UNIVERSIDADE ESTADUAL PAULISTA “Júlio de Mesquita Filho”

Pós-Graduação em Ciência da Computação

Henrique Pachioni Martins

Tolerância a Falha em um Ambiente de Computação em Nuvem *open source*

BAURU

2014

Henrique Pachioni Martins

Tolerância a Falha em um Ambiente de Computação em Nuvem *open source*

Orientador: Profa. Dra. Roberta Spolon

Dissertação de Mestrado elaborada junto ao Programa de Pós-Graduação em Ciência da Computação – Área de Concentração em Arquitetura de Computadores e Sistemas Distribuídos, como parte dos requisitos para a obtenção do título de Mestre em Ciência da Computação.

BAURU

2014

Martins, Henrique Pachioni.
Tolerância a falha em um ambiente de computação em nuvem open source / Henrique Pachioni Martins. -- São José do Rio Preto, 2014
82 f. : il., tab.

Orientador: Roberta Spolon
Dissertação (mestrado) – Universidade Estadual Paulista "Júlio de Mesquita Filho", Instituto de Biociências, Letras e Ciências Exatas

1. Computação. 2. Computação em nuvem. 3. Software gratuito. 4. Tolerância a falha (Computação) 5. Arquitetura de computador. I. Spolon, Roberta. II. Universidade Estadual Paulista "Júlio de Mesquita Filho". Instituto de Biociências, Letras e Ciências Exatas. III. Título.

CDU – 681.3.025

Ficha catalográfica elaborada pela Biblioteca do IBILCE
UNESP - Câmpus de São José do Rio Preto

Henrique Pachioni Martins

Tolerância a Falha em um Ambiente de Computação em Nuvem *open source*

Dissertação de Mestrado elaborada junto ao Programa de Pós-Graduação em Ciência da Computação – Área de Concentração em Arquitetura de Computadores e Sistemas Distribuídos, como parte dos requisitos para a obtenção do título de Mestre em Ciência da Computação.

Comissão Examinadora

_____

Profa. Dra. Roberta Spolon
Universidade Estadual Paulista - Bauru
Orientadora

_____

Prof. Dr. Antonio Carlos Sementille
Universidade Estadual Paulista - Bauru

_____

Prof. Dr. Luis Carlos Trevelin
Universidade Federal de São Carlos

Bauru, 17 de novembro de 2014.

Agradecimentos

Agradeço sempre a Deus, por sempre me orientar nos caminhos certos. A minha esposa Lucélia, por toda paciência e compreensão. Aos professores Drs. Marcos Cavenaghi, Renata Spolon Lobato, Aparecido Nilceu Marana e em especial a Roberta Spolon por acreditar e me orientar em todos os momentos no mestrado. E ao colega e parceiro Naylor Garcia, por toda ajuda e colaboração nessa fase do mestrado.

Resumo

A computação em nuvem é um conjunto de recursos e serviços oferecidos através da *internet*, entregues a partir de centros de dados localizados em todo o mundo. Com o rápido crescimento na área de computação em nuvem, aumenta a preocupação com a necessidade de serviços oferecidos e um grande desafio é implementar um ambiente tolerante a falhas. As principais questões de tolerância a falhas na computação em nuvem são a detecção e recuperação de falhas. Para combater esses problemas, muitas técnicas de tolerância a falhas são projetadas para reduzi-las. Gestores pagos oferecem esse tipo de suporte, mas os gestores *open source* não fornecem elementos que permitam tolerar falhas e deixam os usuários vulneráveis as falhas de um ambiente de tecnologia. O objetivo desse trabalho é desenvolver um mecanismo tolerante a falhas no OpenStack. Foi criado um mecanismo de redundância nas máquinas virtuais instanciadas nos *nodes* da nuvem, se um *node* apresentar uma falha transiente ou intermitente, a máquina virtual estará armazenada em um local seguro, aguardando que o *node* retorne de uma falha. O mecanismo desenvolvido é viável e eficiente, pois após um *node* se recuperar de uma falha, a máquina virtual não é perdida, voltando a ficar ativa para o usuário.

Palavras-chave: Computação em Nuvem. *open source*. tolerância a falha. OpenStack.

Abstract

Cloud computing is a set of features and services offered over the internet, delivered from data centers located around the world. With the rapid growth in cloud computing, increases the concern about the need for services offered and a major challenge is to implement a fault-tolerant environment. The main issues of fault tolerance in cloud computing are the fault detection and recovery. To combat these problems, many fault tolerance techniques are designed to reduce them. Paid managers offer this kind of support, but the open source managers do not provide evidence to tolerate failures and leave users vulnerable failures of a technology environment. The aim of this work is to develop a tolerant mechanism to failures in OpenStack. It was created a redundancy mechanism in virtual machines instantiated in cloud nodes, if a node present a transient or intermittent failure, the virtual machine will be stored in a safe place, waiting for the return of a node failure. The mechanism developed is feasible and efficient because after a node to recover from a failure, the virtual machine is not lost, getting back to the active user.

Keywords: *Cloud Computing, open source, fault tolerance, OpenStack.*

Lista de Figuras

Figura 1 – Características de computação em nuvem. 18
Figura 2 – Exemplo de computação em nuvem 19
Figura 3 – Formas de arquitetura da nuvem. 23
Figura 4 – Arquitetura de computação em nuvem. 26
Figura 5 – Modelo de negócios de computação em nuvem. 28
Figura 6 – Relação dos componentes da OpenStack. 36
Figura 7 - Cadeia de ameaça da dependabilidade. 41
Figura 8 – Ambiente inicial. 53
Figura 9 – Ambiente do OpenStackTF. 53
Figura 10 – Imagens ativas no ambiente. 55
Figura 11 – Configuração da máquina virtual. 55
Figura 12 – Máquina virtual ativa. 56
Figura 13 – Processo da máquina virtual. 56
Figura 14 - Fluxo da Máquina Virtual (MV) no ambiente. 57
Figura 15 - Gerenciamento de disco no OpenFiler. 58
Figura 16 - Volume iSCSI criado para cada *node*. 59
Figura 17 - Configuração de LUN Mapping. 59
Figura 18 - Configuração Target IQN. 60
Figura 19 – Simulando falha no *node*. 62
Figura 20 – Iniciando uma máquina virtual com solução de tolerância a falha. 62
Figura 21 – Simulando uma falha no *node* com solução de tolerância a falha. 63
Figura 22 – Máquina virtual ativa após falha no *node*. 63
Figura 23 – Lançamento de 10 máquinas virtuais. 64
Figura 24 – Limite máximo de instancias atingidas. 64
Figura 25 – Ambiente de Instalação. 75

Lista de Quadros

Quadro 1 – Diferenças entre os modelos tradicionais e de computação em nuvem. 20

Quadro 2 – Uma comparação de produtos comerciais representativos. 38

Quadro 3 – Comparação de gestores de computação em nuvem. 39

Quadro 4 – Fases para aplicação das técnicas de tolerância a falhas..... 44

Quadro 5 – Configuração dos computadores. 52

Lista de Abreviaturas

ACM - *Association for Computing Machinery*

Amazon EC2 - *Amazon Elastic Compute Cloud*

Amazon S3 - *Amazon Simple Storage Service*

AMQP - Protocolo Avançado de Fila de Mensagens

APIs - *Application Programming Interface*

AWS - *Amazon Web Services*

CC - *Cluster Controller*

CEO - *Chief executive officer*

CLC - *Cloud Controller*

CRM - Gestão de Relacionamento com Clientes

DBMS - Sistema de Gerenciamento de Banco de Dados

DRBD - *Distributed Replicated Block Device*

EC2 - *Elastic Compute Cloud*

EPIC - *Electronic Privacy Information Center*

FTC - Escritório da Comissão de Comércio Federal

HTTP - *HyperText Transfer Protocol*

HTTPS - *HyperText Transfer Protocol Secure*

IaaS - Infraestrutura como serviço

IEEE - *Instituto de Engenheiros Eletricistas e Eletrônicos*

iSCSI - *Internet Small Computer System Interface*

KVM - *Kernel-based Virtual Machine*

LAN - *Local Area Network*

LEPEC - Laboratório de Ensino, Pesquisa e Extensão em Computação

LVM - *Logical Volume Manager*

NAS - *Network Attached Storage*

NASA - *National Aeronautics and Space Administration*

NFS - *Network File System*

NC - *Node Controller*

NIST - *National Institute of Standards and Technology*

OSI - *Open Systems Interconnection*

PaaS - Plataforma como serviço

PDA - *Personal Digital Assistant*

REST - *Representational State Transfer*

SaaS - *Software* como serviço

SAN - *Storage Area Network*

SC - *Storage Controller*

SLA - *Service Level Agreement*

SOAP - *Simple Object Access Protocol*

SPOF - *Single Point of Failure*

TI - Tecnologia da Informação

UNESP - Universidade Estadual Paulista

URI - *Uniform Resource Identifier*

VLAN - Redes Locais Virtuais

VM - *Virtual Machines*

VPC - Nuvem Privada Virtual

VPN - Rede Privada Virtual

W - *Walrus*

WAN - *Wide Area Network*

WCF - *Windows Communication Foundation*

XML - *eXtensible Markup Language*

Sumário

1 – Introdução 13

1.1 Contexto 13

1.2 Objetivo 14

1.3 Organização do Texto 15

2 – Computação em Nuvem 16

2.1 Considerações iniciais 16

2.2 Arquitetura de Computação em Nuvem 22

2.3 Serviços de Computação em Nuvem 25

2.4 Benefícios e Riscos da Computação em Nuvem 29

2.5 Desafios da Computação em Nuvem 30

2.6 Gestores de Nuvem 31

2.6.1 *Amazon* EC2 31

2.6.2 *Microsoft Windows Azure platform* 33

2.6.3 Google App Engine 34

2.6.4 Vmware 34

2.6.5 OpenStack 35

2.6.6 Outros Gestores 36

2.7 Considerações Finais 37

3 – Tolerância a Falhas 40

3.1 Considerações iniciais 40

3.2 Modelos de falhas 43

3.3 Técnicas de Tolerância a Falhas 43

3.4 Tolerância a Falhas na Computação em Nuvem 45

3.5 Considerações Finais 47

4 – Desenvolvimento do Mecanismo para Tolerância a Falhas e Resultados 48

4.1 Considerações iniciais 48

4.2 Trabalhos Relacionados 50
4.3 Materiais 51
4.3.1 Ambiente Inicial 51
4.3.2 Ambiente do Mecanismo OpenStackTF 53
4.4 Métodos 54
4.5 Desenvolvimento do Mecanismo de Tolerância a Falhas - OpenStackTF 57
4.6 Testes e Resultados 60
4.7 Primeiro Cenário de Teste 62
4.8 Segundo Cenário de Teste 63
4.9 Avaliação dos Resultados 65
4.10 Considerações finais 66
5 – Conclusões e Trabalhos Futuros 67
5.1 Conclusões e Contribuições 67
5.2 Limitações 68
5.3 Trabalhos Futuros 68
REFERÊNCIAS 70
Apêndice A - Instalação do *Eucalyptus* 74
Apêndice B - Instalação do *OpenNebula* 76
Apêndice C - Instalação do *OpenStack* 78
Apêndice D - Instalação do *OpenFiler* 81

1 – Introdução

Esse capítulo apresenta uma introdução sobre os conceitos de computação em nuvem, o objetivo e a organização desse trabalho.

