações podem ser atribuídas as características de rede entre a nuvem e usuários. Sendo assim, a distância entre a nuvem e os usuários é de grande determinação para o desempenho.

No trabalho [7], foram utilizadas as ferramentas OpenQRM e Eucalyptus com o intuito de verificar qual delas é a melhor. Para a realização dos testes é utilizado um ambiente isolado de 6 computadores *desktop*, com o sistema operacional Ubuntu

10.04 e uma rede com acesso à *Internet*. Como testes, foram realizadas tarefas em cada um dos componentes das ferramentas individualmente, no qual se executavam tarefas como envio de pacotes ICMP e transferência de arquivos e era realizada alguma falha proposital para ver o resultado. Assim, os melhores resultados foram do OpenQRM.

A pesquisa de [8] afirma que existe uma grande quantidade de características que devem ser levados em consideração para avaliar a CN. Assim, uma série de ferramentas são consideradas como: OpenNebula, Eucalyptus, Ubuntu *Enterprise Cloud*, OpenQRM, Abiquo, Red Hat *Cloud Foundations*, Edition One, OpenStack, Nimbus, mOSAIC. As características são agrupadas em: armazenamento, virtualização, gestão, rede, segurança e apoio. O resultado é que com base nas características pode ser efetuada a escolha da ferramenta mais apropriada, a que se adeque as necessidades da organização.

Os trabalhos [4], [5] e [8] apresentam objetivos em comum a esta pesquisa, o de comparar ferramentas de CN *open source*. No entanto, o presente trabalho compara um conjunto de características maior e traz uma atualização da situação atual das ferramentas que já foram estudadas nos outros trabalhos. Em relação aos trabalhos [4] e [5] são estudadas seis ferramentas diferentes e no trabalho [8], apenas três diferentes são estudadas. Porém o conjunto de características comparadas no presente trabalho é bem mais amplo, contribuindo também com a comparação de: interface, gerenciamento de energia, balanceamento de carga, integração, segurança e monitoramento. Isso deixa claro, que embora exista uma semelhança, várias contribuições podem vir a surgir através de uma comparação mais ampla e detalhada.

As próximas direções desta pesquisa se encaminham no sentido dos trabalhos [6] e [7], pois no futuro, a ideia é que estas ferramentas também sejam avaliadas em um ambiente controlado, sendo possível identificar o comportamento delas e verificar se são coerentes com o que a literatura nos apresenta.

III FERRAMENTAS DE COMPUTAÇÃO EM NUVEM

Nesta seção serão apresentadas as ferramentas *Open Source* de computação em nuvem para o modelo de serviço IaaS. Foram selecionadas ferramentas que oferecessem este modelo de serviço.

Eucalyptus

O Eucalyptus é indicado para computação em nuvem em ambientes de computação empresarial corporativa, pois possibilita oferecer aos usuários acesso as ferramentas utilizadas pela empresa. Tendo como arquitetura cinco componentes (*Cloud Controller*, Walrus, *Cluster Controller*, *Storage Controller* e *Node Controller*) básicos, responsáveis pelo seu funcionamento. Além disso, possibilita o uso de diferentes servidores para implantar os componentes e facilitar a configuração [9].

OpenNebula

O OpenNebula foi desenvolvido para uma gestão mais eficiente e escalável de máquinas virtuais em infraestruturas distribuídas. Suas características são voltadas para atender aos requisitos de empresas que utilizavam a ferramenta em versões anteriores. Sua arquitetura é composta por um *host* responsável pela administração da nuvem e os outros *hosts* são responsáveis pela virtualização das máquinas virtuais [10].

OpenStack

OpenStack permite criar nuvens públicas e privadas. Através de uma interface, o administrador pode gerenciar a capacidade de computação, armazenamento e recursos de rede presentes no *datacenter*. Entre seus componentes estão o OpenStack *Compute* (Nova), OpenStack *Object Storage* (*Swift*), OpenStack *Image Service* (*Glance*), Painel de ferramentas (*Horizon*), Rede (*Quantum*), *Storage Block* (*Cinder*) e Identificação (*Keystone*) [5,11].