1.1 Contexto

A Tecnologia da Informação (TI) tem transformado e modificado a vida de pessoas e empresas durante os últimos anos, mudanças essas que estão fazendo com que seja repensada a maneira de se administrar um setor de tecnologia de uma empresa, ou até mesmo de como armazenar os dados pessoais. Isso porque a *internet* passa por muitos avanços, e que tem contribuído com um novo conceito de gerenciamento dos serviços de tecnologia chamado de *Cloud Computing* (Computação em Nuvem).

A Computação em nuvem é uma forma de oferecer recursos computacionais como os serviços. A forma de funcionamento da nuvem se assemelha aos sistemas operacionais de rede, onde os recursos computacionais são fornecidos como um serviço regular. (WU et al., 2010).

De maneira geral, a computação em nuvem favorece a migração dos servidores de dentro de uma empresa para uma nuvem. Os requisitos necessários para o desenvolvimento de

uma infraestrutura ou de um serviço são definidos, e por sua vez o fornecedor de serviços em nuvem cria essa infraestrutura. (ZHANG et al., 2010).

A Computação em nuvem tem sido vista como uma tendência no cenário atual em quase todas as organizações. As vantagens de usar a computação em nuvem são: i) redução de *hardware* e custo de manutenção, ii) acessibilidade em todo o mundo, e iii) flexibilidade (processo altamente automatizado em que o cliente não precisa se preocupar com atualização de *software*). (BHADAURIA; CHAKI, 2011). Questões como a localidade de dados devido a questões legais começam também a serem resolvidas.

Sabahi (2011) define Computação em nuvem como sendo um ambiente de rede baseado no compartilhamento de recursos computacionais. Nuvens são baseadas na *internet* e tentam disfarçar a complexidade para os clientes. Computação em nuvem refere-se a ambos pedidos entregues como serviços através da *internet* como, por exemplo, o *hardware* e o *software* nos centros de dados. Empresas que fornecem nuvens utilizam tecnologias de virtualização combinadas com suas habilidades para disponibilizar recursos de computação através de sua infraestrutura de rede.

Ainda segundo Sabahi (2011), em ambientes de nuvem, vários tipos de máquinas virtuais estão hospedadas no mesmo servidor físico, servindo como infraestrutura. Os recursos como armazenamento ou infraestrutura de uma nuvem são utilizados pelos clientes, e devem ser pagos somente por o que utilizarem.

Diante do contexto apresentado, surge a preocupação em ter-se um recurso tolerante a falha, pois se um recurso falha, outro precisa estar disponível. Entretanto, nem sempre a falha está no servidor que a apresenta, pois o servidor pode depender de outros servidores para prestar seus serviços, nesse caso a causa de um erro deve de ser detectada e corrigida.

1.2 Objetivo

Esse trabalho tem como objetivo propor um mecanismo de tolerância a falha nas máquinas virtuais ativas na nuvem OpenStack, fazendo com que a máquina virtual possa ser reutilizada quando o *node* com a falha volte a funcionar.

1.3 Organização do Texto

O restante desta dissertação está organizada em 5 capítulos descritos a seguir:

- No Capítulo 2 é apresentada uma revisão bibliográfica sobre o conceito de computação em nuvem, apresentando o estado da arte na área;
- No Capítulo 3 é feito um estudo sobre a tolerância a falhas, apresentando suas características e comparações em um ambiente em nuvem;
- O Capítulo 4 apresenta o desenvolvimento do trabalho, assim como os materiais e métodos utilizados durante a pesquisa, e são apresentados os resultados realizados no ambiente proposto;
- No Capítulo 5 são apresentadas as contribuições e conclusões obtidas, assim como as limitações encontradas durante o desenvolvimento da pesquisa, e os trabalhos futuros para continuação do trabalho.

2 – Computação em Nuvem

Esse capítulo apresenta conceitos sobre computação em nuvem, destacando as arquiteturas, tipos de serviços e gestores de nuvem.

2.1 Considerações iniciais

Computação em nuvem tem como principal característica permitir que os acessos às informações sejam feitos pela *internet* e não localmente. A utilização das informações na nuvem é facilitada pela utilização da infraestrutura, e a possibilidade de plataformas serem configuradas dinamicamente de acordo com as necessidades do usuário. (ZHANG et al., 2010).

Segundo a IBM (2012), a computação em nuvem pode ser categorizada como uma coleção ou soluções de tecnologias ou serviços, permitindo que os usuários acessem os recursos de computação de acordo com a demanda, conforme necessário. As necessidades podem ser de recursos físicos ou virtuais, dedicados ou compartilhados, e não importa como eles podem ser acessados, basta estar conectado na *internet*.

Mas o conceito de computação em nuvem ainda está sendo aprimorado, pois a ideia inicial de computação em nuvem é de armazenar e processar os dados fora do ambiente corporativo, dentro da grande rede, em estruturas conhecidas como centro de dados, otimizando o uso dos recursos. Os centros de dados irão fazer o papel de processar aplicações

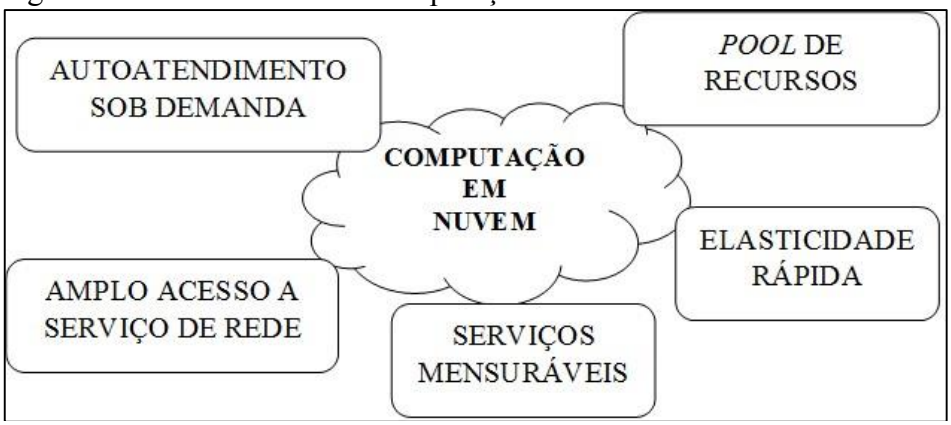
e armazenar os dados da organização que atuam em rede. (VERAS, 2012). Deixar os arquivos em rede, pode ser o mesmo que deixar os aplicativos e arquivos hospedados em uma nuvem, que consiste em milhares de computadores e servidores, todos ligados entre si e acessíveis via *internet*. Com a computação em nuvem, tudo que um usuário faz é baseado na *web* em vez de ser em área de trabalho local. O usuário tem acesso aos programas e documentos de qualquer computador que esteja conectado à *internet*.

Uma das maiores contribuições que se tem com a computação em nuvem é referente a maneira de trabalhar. O usuário não está mais vinculado a um único computador. Pode levar o trabalho em qualquer lugar porque é sempre acessível através da *web*. Além disso, a computação em nuvem facilita a colaboração em grupo, como por exemplo todos os membros do grupo podem ter acesso aos mesmos programas e documentos de onde quer que estejam localizados. (MILLER, 2008).

A Figura 1 ilustra as características essenciais de um ambiente em nuvem, que deve apresentar as seguintes características essenciais (NIST, 2011):

- **Autoatendimento sob demanda**: funcionalidades computacionais são providas automaticamente sem a interação humana com o provedor de serviço;
- **Amplo acesso a serviço de rede**: recursos computacionais estão disponíveis através da *internet* e são acessados via mecanismos padronizados, para que possam ser utilizados por dispositivos móveis e portáteis e computadores;
- ***Pool* de recursos**: recursos computacionais (físicos ou virtuais) do provedor são utilizados para servir a múltiplos usuários, sendo alocados e realocados dinamicamente conforme a demanda;
- **Elasticidade rápida**: as funcionalidades computacionais devem ser rápidas e elasticamente providas, assim como rapidamente liberadas. O usuário dos recursos deve ter a impressão de que ele possui recursos ilimitados, que podem ser adquiridos (comprados) em qualquer quantidade e a qualquer momento. A elasticidade tem três principais componentes: escalabilidade linear, utilização *on-demand* e pagamento por unidades consumidas de um recurso;
- **Serviços mensuráveis**: os sistemas de gerenciamento utilizados pela computação em nuvem controlam e monitoram automaticamente os recursos para cada tipo de serviço (armazenamento, processamento e largura de banda). Esse monitoramento do uso dos recursos deve ser transparente para o provedor de serviços, assim como para o consumidor do serviço utilizado.

Figura 1 – Características de computação em nuvem.



Fonte: Adaptado de Veras (2012).

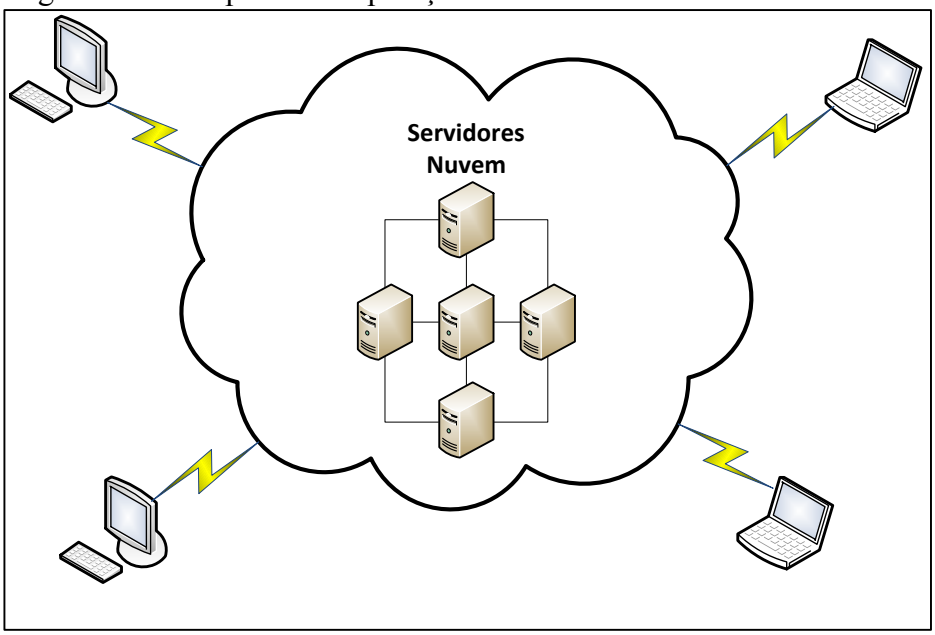
Segundo Breitman e Viterbo (2010), a ideia da computação em nuvem é a transição da computação tradicional para um novo modelo de administração, gerenciamento e consumos de recursos computacionais, como por exemplo, armazenamento, processamento, entre outros, e esses recursos são realizados através de serviços. A proposta da computação em nuvem é transformar os recursos computacionais da empresa para a responsabilidade de algumas empresas especializadas, as quais ficarão responsáveis por sua gestão e comercialização através de serviços.

O termo computação em nuvem foi introduzido em 2006, quando o então CEO (*Chief Executive Officer*) da Google[1], Eric Schmidt, utilizou o termo para descrever os serviços da própria empresa, e pouco tempo depois, quando a Amazon[2] utilizou o mesmo termo para lançar seu serviço EC2 (*Elastic Compute Cloud*). Foi, na verdade popularizado através do trabalho de George Gilder intitulado "*The Information Factories*". (GILDER, 2006).

Conforme ilustrado na Figura 2, os usuários individuais estão conectados na nuvem de seus próprios computadores pessoais ou aparelhos portáteis, através da *internet*. Para estes utilizadores individuais, a nuvem é visualizada como uma única aplicação, de documento ou dispositivos. O *hardware* na nuvem (e do sistema operacional que gerencia as conexões de *hardware*) é invisível. (MILLER, 2008).

[1] Empresa multinacional de serviços *online* e *software* dos Estados Unidos. O Google hospeda e desenvolve uma série de serviços e produtos baseados na *internet*.

Figura 2 – Exemplo de computação em nuvem



Fonte: Adaptado de MILLER (2008).

Mesmo não tendo uma definição única objetiva para o termo computação em nuvem, o termo geralmente é utilizado para descrever determinadas aplicações que são acessadas pela *internet*, ou para serviços de centro de dados. No caso das aplicações pode-se citar como exemplo os programas utilizados nos *desktops*, como editores de texto, planilhas ou, até mesmo, editores de imagens, onde a ideia é acessar essas aplicações através da *internet*, e todo o processamento e armazenamento de dados deveriam ocorrer no próprio computador do usuário, de uma forma *on-line*, ou "na nuvem". Já nos serviços de centro de dados, o termo computação em nuvem é utilizado quando o conjunto de recursos, como servidores, balanceadores de carga, armazenamento, são comercializados por uso e, normalmente, cobrados por hora. Este novo modelo de negócios trouxe uma série de benefícios técnicos e financeiros para consumidores de seus serviços como rápido provisionamento, escalabilidade, facilidade para lançamento de novos produtos/serviços por empresas de menor tamanho, entre outros. (BREITMAN; VITERBO, 2010).

Computação em nuvem e *grid*[3] compartilham os mesmos objetivos de redução de custos, aumento de flexibilidade e confiabilidade através da utilização de *hardware* operado por terceiros. A maior distinção entre os dois diz respeito a alocação de recursos. No caso do *grid* tenta-se fazer uma distribuição uniforme de recursos, e em um ambiente de computação

[2] Empresa multinacional de comércio eletrônico dos Estados Unidos da América com sede em Seattle. Foi uma das primeiras companhias com alguma relevância a vender produtos na *internet*.
[3] *Grid* computacional é uma coleção de recursos computacionais e de comunicação utilizados para execução de aplicações. (FOSTER et al., 2008).

em nuvem os recursos são alocados sob demanda. Outra diferença é referente a utilização dos recursos, pois a virtualização garante uma separação entre os recursos utilizados pelos vários usuários em ambientes de computação em nuvem. (BREITMAN; VITERBO, 2010).

Também existem importantes diferenças que distinguem o modelo de computação em nuvem do modelo tradicional de computação. O Quadro 1 apresenta um resumo dessas diferenças, destacando o modelo de negócio, que á caracterizado pelo usuário pagar pela utilização, e o modelo de acesso destaque que na nuvem é realizado através da *internet*.

Quadro 1 – Diferenças entre os modelos tradicionais e de computação em nuvem.

	Computação Tradicional	**Computação em Nuvem**
Modelo de aquisição	*Hardware* Espaço físico Infraestrutura de instalação e funcionamento	Aquisição de serviço
Modelo de negócio	Custo e depreciação de ativos *Overhead* administrativo (manutenção, suporte, segurança do equipamento, refrigeração)	Pagamento baseado na utilização
Modelo de Acesso	Rede interna *Intranet*	*Internet*, através de vários tipos de dispositivos (não apenas computadores)
Modelo técnico	Único "morador" Sem compartilhamento Estático	Escalável Elástico Dinâmico Condominio

Fonte: Adaptado de CEARLEY (2009).

De acordo com Zhang (2010), a computação em nuvem oferece várias características importantes que são diferentes do serviço de computação tradicional, tai como:

- **Vários inquilinos**: Em um ambiente de nuvem, serviços de propriedade de múltiplos fornecedores são alocados em um único centro de dados. Os problemas de desempenho e gestão desses serviços são compartilhados entre prestadores de serviços e infraestrutura do provedor. A arquitetura em camadas de computação em nuvem oferece uma divisão natural de responsabilidades: o proprietário de cada camada só precisa se concentrar nos objetivos específicos associados a essa camada. No entanto, alocar vários fornecedores também apresenta dificuldades em entender e gerenciar as interações entre as várias partes interessadas;
- **Compartilhamento de *pool* de recurso**: O provedor de infraestrutura oferece um *pool* de recursos computacionais, os quais podem ser atribuídos dinamicamente aos consumidores de recursos múltiplos. A capacidade de tal atribuição dinâmica de

recursos fornece uma grande flexibilidade aos provedores de infraestrutura para gerenciar seu uso de recursos próprios e os custos operacionais. Por exemplo, um provedor de Infraestrutura como Serviço (IaaS[4]) pode aproveitar a tecnologia de migração VM (*Virtual Machines*) para atingir um alto grau de consolidação de servidores, portanto, maximizando a utilização dos recursos, minimizando custos, tais como o consumo de energia e refrigeração;

- **Acesso à rede geograficamente distribuída e onipresente**: As nuvens são geralmente acessíveis através da *internet*, por isso, qualquer dispositivo com ligação à *internet*, seja ele um celular, um PDA (*Personal Digital Assistant*) ou um *laptop*, é capaz de acessar serviços em nuvem. Além disso, para atingir alto desempenho de rede e localização, muitas das nuvens de hoje contam com centros de dados localizados em diversas localidades ao redor do globo. Um prestador de serviços pode facilmente alavancar geodiversidade para alcançar utilidade máxima de serviço;
- **Orientada a serviços**: Como mencionado anteriormente, a computação em nuvem adota um modelo operacional orientado para serviço. Por isso, coloca uma forte ênfase na gestão de serviços. Em uma nuvem, cada provedor de Infraestrutura como Serviço (IaaS), Plataforma como Serviço (PaaS[5]) e *Software* como Serviço (SaaS[6]) oferece o seu serviço de acordo com o *Service Level Agreement* (SLA) negociado com seus clientes. Garantia de SLA é, portanto, um objetivo fundamental de cada provedor;
- **Provisionamento de recursos dinâmicos**: Uma das características principais da computação em nuvem é que os recursos de computação podem ser obtidos e divulgados. Comparado com o modelo tradicional em que os recursos disponíveis de acordo com a demanda, os recursos são provisionados dinamicamente, permitindo que prestadores de serviços possam adquirir recursos com base na demanda atual, reduzindo assim o custo operacional;
- **Auto-organização**: Os recursos podem ser movidos dos prestadores de serviços. Estão capacitados a gerenciar seu consumo de recursos de acordo com as suas próprias necessidades. Além disso, o recurso de gerenciamento automatizado permite aos provedores de serviços responder rapidamente às mudanças na demanda de serviços;

---

[4] IaaS - É a camada mais baixa da infraestrutura de uma nuvem, que fornece recursos de infraestrutura, geralmente usando a tecnologia de virtualização.
[5] PaaS - É a camada intermediária da infraestrutura de uma nuvem, que oferece serviços orientadas de plataforma, proporcionando hospedagens e execução de *softwares* ao usuário.
[6] IaaS - É a camada mais baixa da infraestrutura de uma nuvem, que fornece recursos de infraestrutura, geralmente usando a tecnologia de virtualização.

- **Utilitário baseado em preços**: A computação em nuvem utiliza um modelo de "pague-pelo-uso". O esquema de preço exato pode variar de serviço para serviço. Por exemplo, um provedor de SaaS pode alugar uma máquina virtual de um provedor de IaaS em uma base por hora. Por outro lado, um provedor de SaaS que fornece *on-demand* de gestão de relacionamento com clientes (CRM) pode cobrar de seus clientes com base no número de clientes que serve (por exemplo, Salesforce[7]). Utilitários baseados em preços reduzem o custo operacional do serviço, pois cobram dos clientes em uma base "pague-pelo-uso". No entanto, também introduz complexidades em controlar o custo de operação. Nesta perspectiva, empresas como a VKernel[8] fornecem *software* para ajudar os clientes em nuvem a compreender, analisar e reduzir o custo desnecessário no consumo de recursos.

2.2 Arquitetura de Computação em Nuvem

A chave para a computação em nuvem é a "nuvem", uma rede massiva de servidores, ou até mesmo computadores individuais interligados em uma *grid*. Esses computadores funcionam em paralelo, combinando os recursos de cada um para gerar um supercomputador com alto poder. (MILLER, 2008).

Independentemente dos modelos de serviços, a arquitetura da nuvem pode ser implantada de quatro formas, dependendo dos requisitos dos clientes. Veras (2012) descreve as seguintes arquiteturas:

- **Nuvem Privada (*Private Cloud*):** compreende uma infraestrutura de computação em nuvem operada e quase sempre gerenciada pela organização cliente. Os serviços são oferecidos para serem utilizados pela própria organização, não estando publicamente disponível para uso geral. Em alguns casos pode ser gerenciada por terceiros;
- **Nuvem Pública (*Public Cloud*):** é disponibilizada publicamente através do modelo "pague-pelo-uso". São oferecidas por organizações públicas ou por grandes grupos industriais que possuem capacidade de processamento e armazenamento;
- **Nuvem Comunitária (*Community Cloud*):** neste caso, a infraestrutura de computação em nuvem é compartilhada por diversas organizações e suporta uma

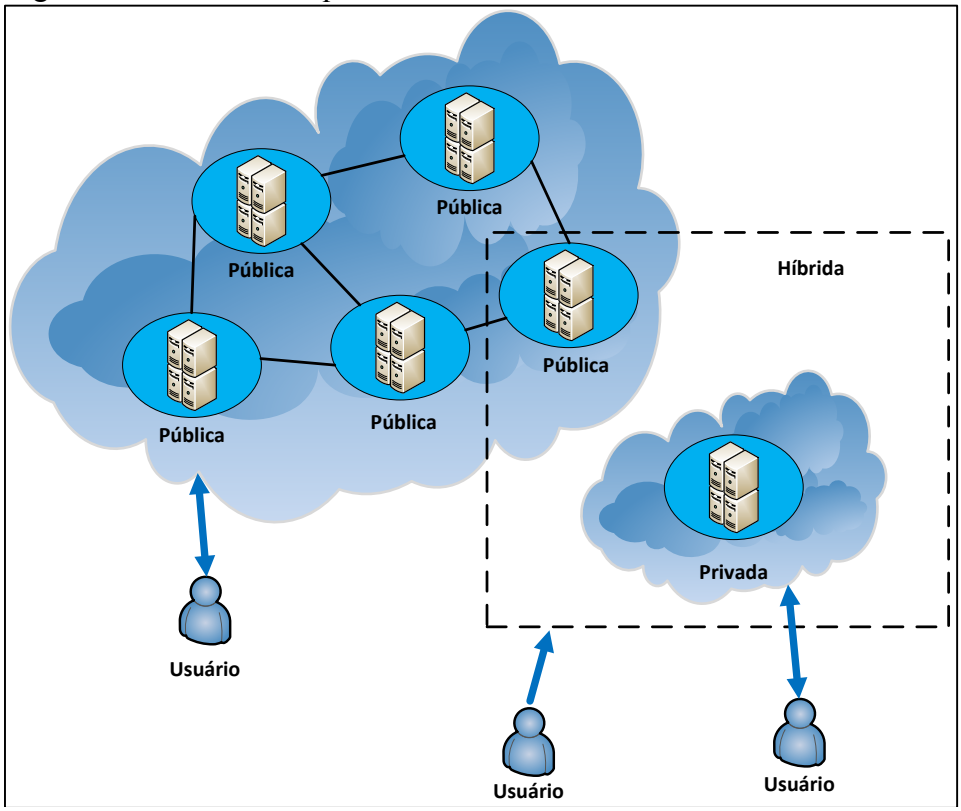
[7] Salesforce é uma empresa líder de mercado em CRM no mundo. Sua plataforma flexível permite a uma empresa gerenciar todo o relacionamento com o cliente, desde a área comercial até o atendimento.
[8] A VKernel, fundada em 2007 é a empresa líder em produtos de gestão do desempenho e capacidade para VMware e Microsoft Hyper-V.

comunidade que possui interesses comuns. A nuvem comunitária pode ser administrada pelas organizações que fazem parte da comunidade ou por terceiros e pode existir tanto fora como dentro da organização;

- **Nuvem Híbrida (*Hybrid Cloud*):** a infraestrutura é uma composição de duas ou mais nuvens (privadas, pública ou comunitária) que continuam a ser entidades únicas, porém, conectadas através de tecnologias proprietárias ou padronizadas que propiciam a portabilidade de dados e aplicações. A nuvem híbrida impõe uma coordenação adicional a ser realizada para uso das nuvens privadas e públicas.