CloudStack

O CloudStack foi desenvolvido para implantar e gerenciar grandes redes de máquinas virtuais, pois possui escalabilidade e alta disponibilidade de infraestrutura. Permite a criação de nuvens privadas, híbridas e públicas que podem fornecer infraestrutura como um serviço para os usuários. A arquitetura é composta pelo armazenamento primário, o *cluster,* Pod (grupo de *clusters*) e armazenamento secundário [12].

OpenQRM

OpenQRM é uma ferramenta que gerencia virtualização, armazenamento, a rede e toda a infraestrutura de TI a partir de um console. Permite criação de nuvens privadas com alta disponibilidade e também funciona de maneira gerente-agente. Para isso, o controlador de nuvem é o gerente e os recursos que são integrados a ele são os agentes. Neste caso, a estrutura apresenta o gerente, o *storage* e o nós (recursos) [13,7].

Ubuntu Enterprise Cloud

O Ubuntu *Enterprise Cloud* é baseado na ferramenta Eucalyptus. Devido a isso, apresenta os mesmos componentes (*Cloud Controller*, Walrus, *Cluster Controller*, *Storage Controller* e *Node Controller*). Também permite criar um perfil de instalação mínima para gerenciar as máquinas físicas e as virtuais, além de monitorar os componentes da nuvem [14].

Abiquo

Abiquo visa criar nuvens privadas baseadas em uma infraestrutura já existente ou controlar o uso de serviços em nuvem pública. Fornece *logs* para analisar o que e para que estão sendo utilizados os recursos e possui um mecanismo de preços que atribui um valor a qualquer recurso (CPU, RAM, armazenamento). O Abiquo é constituído pelo Gerenciador de rede, *cluster*, servidor Abiquo, rede de armazenamento, Abiquo serviços remotos e um servidor de armazenamento [15].

Convirt

O ConVirt possibilita centralizar o gerenciamento através de *datacenters* virtuais. Ele também é capaz de monitorar os recursos do servidor e dos clientes da máquina virtual, possibilitando o controle da carga exercida sobre o servidor. A sua arquitetura é composta pelo *Datacenter-wide*, *Universal web Access* e *Agent-less* [16].

Apache Virtual Computing Lab (Apache VCL)

Apache VCL oferece como ambiente uma máquina virtual ou até mesmo um *cluster* de servidores físicos. É utilizado para acesso remoto a partir da *Internet* de maneira dinâmica, utilizando-se de reservas de recursos computacionais. Ele tem sua arquitetura formada por portal *web*, banco de dados, nós de gestão e nós de computação [17].

Nimbus

O Nimbus possibilita a construção de nuvens privadas, implantando *clusters* virtuais autoconfiguráveis. Sua arquitetura é composta pelo: *workspace* de serviços (que permite ao cliente implantar e gerenciar grupos definidos de VMs) gerenciador de recursos**,** (que realiza a implantação de contratos de locação de VM), *workspace pilot* (se estende a gestores de recursos locais), IaaS *gateway* (permite que um cliente utilize outra infraestrutura como serviço) e *workspace client* (fornece a funcionalidade total do serviço) [18].

IV ANÁLISE COMPARATIVA

Nesta seção é realizada uma análise comparativa das ferramentas estudadas na Seção III. Esta comparação é feita usando tabelas que elencam as características de: Interface, Gerenciamento de energia, Balanceamento de carga, Rede, Armazenamento, Monitoramento, Integração, Virtualização, Segurança, Escalabilidade e Tolerância a falhas.

Na Tabela I são comparadas as características de Interface, Gerenciamento de energia e Balanceamento. A interface de acesso pode ser de duas formas: através de SSH (*Secure Shell*) que é uma conexão segura entre o cliente e o servidor ou através de uma página *web,* por HTTP (*Hyper-text Transfer Protocol*). O gerenciamento de energia tem o objetivo de reduzir os custos com energia elétrica. O Eucalyptus realiza isso através da suspensão das máquinas que não estão em uso. A mesma coisa é realizada pelo OpenNebula utilizando o sistema CLUES (*Cluster Energy Saving*) e pelo UEC através do UEC Power Management. Já o OpenStack oferece extensões que somente funcionam junto a processadores Intel Xeon. O CloudStack e Apache VCL colocam os *hosts* e recursos em modo *standby* quando não estão sendo usados. O OpenQRM busca por recursos não utilizados ou em baixo uso. E o Nimbus move máquinas virtuais para outros servidores. O Abiquo e o Convirt não apresentam esta característica em sua descrição.

TABELA I. COMPARAÇÃO DAS FERRAMENTAS I.

Ferramenta	Interface	Gerenciamento de Energia	Balanceamento de carga
Eucalyptus	SSH e *WEB*	Possui	*Elastic Load Balancer*
OpenNebula	SSH e *WEB* (Sustone GUI)	CLUES	Possui
OpenStack	SSH e *WEB* (*Horizon*)	*Power Management*	*Quantum Network Load Balancing*
Cloud Stack	SSH e *WEB*	Possui	Citrix NetScaler
OpenQRM	SSH e *WEB*	Possui	Possui
UEC	SSH e *WEB*	UEC *Power Management*	Não
Abiquo	SSH e *WEB*	Não	Sim
Convirt	SSH e *WEB*	Não	Não
Nimbus	SSH e *WEB*	Possui	Não
Apache VCL	SSH e *WEB*	Possui	Não

O balanceamento de carga é uma maneira eficiente de fazer a divisão das tarefas e melhor aproveitar os recursos computacionais. O Eucalyptus utiliza o *Elastic Load Balancer* que distribui automaticamente o tráfego de entrada das aplicações entre os nós do Cluster. No OpenNebulaas máquinas virtuais em execução, são divididas entre os nós operantes na nuvem. O OpenStack, utiliza o *Quantum Network Load Balancing* para dividir a carga de processos entre os nós. O CloudStack utiliza-se do Citrix NetScaler para dividir as tarefas entre os nós e o OpenQRM faz o balanceamento dos recursos do *cluster* para execução dos processos. Já o Abiquo divide entre os nós as conexões e as demais ferramentas não possuíam informações na literatura sobre esta característica.

TABELA II. COMPARAÇÃO DAS FERRAMENTAS II

Ferramenta	Rede	Armazenamento	Monitoramento
Eucalyptus	*Bridge* e VLAN	AoE, iSCSI e NFS	Nagios
OpenNebula	*Bridge*, VLAN e *Open* Vswitch	NFS, ISCSI, LVM	OpenNebula Sunstone
OpenStack	VLAN e *Open* Vswitch	AoE, iSCSI e NFS	OpenStack Clanavi
Cloud Stack	VLAN	ISCSI e NFS	*Traffic Sentinel*
OpenQRM	*Bridge* e VLAN	NFS, iSCSI, AoE e LVM	openqrm-monitord
UEC	*Bridge* e VLAN	iSCSI e AoE	UEC *Monitor*
Abiquo	VLAN	NFS, iSCSI, LVM	Abiquo *Monitor*
Convirt	VLAN	NFS, iSCSI e LVM	Convirt *Monitor*
Nimbus	VLAN	AOE, iSCSI e NFS	Nagios
Apache VCL	VLAN	iSCSI	Não

A Tabela II compara as funcionalidades de Rede, *Armazenamento* e Monitoramento. Na funcionalidade de rede são demonstrados os métodos de conexão utilizados entre os componentes de uma nuvem. Podem ser citadas a VLAN(permite dividir uma rede física em diversas redes lógicas), o *Bridge* (permite conectar duas ou mais redes distintas) e o *Open* vSwitch (cria um *switch* virtual que encaminha o tráfego de máquinas virtuais dentro de um mesmo *host*).