As organizações podem optar por implantar aplicativos em três tipos de nuvens públicas, privadas ou híbridas, conforme apresenta a Figura 3.

Figura 3 – Formas de arquitetura da nuvem.



Fonte: Adaptado de Furht (2010).

Segundo Zhang (2010), há muitas questões a considerar quando se opta por migrar um ambiente para a nuvem. Por exemplo, alguns prestadores de serviços estão em sua maioria interessados em reduzir o custo de operação, enquanto outros podem preferir alta confiabilidade e segurança. Assim, existem diferentes tipos de nuvens, cada um com suas próprias vantagens e desvantagens:

- **Nuvens públicas:** Uma nuvem em que os prestadores de serviços oferecem seus recursos como serviços ao público em geral. As nuvens públicas oferecem vários benefícios importantes para os prestadores de serviços, incluindo investimento de capital inicial em infraestrutura e transferência de riscos para fornecedores de infraestrutura. No entanto, as nuvens públicas não têm um controle refinado sobre os dados de rede e configurações de segurança, o que dificulta a sua eficácia em vários cenários de negócios;
- **Nuvens privadas:** Também conhecidas como nuvens internas, as nuvens privadas são projetadas para uso exclusivo de uma única organização. Uma nuvem privada pode ser construída e gerenciada pela organização ou por provedores externos. Uma nuvem privada oferece o mais alto grau de controle sobre o desempenho, confiabilidade e segurança;
- **Nuvens híbridas:** Uma nuvem híbrida é uma combinação de modelos de nuvens públicas e privadas que tenta resolver as limitações de cada abordagem. Em uma nuvem híbrida, parte da infraestrutura de serviço é executada em nuvens privadas, enquanto a parte restante é executada em nuvens públicas. Nuvens híbridas oferecem mais flexibilidade do que as nuvens públicas e privadas. Especificamente, provêm maior controle e segurança sobre os dados de aplicação em comparação com as nuvens públicas, ao mesmo tempo facilitam o *on-demand* de expansão e contratação de serviços. Projetar uma nuvem híbrida requer cuidado ao determinar a melhor divisão entre os componentes de nuvens públicas e privadas;
- **Nuvem privada virtual:** Uma solução alternativa para enfrentar as limitações de nuvens públicas e privadas é chamada de *Private Cloud Virtual* (VPC). Uma VPC é essencialmente uma plataforma executando em cima de nuvens públicas. A principal diferença é que um VPC utiliza tecnologia de rede privada virtual (VPN) que permite que os prestadores de serviços projetem a sua própria topologia e configurações de segurança, tais como regras de *firewall*. VPC é essencialmente um projeto mais abrangente, uma vez que não só virtualiza servidores e aplicações, mas também a rede de comunicação subjacente. Além disso, para a maioria das empresas, VPC fornece transição suave de uma infraestrutura de serviço de propriedade de uma infraestrutura baseada em nuvem, devido à camada de rede virtualizada.

2.3 Serviços de Computação em Nuvem

A computação em nuvem pode ser considerada como uma arquitetura formada por várias camadas, as quais podem ser consideradas como um conjunto de recursos virtualizados, e podem fornecer serviços aos usuários. (ZHANG et al., 2010).

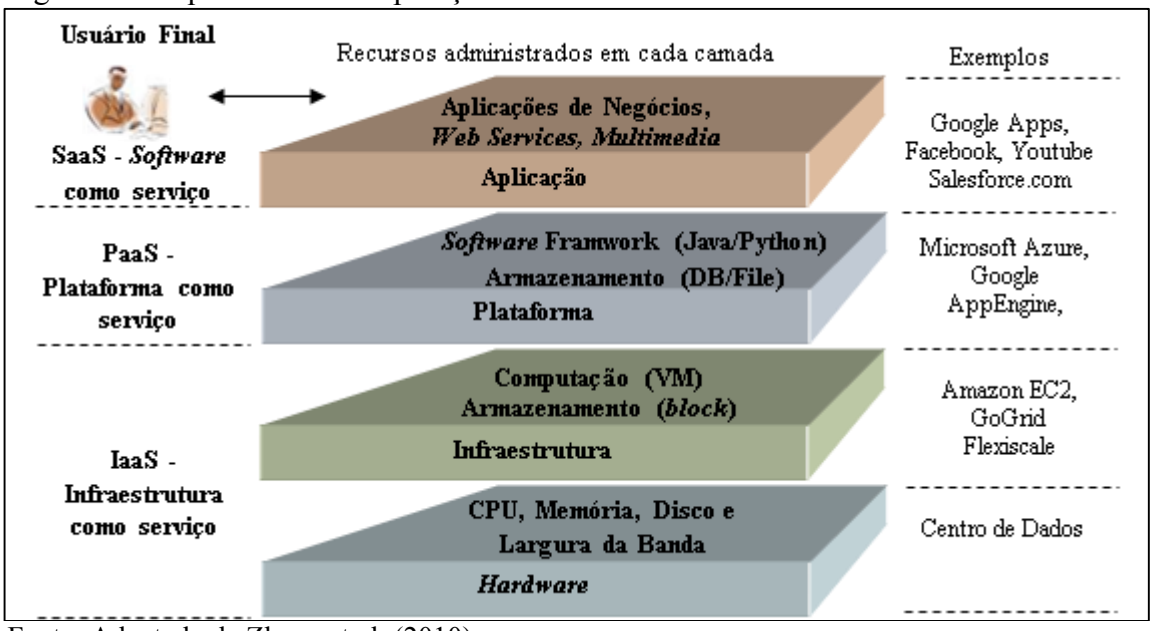
A computação em nuvem pode oferecer diferentes tipos de serviços como *hardware*, *software*, dados e infraestrutura. Esses serviços podem ser classificados e agrupados em três principais categorias:

- **IaaS (Infraestrutura como serviço):** É a camada mais baixa que fornece serviço de suporte básico de infraestrutura. Refere-se à partilha dos recursos de *hardware* para serviços de execução, geralmente usando a tecnologia de virtualização. Os recursos podem facilmente ser aplicados dependendo da demanda do usuário e normalmente são cobrados na forma de "pague-pelo-uso". Segundo Veras (2012), é definida como sendo a capacidade que o provedor tem de oferecer uma infraestrutura de processamento e armazenamento de forma transparente. Neste cenário, o usuário não tem o controle da infraestrutura física, mas, através de mecanismos de virtualização, possui controle sobre as máquinas virtuais, armazenamento, aplicativos instalados e possivelmente um controle limitado dos recursos da rede;
- **PaaS (Plataforma como serviço):** É a camada intermediária, que oferece serviços orientadas de plataforma, além de proporcionar o ambiente para hospedagem de aplicativos do usuário. A oferta também inclui um ambiente de execução de *software*. Como por exemplo, poderia haver um servidor de aplicação PaaS que permite aos desenvolvedores implantar aplicativos baseados na *web* sem comprar servidores reais e configurá-los. O modelo PaaS tem como objetivo proteger os dados, o que é especialmente importante no caso de armazenamento como um serviço. Assim, a necessidade de segurança contra a interrupção é importante para garantir um serviço de balanceamento de carga. Os dados precisam ser criptografados quando hospedados em uma plataforma deste tipo;
- **SaaS (*Software* como serviço):** É a camada superior que possui uma aplicação completa oferecida como serviço *on-demand*. O pagamento é feito em um modelo "pague-pelo-uso". Ele elimina a necessidade de instalar e executar o aplicativo no computador local do cliente, aliviando assim a carga do cliente para manutenção de *software*. De acordo com Veras (2012), permite que os aplicativos de interesse para grande quantidade de clientes possam ser hospedados na nuvem como uma alternativa

ao processamento local. Os aplicativos são oferecidos no *browser*. Todo o controle e gerenciamento da rede, sistemas operacionais, servidores e armazenamento é feito pelo provedor de serviço.

Segundo Zhang et al. (2010), de um modo geral, a arquitetura de um ambiente de computação em nuvem pode ser dividida em 4 camadas (Figura 4): a camada de *hardware*/*datacenter*, a camada de infraestrutura, a camada de plataforma e a camada de aplicação, detalhadas a seguir.

Figura 4 – Arquitetura de computação em nuvem.



Fonte: Adaptado de Zhang et al. (2010).

- **Camada de *hardware*:** Responsável por gerenciar os recursos físicos da nuvem, incluindo servidores físicos, roteadores, *switches*, energia e sistemas de refrigeração. Na prática, a camada de *hardware* é tipicamente implementada em centros de dados. Um centro de dados geralmente contém milhares de servidores que são organizados em prateleiras e interligados através de *switches*, roteadores ou outros tecidos. Questões típicas da camada de *hardware* incluem *hardware*, configuração de tolerância a falhas, gestão do tráfego de alimentação e gestão de recursos de resfriamento;
- **Camada de infraestrutura:** Também conhecida como a camada de virtualização, a camada de infraestrutura cria um *pool* de recursos de armazenamento e computação, dividindo os recursos físicos que utilizam

tecnologias de virtualização, como Xen[9], KVM[10] e VMware[11]. A camada de infraestrutura é um componente essencial da computação em nuvem, uma vez que muitas características chave, tais como a atribuição dinâmica de recursos, só são disponibilizadas através de tecnologias de virtualização;

- **Camada de plataforma:** Construída sobre a camada de infraestrutura, a camada de plataforma consiste de sistemas operacionais e estruturas de aplicativos;
- **Camada de aplicação:** No nível mais alto da hierarquia, a camada de aplicação consiste nas aplicações em nuvem reais. Diferente de aplicações tradicionais, aplicações em nuvem podem aproveitar o recurso de dimensionamento automático para obter melhor desempenho, disponibilidade e menor custo operacional.

Comparada com ambientes de serviços tradicionais de hospedagem, tais como *sites* de servidores dedicados, a arquitetura de computação em nuvem é modular. Cada camada é fracamente acoplada com as camadas anteriores e posteriores, permitindo que cada camada possa evoluir separadamente. A modularidade arquitetônica permite a computação em nuvem suportar uma ampla gama de requisitos de aplicação, reduzindo custo de manutenção e gestão. (ZHANG et al., 2010).