O Armazenamento pode ser realizado utilizando de diferentes formas. Como por exemplo, o iSCSI (*Internet Small Computer System Interface*) que é um protocolo de transporte de comandos SCSI, é usado onde dados são armazenados em diversos *hosts* de uma rede. O AoE (ATA *Over Ethernet*) é um protocolo de rede para acesso a dispositivos de armazenamento SATA através da rede. O NFS (*Network File System*) é um sistema de arquivos em que diretórios são compartilhados entre os computadores de uma rede. E o LVM (*Logical Volume Management*) é usado para criar um grande disco virtual que pode conter mais de um dispositivo de armazenamento.

Eucalyptus e Nimbus utilizam o Nagios para monitorar recursos como CPU, memória, HD e VMs. No entanto, as demais ferramentas se utilizam de sistema próprio, com exceção o Apache VCL que não possuiu relato sobre isso.

A Tabela III mostra a comparação da Integração, Virtualização e Segurança. Na Integração são descritas se as ferramentas possuem integração com algum outro serviço. As ferramentas Apache VCL e Abiquo são as únicas que não permitem integração com a Amazon. O CloudStack também permite integração com o CloudBridge, uma plataforma integrada que conecta aplicativos e melhora a utilização da largura de banda em nuvem pública e redes privadas. O

OpenQRM permite integração também com as ferramentas UEC e Eucalyptus. O Abiquo possibilita integração com Cisco UCS, o que facilita a mudança para o modelo de serviço IaaS. Já o Nimbus também permite integração com o Cumulus, um sistema de armazenamento em nuvem.

TABELA III. COMPARAÇÃO DAS FERRAMENTAS III

Ferramenta	Integração	Virtualização	Segurança
Eucalyptus	EC2, EBS, AMI, S3, IAM	Xen, KVM e Vmware ESXi	Autenticação, CUG e *Active Directory*
OpenNebula	EC2	Xen, KVM e Vmware	Autenticação e CUG
OpenStack	EC2 e S3	XenServer, KVM e Hyper-V	*Keystone*, LDAP, e métodos externos
Cloud Stack	CloudBridge e EC2	Xen, KVM e Vmware ESXi	Autenticação e CUG
OpenQRM	UEC, EC2 e Eucalyptus	Vmware ESX, Xen, KVM e XenServer	Autenticação, CUG e LDAP
UEC	EC2	KVM	Autenticação e CUG
Abiquo	Cisco UCS	VMware ESXi, Hyper-V, XenServer, Xen, KVM	Autenticação, CUG e LDAP
Convirt	EC2	Xen e KVM	Não
Nimbus	EC2, S3, Cumulus	Xen e KVM	Autenticação e CUG
Apache VCL	Não	Vmware, KVM	Autenticação LDAP

As únicas ferramentas que não oferecem suporte a virtualização Xen (tecnologia que pode ser atrelada diretamente ao *hardware)* são Apache VCL e o UEC. O Eucalyptus e CloudStack também podem oferecer virtualização com VMware ESXi que não necessita de sistema operacional e pode ser integrado diretamente aos servidores. O OpenNebula e o Apache VCL utilizam o VMware que permite a instalação de um sistema operacional dentro de outro em execução simultânea. Já o OpenStack, OpenQRM e Abiquo utilizam o Xen Server, possibilitando que várias máquinas virtuais rodem em uma máquina física. OpenStack e Abiquo podem utilizar também o Hyper-V, que fornece infraestrutura de *software* e ferramentas para gerenciar ambientes de virtualização de servidores.