A computação em nuvem emprega um modelo de negócio orientado para serviço. Em outras palavras, plataforma de *hardware* e de nível de recursos são fornecidos como serviços em uma base *on-demand*. Conceitualmente, cada camada da arquitetura pode ser implementada como um serviço para a camada superior. Por outro lado, cada camada pode ser percebida como um cliente da camada abaixo. Na prática, as nuvens oferecem serviços que podem ser agrupados em categorias: *software* como serviço, plataforma como serviço e infraestrutura como serviço, descritos a seguir: (ZHANG et al., 2010).

- Infraestrutura como Serviço: (IaaS) refere-se a pedido de provisionamento de recursos de infraestrutura, geralmente em termos de VMs. O proprietário da nuvem que oferece

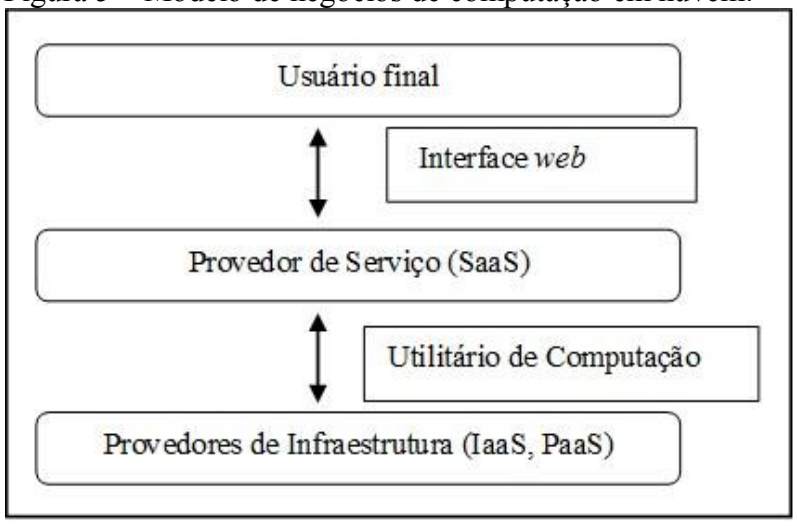
[9] Definição na seção 2.7 desse trabalho.
[10] Definição na seção 2.7 desse trabalho.
[11] Definição na seção 2.6 desse trabalho.

IaaS é chamado de um provedor de IaaS. Um exemplo de provedores IaaS é a Amazon EC2[12];

- Plataforma como um Serviço: (PaaS) refere-se a fornecer recursos da plataforma de camadas, incluindo suporte a sistemas operacionais e *frameworks* de desenvolvimento de *software*. Exemplos de provedores PaaS incluem Google App Engine[13] e Microsoft Windows Azure[14];
- *Software* como Serviço: (SaaS) refere-se a fornecer aplicações *on-demand* através da *internet*. Exemplos de provedores de SaaS incluem Salesforce.com[15] e Rackspace[16].

O modelo de negócios de computação em nuvem está representado na Figura 5. De acordo com a arquitetura em camadas de computação em nuvem, é inteiramente possível que um provedor de PaaS execute sua nuvem sobre um provedor de IaaS. No entanto, na prática atual, os fornecedores de IaaS e PaaS são muitas vezes partes da mesma organização (por exemplo, Google e Salesforce). É por isso que PaaS e prestadores de IaaS são frequentemente chamados de provedores de infraestrutura ou de provedores de nuvem.

Figura 5 – Modelo de negócios de computação em nuvem.



Fonte: Adaptado de Zhang et al. (2010).

[12] Definição na seção 2.6 desse trabalho.
[13] Definição na seção 2.6 desse trabalho.
[14] Definição na seção 2.6 desse trabalho.
[15] Empresa americana de *software on demand*, conhecida por ter produzido o CRM com o mesmo nome da empresa.
[16] Provedora americana de infraestrutura de nuvem.

2.4 Benefícios e Riscos da Computação em Nuvem

O principal benefício da computação em nuvem é o ganho de escala propiciado pela arquitetura. Por exemplo, servidores sem uso representam um problema tanto no aspecto do gerenciamento quanto no aspecto do consumo de energia. Servidores a plena carga e servidores a baixa carga consomem energia de forma próxima, portanto, servidor sem uso é sinônimo de ineficiência. Na nuvem, a utilização destes servidores seria otimizada. E segundo Veras (2012), uma forma de entender a economia de escala pela computação em nuvem é entender os três aspectos envolvidos: economia de escala do lado do fornecimento, economia de escala do lado da demanda e economia de escala da arquitetura *multitenancy*[17], os quais são descritos a seguir:

- **Economia de escala do lado do fornecimento:** é propiciada por grandes centros de dados que baixam o custo do servidor, onde pode ocorrer a redução do custo de energia, e até mesmo, redução de custos de pessoal, onde um administrador consegue gerenciar vários servidores;
- **Economia de escala do lado da demanda:** ocorre devido a agregação de demanda que atenua o problema da variabilidade de carga e permite aumentar a taxa de uso do servidor;
- **Economia de escala da arquitetura *multitenancy*:** onde pode aumentar o número de compartilhamento do mesmo aplicativo, reduzindo a necessidade de gerenciamento e o custo de servidores por usuários.

Na computação em nuvem, existe a possibilidade que algum evento imprevisto falhe, ou mesmo o mau uso ameace um objeto de um negócio. Existem várias características importantes no modelo da computação em nuvem, e a combinação destas características pode fazer o risco variar. Alguns cuidados que o cliente deve ter para entender o risco referente a aquisição de serviços de um provedor de computação em nuvem, são por exemplo: saber como é feito o acesso dos usuários, saber como o provedor obedece as normas de regulação, saber onde se localizam os dados, saber como os dados são segregados, saber como os dados são recuperados, saber como é feito o suporte, e entender a viabilidade do provedor a longo prazo.

[17] Essa arquitetura provê uma única aplicação compartilhada por vários clientes. Nessa abordagem várias aplicações virtuais são criadas na mesma instância. (VERAS, 2012).

2.5 Desafios da Computação em Nuvem

Conforme Roberts e Al-Hamdani (2011), caso uma empresa decida utilizar da nuvem, deve seguir algumas recomendações de segurança. Um dos fatores que uma empresa pode seguir é ter uma padronização dos níveis de segurança, pois a computação em nuvem é composta de vários servidores e centro de dados através da *internet*. A padronização pode ser feita nos níveis de segurança de toda organização. Essa padronização permitiria que usuários pudessem designar níveis específicos de segurança para diferentes informações. Por exemplo, uma organização que trabalha em pesquisa e desenvolvimento poderia colocar uma segurança alta em projetos que ainda não foram publicados ou apresentados ao mercado. Na teoria, ao permitir que diferentes itens possam ter atribuições de segurança diferentes, a carga de trabalho e níveis de segurança poderia estar focada apenas nos itens que precisam de segurança.

Outra preocupação sobre a segurança na nuvem em relação aos conhecimentos das pessoas sobre os itens de segurança, pois as mudanças na área de segurança são muito dinâmicas, e uma dessas preocupações é a engenharia social[18], onde os *hackers* obtém acessos a informações confidenciais. (ROBERTS; AL-HAMDANI, 2011).

A computação em nuvem tem falhas de segurança, mas o mesmo acontece com a computação tradicional. Há falhas de segurança em todas as formas de computação. O principal fator é o quanto alguém está disposto a tirar proveito das falhas de segurança para obter suas informações. Cada usuário deve estar ciente de que nenhuma forma de computação é segura, no entanto medidas podem ser tomadas para diminuir as chances de exposição.

Em conformidade com Sumter (2010), devido a computação em nuvem ser um assunto novo, ainda surgem dúvidas sobre sua utilização. Além de que, pesquisas sobre segurança e custo estão sendo realizadas, uma vez que a computação em nuvem pode ser o substituto dos *desktops*, mas antes que isso aconteça várias questões precisam ser pesquisadas, tais como segurança, domínio da empresa e questões jurídicas. O problema de segurança na nuvem tem sido levantado pelos governos de diversos países. No caso do governo Federal dos Estados Unidos, a *Electronic Privacy Information Center* (EPIC[19]) pediu ao Escritório da

[18] Engenharia social é termo utilizado para descrever um método de ataque, onde alguém faz uso da persuasão, muitas vezes abusando da ingenuidade ou confiança do usuário, para obter informações que podem ser utilizadas para ter acesso não autorizado a computadores ou informações.

[19] Organização de interesse público com sede em Washington, D.C. Fundada em 1994 com foco na proteção de liberdade civil dentre elas a privacidade, que é a primeira emenda da constituição dos Estados Unidos da América.

Comissão de Comércio Federal (FTC[20]) em sua última pesquisa de março de 2009 para levantar informações sobre o Gmail, Google Docs, Google Calendare e outros aplicativos da *web* da empresa Google aprovados pelo governo. A FTC demonstrou preocupação sobre as informações que sejam armazenadas em uma nuvem, e como podem fluir através das fronteiras do país, em violação das leis comerciais estabelecidas na exportação. A agência identificou os países que tem jurisdição sobre os dados que fluem através das fronteiras. Outra grande preocupação é sobre as informações que estão armazenadas na nuvem, se estão a salvo dos *hackers*.

Existem muitos problemas de vulnerabilidades existentes na nuvem, e o problema não é somente encontrá-los, e sim aplicar resposta adequada contra todos os problemas. Geralmente, a nuvem tem por base um conjunto de mecanismos de armazenamento especializados, dirigido por um coordenador de transações distribuídas, que também suporta alta disponibilidade. Para atingir flexibilidade, escalabilidade e eficiência de utilização dos recursos disponíveis, os provedores de nuvem devem enfrentar grandes desafios na adaptabilidade da carga de trabalho. (SABAHI, 2011).

2.6 Gestores de Nuvem

Essa seção descreve alguns gestores (provedores de serviços) de computação em nuvem.

2.6.1 *Amazon* EC2

Segundo Zhang (2010), a *Amazon Web Services*[21] (AWS) é um conjunto de serviços em nuvem, fornecendo computação em nuvem baseado em armazenamento e outras funcionalidades que permitem que empresas e indivíduos possam implantar aplicativos e serviços em uma base *on-demand* e no preço das *commodities*. Ofertas de serviços Web da Amazon são acessíveis através de HTTP (HyperText Transfer Protocol), usando protocolos REST (*Representational State Transfer)* e SOAP (*Simple Object Access Protocol*).

[20] Orgão que promove a defesa do consumidor e erradica práticas anticompetitivas. A FTC regula as práticas comerciais em geral.

Amazon Elastic Compute Cloud (Amazon EC2) permite que os usuários da nuvem possam gerenciar as instâncias de servidores em centros de dados usando APIs ou ferramentas disponíveis e utilitários. Instâncias do EC2 são máquinas virtuais rodando em cima do mecanismo de virtualização Xen. Depois de criar e iniciar um exemplo, os usuários podem fazer upload de *software* e fazer alterações nele. Quando as alterações estão finalizadas, eles podem ser agrupados como uma imagem em uma nova máquina. Uma cópia idêntica pode então ser lançada em qualquer momento. Os usuários têm controle quase total da pilha de *software* inteiro sobre as instâncias do EC2 que se parecem com *hardware* para eles. Por outro lado, esta característica torna inerentemente difícil para a Amazon para oferecer a escala automática de recursos. (ZHANG, 2010).

EC2 fornece a capacidade de colocar ocorrências em vários locais. EC2 locais são compostas por regiões e zonas de disponibilidade. Regiões consistem de um ou mais zonas de disponibilidade, e dispersos geograficamente. Imagens de máquinas EC2 são armazenados e retirado da *Amazon Simple Storage Service (Amazon* S3). S3 armazena dados como "objetos" que são agrupados em "cubos". Cada objeto contém de 1 *byte* a 5 *gigabytes* de dados. Nomes de objetos são essencialmente caminhos URI (*Uniform Resource Identifier*). Cubos devem ser criados explicitamente, antes de poderem ser utilizados. Um cubo pode ser armazenado em várias regiões. Os usuários podem escolher uma região para otimizar a latência, minimizar os custos, ou atender às exigências regulamentares. (ZHANG, 2010).

Amazon Virtual Private Cloud (VPC) é uma ponte segura e sem interrupções entre uma empresa de infraestrutura de TI existentes e da nuvem AWS. Amazon VPC permite às empresas conectar sua infraestrutura existente para um conjunto de recursos computacionais AWS isolado, através de uma conexão de rede privada virtual (VPN), e para estender suas capacidades de gestão existentes, tais como serviços de segurança, *firewalls* e sistemas de detecção de intrusão para incluir seu AWS recursos. (ZHANG, 2010).

---

[21] A Amazon Web Services (AWS) oferece um conjunto completo de serviços de aplicativos e infraestrutura que permitem executar praticamente tudo na nuvem: desde aplicativos empresariais e projetos de dados grandes a jogos sociais e aplicativos móveis.

2.6.2 *Microsoft Windows Azure platform*

Ainda segundo Zhang (2010), o gestor *Microsoft Windows Azure*[22] consiste de três componentes, e cada um deles fornece um conjunto específico de serviços para usuários em nuvem. *Windows Azure* oferece um ambiente baseado em Windows para execução de aplicativos e armazenamento de dados em servidores nos centros de dados; SQL *Azure* fornece serviços de dados na nuvem baseado em SQL *Server*; e NET *Services* oferecem serviços de infraestrutura distribuídos para aplicativos baseados em nuvem e locais. O Gestor *Windows Azure* pode ser usado tanto por aplicativos executados na nuvem e por aplicativos executados em sistemas locais.

Windows Azure também suporta aplicativos criados no .NET *Framework* e outras linguagens comuns suportadas em sistemas *Windows*, como C#, *Visual Basic*, C++, e outros. Windows Azure suporta programas de uso geral, ao invés de uma única classe de computação. Os desenvolvedores podem criar aplicações web utilizando tecnologias como o ASP.NET e Windows Communication Foundation (WCF), e os aplicativos são executados como processos em segundo plano independentes, ou aplicações que combinam os dois. *Windows Azure* permite armazenar dados em *blobs*[23], tabelas e filas, todos acessados em um estilo RESTful via HTTP ou HTTPS. (ZHANG, 2010).

Componentes do SQL *Azure* são bancos de dados SQL *Azure* e "Huron" Sync Data. O Banco de dados SQL *Azure* é construído sobre o *Microsoft* SQL *Server*, oferecendo um sistema de gerenciamento de banco de dados (DBMS) em nuvem. Os dados podem ser acessados usando o ADO.NET e as interfaces do Windows acessam outros dados. Os usuários também podem usar o *software* no local para trabalhar com esta informação baseada em nuvem. (ZHANG, 2010).

Para Zhang (2010), os Serviços .NET facilitam a criação de aplicações distribuídas. O componente de controle de acesso fornece uma implementação baseada em nuvem de verificação de identidade única em aplicações de empresas. A cada terminal exposto é atribuído um URI, que os clientes podem usar para localizar e acessar um serviço.

Todos os recursos físicos e aplicações no centro de dados são monitoradas por um *software* chamado de *controller*. A cada aplicação, os usuários fazem *upload* de um arquivo

---

[22] Windows Azure é uma plataforma especial para execução de aplicativos e serviços, baseada nos conceitos da computação em nuvem.

[23] Blobs: Objetos de dados que armazenam informações multimídias no formato binário em tabelas de banco de dados. Como ele é um objeto de armazenamento de dados pode ser chamado também de campo de dados. Blob se origina do nome lob que significa Large Object, traduzindo para o português, algo como objeto grande.

de configuração que fornece uma descrição baseada em XML (*eXtensible Markup Language*) do que a aplicação precisa. Com base nesse arquivo, o *controller* decide onde novas aplicações devem ser executadas, escolhendo servidores físicos para otimizar a utilização de *hardware*. (ZHANG, 2010).

2.6.3 Google App Engine

Os autores citados anteriormente destacam que a *Google App Engine* é uma plataforma para aplicações *web* tradicional da Google para o gerenciamento de dados. Atualmente, as linguagens de programação suportadas são Python e Java. *Frameworks* de desenvolvimento *web* que são executadas no *Google App Motor* incluem Django, CherryPy, Pylons, e web2py. Google lida com implantação de código de um *cluster*, o monitoramento, *failover*[24] e lançamento de instâncias de aplicativos, quando necessário. Possui recursos de suporte atuais das APIs, como armazenamento e recuperação de dados Os desenvolvedores têm acesso somente de leitura para o sistema de arquivos no *App Engine*.

2.6.4 Vmware

É líder global em tecnologia de infraestrutura virtual, ajudando as organizações grandes e pequenas para aumentar a eficiência e rentabilidade de suas operações de TI. Usando uma solução de virtualização VMware permite transformar os recursos de *hardware* de um computador baseado em x86 para criar uma máquina virtual totalmente funcional. (VMWARE, 2014).

A VMware se destaca pelo produto vSphere, que permite transferir uma máquina virtual inteira em execução de um servidor físico para outro, sem tempo de inatividade. A máquina virtual mantém sua identidade e suas conexões de rede, garantindo um processo de migração livre de problemas. Transfere a memória ativa e o estado de execução exato da máquina virtual através de uma rede de alta velocidade, permitindo que a máquina virtual alterne sua execução do host de origem para o host de destino. (VMWARE, 2014).

[24] *Failover*: caso um nó do *cluster* vier a falhar, aplicações ou serviços passam a estar disponíveis em outro nó.

2.6.5 OpenStack

OpenStack é um software de *open source* e computação em nuvem. O OpenStack é classificado como fornecedora de Infraestrutura como Serviço (IaaS), foi iniciado em um projeto em 2010 pela empresa Rackspace e pela agência espacial americana, a NASA. Depois de 4 anos do seu lançamento, mais de 150 empresas aderiram ao projeto, empresas como AMD, Intel, Canonical, SUSE Linux, Red Hat, IBM, Dell, HP e Cisco. O OpenStack desenvolve alguns projetos como o OpenStack *Compute*, que oferece recursos de computação entre máquinas virtuais e gerenciamento de rede, e OpenStack *Object Storage*, que proporciona um serviço de armazenamento de objetos escaláveis. (OPENSTACK, 2014).

O OpenStack pode ser usado por qualquer organização que procura implantar uma nuvem em grande escala, tanto para uso privado como para uso público. OpenStack é um projeto indicado para vários tipos de empresas: desde pequenas e médias empresas, como empresas do governo, grandes corporações, provedores de serviços e centros de dados. Basicamente o OpenStack é um *software open-source* usado para construir nuvens públicas e privadas. OpenStack representa tanto uma comunidade de um projeto de Software Livre, como um software para as organizações executarem suas próprias nuvens ou de armazenamento virtual. (OPENSTACK, 2014).

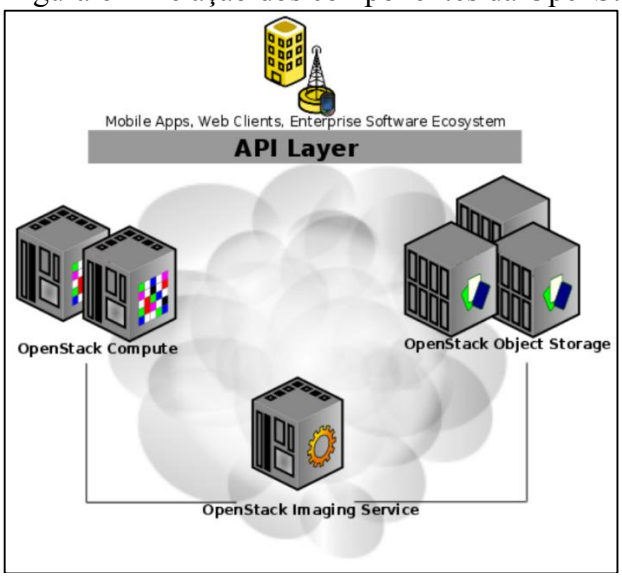
Do ponto de vista de *software*, OpenStack é uma coleção de projetos de código aberto mantido pela comunidade do OpenStack que incluem vários componentes, sendo os mais importantes:

- **OpenStack *Compute***: chamado de *Nova*, é o controlador da estrutura básica da nuvem. Ele é responsável por iniciar as instâncias (VMs) para os usuários. Esse serviço também é responsável pela gestão da rede virtual para cada instância que fazem parte de um projeto;
- **OpenStack *Object Storage***: chamado de *Swift*, é o serviço responsável pelo armazenamento de objetos. Existem várias aplicações, dentre as mais importantes são destacados o armazenamento de arquivo, backups, armazenamento de áudio e vídeo, desenvolvimento de novas aplicações com armazenamento integrado;
- **OpenStack *Image Service***: chamado de *Glance*, é um serviço de pesquisa e recuperação de imagens de máquinas virtuais. Este serviço pode salvar imagens diretamente ou usar mecanismos mais avançados, como o uso de armazenamento de objetos;

- **OpenStack *Dashboard***: é um painel web para gerenciar instâncias e volumes. Este serviço é realmente uma aplicação *web* que permite a comunicação com diferentes APIs.

A Figura 6 mostra as relações entre os componentes principais (*Nova*, *Glance* e *Swift*), como eles estão relacionados e como eles podem atender aos objetivos propostos para a implantação de infraestruturas de computação em nuvem OpenStack.

Figura 6 – Relação dos componentes da OpenStack.



Fonte: OpenStack (2014).

2.6.6 Outros Gestores

Segundo Mahjoub (2011) existe uma infinidade de getores de nuvem, cada um com suas característica e particularidades que estão apresentadas a seguir, e por fim um quadro comparativo é apresentado.