Em relação a segurança, a maioria das ferramentas permitem autenticação e Controle por Usuários e Grupos (CUG). A ferramenta Eucalyptus também possibilita a integração com *Active Directory* (AD). O OpenStack, OpenQRM, Abiquo e o Apache VCL permitem usar autenticação por LDAP, que assim como o AD, realiza autenticação dos usuários. O OpenStack ainda pode utilizar do componente *Keystone*, utilizado para autenticação.

Se uma das necessidades da nuvem é o gerenciamento de energia, as ferramentas Abiquo e Convirt não são indicadas, pois as mesmas não realizam tal ação. Se a prioridade for o balanceamento de carga, dentre as ferramentas avaliadas somente o Eucalyptus, OpenNebula, OpenStack, CloudStack, OpenQRM e Abiquo realizam esta tarefa, logo, estas são mais indicadas. Em relação à rede, se houver a necessidade de conectar diferentes redes, são indicadas as ferramentas Eucalyptus, OpenNebula, OpenQRM e Ubuntu Enterprise Cloud, por possuírem o método de conexão *bridge* e VLAN. Se for preciso uma maior variedade de métodos de armazenamento, são indicados o Eucalyptus, OpenNebula, OpenStack, OpenQRM, Abiquo, Convirt e Nimbus.

Quando é indispensável o monitoramento tanto do *hardware* que compõem a nuvem, quanto das máquinas virtuais que operam nela, não é indicada a utilização do Apache VCL, pois este não apresenta nenhuma alternativa de monitoramento, diferente das outras ferramentas. Embora todas as ferramentas citadas na Tabela III ofereçam integração, exceto o Apache VCL. Ao se deparar com um cenário em que se tem como prioridade a integração com diferentes nuvens, as ferramentas Eucalyptus, Nimbus e OpenQRM parecem ser mais apropriadas, por possuírem mais opções de integração.

Ao se deparar com um ambiente em que existe grande heterogeneidade arquitetural, ter diversas ferramentas de virtualização torna-se uma vantagem. Nestes casos, é mais indicado a utilização das ferramentas OpenQRM e Abiquo, por possibilitarem a utilização de uma grande quantidade de virtualizadores. Em relação à segurança, se o desejo é utilizar uma base de dados externa de autenticação, são indicados o Eucalyptus, OpenStack, OpenQRM, Abiquo e Apache VCL, por possibilitarem autenticação LDAP ou integração com o *Active Directory*.

V CONCLUSÃO

A pesquisa realizada teve como objetivo apresentar, estudar e comparar as principais ferramentas de código aberto utilizadas na administração de nuvens. Após uma breve descrição das qualidades e funcionalidades individuais de cada uma delas, foram analisadas e comparadas com base em um conjunto de característica pré-definidas. Com isso, foi possível constatar que dentre as ferramentas pesquisadas existem várias diferenças e estas se tornaram mais simples de identificar com a comparação efetuada.

Com esta comparação, também foi possível ver que o OpenQRM foi a ferramenta que se mostrou mais completa nos quesitos avaliados, pois oferece uma maior gama de opções e possui todas as características elencadas. As ferramentas Eucalyptus, OpenNebula e OpenStack também se destacaram e ficaram bem próximas das qualidades do OpenQRM. Cabe ressaltar que, a melhor ferramenta é a que atende os requisitos do projeto de rede. Isso foi evidenciado na análise comparativa (Seção IV), na qual se tentou traçar cenários em que as ferramentas possam oferecer um bom comportamento.

Os desafios desta pesquisa como trabalhos futuros são de avaliar estas ferramentas (que se destacaram) em uma nuvem formada apenas por estações de trabalho. Para os experimentos, pretende-se estudar conjuntos de aplicações (*benchmarks*) que permitirão testar as mesmas características avaliadas qualitativamente nesta pesquisa. Assim, será possível efetuar um comparativo do que a literatura nos apresenta e como estas se comportam em um ambiente limitado e heterogêneo como este. Além disso, a hipótese é que isso pode ser uma boa alternativa para melhor aproveitar os recursos das máquinas que se encontram na maior parte do tempo ociosas nas instituições/organizações.