- ***Xen*:** É um Monitor de Máquina Virtual (VMM), originalmente desenvolvido pela Universidade de Cambridge;
- ***KVM* (*Kernel-based Virtual Machine*):** Consiste em criar um ambiente de VM tratados pelo kernel do Linux. Ele está incluído nas versões oficiais do kernel do linux desde a versão 2.6.20;
- ***OpenVZ*:** É um nível de sistema operacional produzido pela Virtuozzo. Alega ser altamente escalável, com isolamento forte entre os recipientes. A equipe OpenVZ está

contribuindo para o kernel Linux, limitando o número de recursos (memória e ações do processador);

- ***VirtualPC*:** Originalmente desenvolvido pela *Microsoft*, desenvolvido dentro do próprio sistema operacional, a fim de ter um desempenho mais eficiente do *hardware*;
- ***HyperV*:** É um sistema de virtualização baseada em sistemas de x86-64. Ele fornece uma plataforma de virtualização fundamental para usuários em trânsito em nuvem;
- ***Eucalyptus*:** É um *framework open source* de computação em nuvem comumente disponível para grupos de pesquisa acadêmica. Ele fornece uma plataforma modular e aberta à instrumentação experimental e estudo. A arquitetura do *Eucalyptus* é simples, flexível e modular com uma concepção hierárquica. Atualmente, ele suporta VMs que são executados com Xen, KVM e VMware;
- ***OpenNebula*:** *OpenNebula* é um toolkit de código aberto usado para construir nuvens privadas, públicas e híbridas. Ele funciona com Xen, KVM e solução de virtualização VMware, e está atualmente trabalhando no apoio VirtualBox;
- ***Nimbus*:** É mais uma solução de computação em nuvem fornecendo infraestrutura como serviço. Nimbus está ligado ao projeto Culumbus e suporta implementações de virtualização diferentes: Xen e KVM;
- ***Xen Cloud Platform (XCP)*:** É uma fonte aberta de virtualização de servidor empresarial e plataforma de computação em nuvem. Ele foi originalmente derivado de Citrix XenServer e está licenciado sob a GNU General Public License (GPL2);
- ***AbiCloud*:** É uma solução desenvolvida pela nuvem privada Abiquo. Ele permite aos usuários construir infraestrutura como um ambiente de serviço de nuvem. As técnicas de virtualização com suporte do AbiCloud são VirtualBox, VMWare, XEN e KVM;

2.7 Considerações Finais

Esse capítulo apresentou um levantamento sobre computação em nuvem, destacando as características de um ambiente em nuvens, um comparativo entre o modelo computacional tradicional e computação em nuvem, as diferentes arquiteturas, os tipos de serviços disponíveis em um ambiente de nuvem, os benefícios, riscos e desafios de um ambiente em nuvem.

O Quadro 2 resume quatro gestores de nuvem pagas, podendo destacar a classe de auto dimensionamento, onde as quatro nuvens possuem essa característica automática, demonstrando que as nuvens possuem tolerância a falhas. Essas nuvens são baseadas em diferentes níveis de abstração e de gestão dos recursos. Os usuários podem escolher um tipo de combinações de vários tipos de ofertas de nuvem para satisfazer necessidades comerciais específicas. (ZHANG, 2010).

Quadro 2 – Uma comparação de produtos comerciais representativos.

Provedor de Nuvem	Amazon EC2	Windows Azure	Google App Engine	VMware
Classe de utilização	Infraestrutura de Serviço	Plataforma de Serviço	Plataforma de Serviço	Infraestrutura de Serviço
Aplicações Alvo	Aplicações gerais	Aplicações gerais e aplicações *Windows*	Aplicações tradicionais web com estrutura suportada	Aplicações gerais
Computação	*OS Level on a Xen Virtual Machine*	*Microsoft Common Language Runtime* (CLR) VM; *Predefined roles of app. instances*	*Predefined web application frameworks*	*VMware vSphere*
Armazenamento	*Elastic Block Store; Amazon* Srviço Simples de Armazenamento (*Simple Storage Service* (S3)); *Amazon SimpleDB*	*Azure storage service and SQL* Serviço de dados	*BigTable and MegaStore*	*Virtual SAN*
Auto Dimensionamento	Automaticamente alterando o número de casos com base em parâmetros que os utilizadores especificam.	Dimensionamento automático com base em funções de aplicativo e um arquivo de configuração especificado pelos usuários	Escala automática que é transparente para os usuário	Automaticamente escalável

Fonte: Adaptado de Zhang (2010).

O Quadro 3 apresenta uma síntese entre os principais gestores de nuvem analisados neste capítulo, demonstrando informações como tipo de virtualizações, plataformas e arquiteturas compatíveis, linguagens que foram desenvolvidas, formas de armazenamentos de dados, algumas compatibilidades e quais empresas utilizam a nuvem. Entre as nuvens abordadas, vale ressaltar os seguintes critérios da nuvem OpenStack: produzido pela Rackspace e a NASA, interface de acesso via web, linguagem de desenvolvimento em python, tolerância a falhas em objetos e arquivos em múltiplos discos espalhados pelos servidores em um datacenter, e não tem migrações *online* de máquinas virtuais.

Quadro 3 – Comparação de gestores de computação em nuvem.

		OpenVZ	Xen	KVM	VirtualBox	VMware	HyperV
Desenvolvido por		SWsoft	Universidade de Cambridge	Qumranet in Israel	Innotek (Sun)	VMware	Microsoft
SO's Compatíveis	**Host**	Todas distribuições Linux	Linux (algumas versões)	Linux incluindo Kernel do Linux	- Todas Versões Windows w Linux - Macintosh - OpenSolaris	- Maioria das versões Windows e Linux - MacOS	- Windows 7, Vista e XP - SUSE Linux Enterprise Server, RedHat Enterprise Linux
	Guest	Linux (mesmos SO's do host)	- Todas versões Linux e Windows - Solaris	Todos SO's	- Todas Versões Windows w Linux - Solaris - OpenSolaris	- Todas Versões Windows - Linux (algumas versões) - Solaris	- Windows 7, Vista e XP - SUSE Linux Enterprise Server, RedHat Enterprise Linux
Tipos de Virtualizações		Isolamento	- Para-virtualização (SO modificado) - Full-virtualização (tecnologias IntelVT e AMD-V)	Full-virtualização (SO malterado)	Full-virtualização (SO malterado)	Full-virtualização (SO malterado)	Para-virtualização
Plataforma		- x86 (32 bits) - AMD64/Intel64 (64 bits)	- x86 (32 bits) - x64 (64 bits) - PowerAMC & ARM	x86 (32 bits)	- x86 (32 bits) - AMD64/Intel64 (64 bits)	x64 (64 bits)	- x86 (32 bits) - x64 (64 bits)
Virtualização habilitada		Com/Sem	Com/Sem	com	Com/Sem	Com/Sem	Com
Migrações on-line		Sim	Sim	Sim	Sim	Sim (VMware, VMotion)	Somente com Compartilhamento de Volume de Cluster
Nuvens			Amazon, Cloud.com, Rackspace			Google,Salesforce	
Free/Shareware		Licença GLP	Licença GLP	Licença GLP	Licença GLP	Shareware	Shareware

		Eucalyptus	OpenNebula	Nimbus	Plataforma de Nuvem Xen	AbiCloud	OpenStack
Produzido por		- Universidade de Santa Barbara - Empresa Eucalyptus	União Europeia	Universidade de Chigado	Citrix XenServer	Abico	Rackspace, NASA, Dell, Citrix, Cisco, Canonical
Objetivo Principal		Nuvem EC2	Construir nuvem privada	Soluções científicas	Evolução do Citrix XenServer	Gerenciamento de nuvem	Oferecer serviços
Usuários		Empresas	Pesquisas sobre nuvem e virtualização	Comunidade científica	Empresas	Empresas	Empresas, prestadores de serviços e pesquisas
SO's Suportados		Linux (Ubuntu, Fedora, CentOS, OpenSUSE e Debian)	Linux (Ubuntu, RedHat, Fedora e SUSE Linux Enterprise Server)	Maioria das distribuições Linux	- Linux (Ubuntu, RedHat, CentOS e SUSE Linux Enterprise Server) - Windows 7	- Linux (Ubuntu e CentOS) - Windows XP - Mac OS	- Linux - Windows - Requer x86 Server
Arquiteturas		- Hierárquica - Cinco Componentes - Mínimo de dois servidores	- Centralizado - Três Componentes - Mínimo dois servidores	- Centralizado - Três Componentes - Mínimo dois servidores	- Centralizado - Três Componentes - Mínimo dois servidores	- Centralizado - Três Componentes - Mínimo dois servidores	- Integração de objetos OpenStack e computação OpenStack
Linguagem		Java, C e python	Java, Ruby e C++	Phyton e java	Caml	Java, Ruby, C++ e python	python
Armazenamento		Walrus	- SCP - SQLite3	-GridFTP, Comulus (nova versão do GridFTP) - SCP	VastSky	HDFS	Armazenamento OpenStack
Rede		DHCP Server no controlador de cluster	Configuração manual	DHCP Server instalado no nodes	Open vSwitch	WSManagement	OpenStack Compute
Interface de acesso		- EC2 WS API - Ferramentas: HybridFox e ElasticFox	- EC2 WS API - OCCI API	- EC2 WS API - Nimbus WSRF	Linha de comandos <<XE>> (XenCenter and Versiera)	Interface Web com Adobe Flex	Interface Web
Login	**Usuário**	- Arquivos ZIP contendo certificações - Conexões HTTP	Autenticação	Certificado X509	- Autenticação - Conexões SSH	Autenticação (senha armazenada em formato MD5)	Certificado
	Administrador	- Conexão SSH - Root requerido	Root (somente se necessário)	- Conexões SSL - Integrate Globus (Certificado)	- Conexão SSH	Autenticação	Certificado
Balanceamento de carga		Controlador de nuvem	Nginx	Le contexto broker	XAPI	AbiServer	Controlador de nuvem
Tolerância a falhas		Separação de Cluster controlados	Backend de Banco de Dados (registra informações de máquina virtual)	Verificação periódica nos nodes da nuvem	Sincronização de máquinas virtuais	X	Replicação
Migrações on-line		X	Compartilhamento FS	X	- Formato de virtualização aberta - Compartilhamento de armazenamento		
Localização de VM's		Controlador Node	Cluster Node	Nodes Físicos	XCP Host	Cluster Node	OpenStack Compute
Compatibilidade com EC2		Sim	Sim	Sim	Sim	X	X
Usado por		NASA	Reservoir Project, NUBA	STAR		Ativo em Span	

Fonte: Adaptado de MAHJOUB (2011).

3 – Tolerância a Falhas

Esse capítulo apresenta o conceito de tolerância a falha, os modelos de falhas e tolerância a falhas na computação em nuvem.

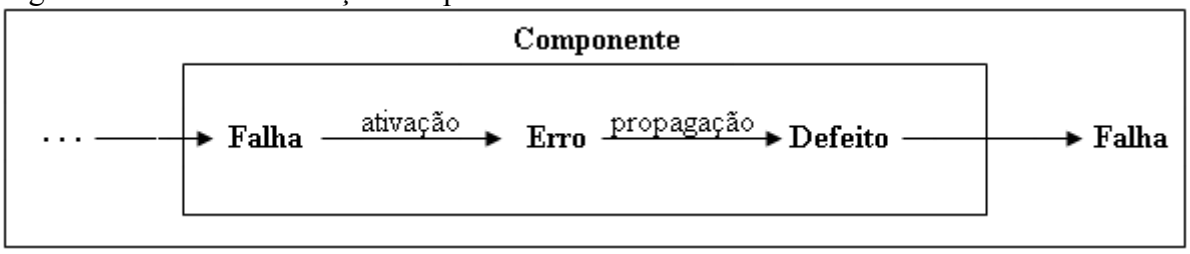
3.1 Considerações iniciais

Para que um sistema de computação funcione corretamente, deve ser muito bem especificado, mas quando ocorre um desvio nessa especificação, significa que existe um defeito (*failure*). Como por exemplo, quando um cliente necessita de um serviço, e esse serviço não está disponível ao cliente devido ao defeito. Dentro desse conceito, um erro, pode ser considerado um estado do sistema cujo processamento posterior a ele poderá levar a um estado não esperado (defeito). E uma falha (*fault*), é a causa do erro. (TANENBAUM; STEEN, 2007).