REFERENCES

[1] P. Mell e T. Grance. The NIST Definition of Cloud Computing. Gaithersburg: National Institute of Standards and Technology, 2011, p.7..

[2] E. A. Marks e B. Lozano. Executive's Guide to Cloud Computing.1ºEd. New Jersey: Published by John Wiley & Sons, Inc., 2010, p.285.

[3] M. Veras. Virtualização: Componente Central do Datacenter. 1º Ed. Rio de Janeiro: Brasport Livros e Multimidia Ltda, 2011, p. 364.

[4] P. T. Endo,G. E. Gonçalves, J. Kelner e D. Sadok. A Survey on Open-source Cloud Computing Solutions. In: VIII Workshop em Clouds, Grids e Aplicações. 2010. Gramado-RS. Anais. Gramado: Sociedade Brasileira de Computação, 2010, p.3-16.

[5] G. V. Laszewski, J. Diaz, F. Wang e G. C. Fox. *Comparison of Multiple Cloud Frameworks.* 2012 IEEE Fifth International Conference on Cloud Computing. Washington. 2012. p.734-741.

[6] A. Khurshid, A. Al-nayeem e I. Gupta. Performance Evaluation of the Illinois Cloud Computing Testbed. [S.I], Urbana-Champaign: Illinois Digital Environment for Access to Learning and Scholarsip, 2009, p.12.

[7] C. P. Machado. Comparação de ferramentas de software Livre para administração de nuvem privada. Canoas: Ulbra, 2011, p.18.

[8] I. Voras, B. Mihaljevic, M. Orlic, M. Pletikosa, M. Zagar, T. Pavic, K. Zimmer, I. Cavrak, V. Paunovic, I. Bosnic e S. Tomic. Evaluating Open-Source Cloud Computing Solutions. In: MIPRO, 2011 Proceedings of the 34th International Convention. 2011. Opatija-HR. Anais. Washington: IEEE Computer Society, 2011. p.209-214.

[9] P. Sempolinski e D. Thain. "A Comparison and Critique of Eucalyptus, OpenNebula and Nimbus". In: 2010 IEEE Second International Conference on Cloud Computing Technology and Science. 2010. Indianapolis-USA. Processo. Washington-USA: IEEE Computer Society, 2010. p.417-426.

[10] OpenNebula. *About the OpenNebula.org Project.* Extraído de < http://OpenNebula.org/about:about >. Acesso em 22 de julho de 2013.

[11] Openstack. *About OpenStack.* Extraído de < http://www.openstack.org/software/ >. Acesso em 30 de julho de 2013.

[12] N. Sabharwal e R. Shankar. *Apache Cloudstack Cloud Computing.* 1º Ed. Reino Unido: Packt Publishing Ltd., 2013, p.294.

[13] J. F. Ransome e J. W. Rittinghouse. *Cloud Computing: Implementation, Management and Security*. 1º Ed. Boca Raton: CRC Press, 2009, p. 407.

[14] S. Wardley, E. Goyer e N. Barcet. *Ubuntu Enterprise Cloud Architecture.* [S.l.], Man Island: Canonical, 2009, p.18.

[15] Abiquo. Abiquo. Extraído de: <http://www.abiquo.com/overview-technical/>. Acesso em 24 de julho de 2013.

[16] R. Buyya, J. Broberg e A. M. Goscinski. *Cloud Computing: Principles and Paradigms*. 1º Ed. John Wiley and Sons, 2010, p. 660.

[17] Apache VCL. Apache VCL. Extraído de < http://vcl.apache.org/>. Acesso em 10 de agosto de 2013.

[18] A. S. Pillai e L. S. Swasthimathi. A Study on Open Source Cloud Computing Plataforms. EXCEL International Journal of Multidisciplinary Management Studies. Zenit, 7 jul 2012. p.31-40.