Uma falha pode ser iniciada por um processo de serviço incorreto até chegar a circunstância de defeito. Seja qual for a natureza de uma falha, ela pode gerar um erro ou não. Uma falha que gera um erro é considerada uma falha ativa, caso contrário, é considerada uma falha latente. Para um erro se tornar ativo, é necessário que atinja um estado de defeito, caso

contrário o erro é considerado latente. Quando um defeito ocorre, o sistema não tem capacidade de entregar corretamente o resultado esperado, e nesse caso entre em um estado de serviço incorreto. Sendo assim, um sistema pode atingir uma situação de falha e posteriormente um erro, mas não chegar ao grau de defeito, entregando mesmo assim o serviço corretamente. A Figura 7 ilustra a cadeia de ameaça da dependabilidade apresentada em Dantas (2005).

Figura 7 - Cadeia de ameaça da dependabilidade.



Fonte: Adaptado de DANTAS (2005).

Segundo Dantas (2005), os tipos de falhas e suas origens são muito variados. Um exemplo, é o mau funcionamento do sistema de ventilação de um computador pessoal, considerado uma falha. Como consequência, caso algum tipo de erro ocorra no funcionamento do equipamento, poderá superaquecer os sistemas, e assim apresentar um defeito. Porém, o mau funcionamento do sistema de ventilação (falha) pode não ser contínuo e não causar a elevação de temperatura (erro) do computador, não causando um mau funcionamento do sistema computacional (defeito).

O termo tolerância a falhas tem sido utilizado em pesquisas acadêmicas para descrever sobre o comportamento de sistemas computacionais. Já na indústria de desenvolvimento de sistemas, o termo não tem muita aceitação, pois os desenvolvedores preferem utilizar o termo “sistemas redundantes”. Na comercialização de sistemas, o termo utilizado é “alta disponibilidade”. Independente do termo utilizado, uma característica comum que todos eles devem ter, é a transparência a falhas para o usuário final. Pois para o usuário uma coisa que lhe importa é que o sistema esteja sempre “no ar”.

Tolerância a falhas é uma questão fundamental de um projeto de computação, pois a tolerância a falhas pode ser relacionada com um sistema que está mascarando uma falha, ou melhor, que não deixa os usuários de um sistema perceberem que algum problema está ocorrendo, deixando o sistema sempre funcionando.

Segundo Tanenbaum e Steen (2007) existem vários tipos de falhas: desde uma falha por queda de um sistema, onde todo o processo de uma empresa pode parar, passando por falhas por omissão, onde um processo não responde a requisições, ou quando um processo responde muito cedo ou muito tarde a uma requisição, até responder uma requisição que chega, mas de modo errado.

Ainda segundo Tanenbaum e Steen (2007), os requisitos úteis relacionados a sistemas distribuídos tolerantes a falhas são:

- **Disponibilidade**: definida como a propriedade de um sistema estar sempre pronto para ser usado imediatamente. Ou seja, um sistema de alta disponibilidade é aquele que sempre estará funcionando no momento que for requisitado;
- **Confiabilidade**: define como a propriedade de um sistema funcionar continuamente sem falhas. Portanto um sistema de alta confiabilidade é definido como aquele que funcionará corretamente sem interrupções durante um tempo relativamente longo;
- **Segurança**: define como propriedade, se por acaso um sistema falhar, durante certo tempo, nada de catastrófico acontecerá;
- **Capacidade de manutenção**: se refere a facilidade de realizar uma manutenção em um sistema caso ele falhe.

Tanenbaum e Steen (2007) definem ainda que um sistema pode prover seus serviços mesmo na presença de falhas. Isso significa que um sistema pode apresentar algumas falhas mais ainda assim funcionar normalmente. E as falhas podem ser classificadas em:

- **Falhas transientes**: ocorrem uma vez e depois desaparecem. Se a operação for repetida, a falha não acontecerá novamente. Se a temporização de uma transmissão se esgotar e for executada novamente, provavelmente funcionará desta segunda vez;
- **Falhas intermitentes**: ocorre e desaparece por sua própria vontade, depois reaparece, e assim por diante. Falhas intermitentes são difíceis de diagnosticar;
- **Falhas permanentes**: é aquela que continua a existir até que o componente faltoso seja substituído.

3.2 Modelos de falhas

Quando um sistema apresenta algum tipo de falha, quer dizer que o sistema não está oferecendo os serviços de acordo com o projetado. Entretanto, nem sempre o problema está nos servidores que estão provendo serviço, pois pode ser que tal serviço dependa de outros servidores para prestar o serviço adequadamente. (TANENBAUM; STEEN, 2007).

Tanenbaum e Steen (2007) descrevem diversos tipos de falhas como:

- **Falha por queda**: ocorre quando um servidor interrompe seu funcionamento prematuramente. Um exemplo de falhas por queda é quando o sistema operacional pára de repente e a única solução é reinicializá-lo;
- **Falha por omissão**: ocorre quando um servidor deixa de responder a uma requisição. Essa falha pode ser pelo motivo do servidor não ter recebido nenhuma requisição, ou que o servidor enviou uma requisição, mas o outro lado não recebeu;
- **Falha de temporização**: ocorre quando a resposta se encontra fora de um intervalo de tempo real especificado. Essa falha pode ser classificada como falha de desempenho;
- **Falha de resposta**: ocorre quando a resposta de um servidor está incorreta. Por exemplo, um mecanismo de busca que retorne sistematicamente páginas *Web* não relacionadas com qualquer uma das palavras de busca usada;
- **Falhas arbitrárias**: conhecida como falhas bizantinas, quando ocorrem essas falhas, o cliente deve se preparar para o pior, pois pode acontecer de um servidor estar produzindo saídas que nunca deveria ter produzido, mas que não podem ser detectadas como incorretas.

3.3 Técnicas de Tolerância a Falhas

Para um sistema ser considerado como tolerante a falhas, não basta se preocupar somente com os requisitos de disponibilidade, confiabilidade e segurança, o sistema deve ser construído utilizando técnicas tolerantes a falhas. Essas técnicas podem garantir que o sistema

tenha um bom funcionamento mesmo na ocorrência de falhas, pois serão baseadas em redundância, exigindo alguns componentes adicionais ou algoritmos especiais.

Segundo Anderson e Lee (1981), as técnicas de tolerância a falhas podem ser classificadas em quatro fases: detecção, avaliação, recuperação e tratamento. O Quadro 4 apresenta as 4 fases de aplicações para tolerância a falhas. Essas fases estão relacionadas em uma sequência de atividades, que devem ser executadas após a identificação de uma ou mais falhas, e são descritos a seguir.

Quadro 4 – Fases para aplicação das técnicas de tolerância a falhas.

Fases	**Mecanismos**
Detecção de erros	Duplicação e comparação Testes de limites de tempo Cão de guarda (*watchdog timers*) Testes reversos Codificação: paridade e códigos de detecção de erros Teste de razoabilidade, de limites e de compatibilidades Testes estruturais e de consistência Pré-Diagnóstico
Avaliação de danos	Ações atômicas Operações primitivas auto encapsuladas Isolamento de processos Regras do tipo tudo que não é permitido é proibido Hierarquia de processos Controle de recursos
Recuperação do erro	Técnicas de recuperação por retorno (*backward error recovery)* Técnicas de recuperação por avanço (*forward error recovery)*
Tratamento da falha e continuidade do serviço	Diagnóstico Reparo

Fonte: Adaptado de Anderson e Lee (1981).

- **Fases de detecção de erro**: A primeira fase é da detecção de erro, onde uma falha se manifesta inicialmente como um erro, para em seguida ser detectada pelos mecanismos propostos no Quadro 4. Sendo que uma falha não pode ser detectada se o erro não se manifestar; uma falha pode estar em um sistema por toda a sua vida útil se o sistema não entrar em um estado errôneo, dentre os mecanismos nessa fase, destaca-se um pré-diagnóstico do erro detectado, que se confirmará na fase de tratamento da falha;
- **Fase de Avaliação**: Na fase de avaliação serão levantadas as informações necessárias, para uma correção e futura recuperação do mesmo;

- **Fase de Recuperação**: São utilizadas técnicas de recuperação por retorno, onde ocorre a condução do estado do sistema para anteriormente ao erro, e técnicas de recuperação por avanço, onde a condução a um novo estado ainda não ocorreu desde a última manifestação de erro, sendo utilizado em sistema de tempo real;
- **Fase de Tratamento**: Nessa fase é considerado que uma falha única ocorra de cada vez, e a localização da falha é realizada em duas etapas: localização rápida e localização demorada. Para cada um dos tipos são utilizados diagnósticos manuais ou automáticos. Após a localização a falha é reparada pela remoção do componente danificado, e nessa fase é possível informar o diagnóstico da falha corrigida.

3.4 Tolerância a Falhas na Computação em Nuvem

A computação em nuvem utiliza a virtualização de recursos computacionais disponíveis como máquinas virtuais, armazenamento virtual e redes virtuais. No caso de serviços de computação e do uso de máquinas virtuais, o *backup* é um método que não pode faltar em sistemas virtualizados, uma vez que a imagem virtual contém tudo o que é necessário para executar o aplicativo e pode ser transparente para migração entre máquinas físicas. Uma das desvantagens da virtualização é que uma falha de *hardware* em um servidor físico pode afetar várias máquinas virtuais e aplicações. *Backups* devem sempre ser implantados em diferentes máquinas físicas. Os recursos de reserva devem também ser dimensionados caso um grande número de máquinas virtuais falhem ou ocorram falhas em um servidor físico. (UNDHEIM et al. 2011).

Segundo Jhawar et al. (2012), a computação em nuvem vem ganhando uma crescente popularidade em relação aos sistemas de processamento de informação clássicos para armazenar e processar grandes volumes de dados. Na maioria das vezes, estes centros de dados são virtualizados, e recursos de computação são provisionados para o usuário como um dos serviços *on-demand* através da *internet* na forma de máquinas virtuais. Estes ambientes prometem vários benefícios, tais como alta escalabilidade, mobilidade, menor entrada e custos de utilização, e uma aparência de disponibilidade de recursos infinitos para o usuário individual.

Por outro lado, a confiabilidade da computação em nuvem ainda é uma grande preocupação entre os usuários. Devido às pressões econômicas, estas infraestruturas de computação costumam expor as condições dos *hardwares* em escalas que não foram originalmente concebidos. Como resultado, um grande número de falhas nos sistemas começam a prejudicar as aplicações hospedadas, afetando a sua disponibilidade e o desempenho. (JHAWAR et al., 2012).

Neste contexto, as aplicações requerem habilidades de tolerância a falhas, para que possam superar o impacto de falhas no sistema e executar suas funções corretamente quando as falhas acontecerem. Atualmente, a maioria dos modelos de entrega em nuvem exigem que os desenvolvedores de aplicações construam *software* confiáveis tendo em conta características específicas do ambiente. No entanto, esta abordagem tem algumas limitações. Em primeiro lugar, exige experiência e conhecimento dos usuários em relação a integração e configuração de aplicações com estruturas de tolerância disponíveis a falha. Em segundo lugar, o gerenciamento das aplicações durante todo o período em que a falha estiver ocorrendo. Por último, exige muito esforço dos desenvolvedores para que a arquitetura da nuvem seja um ambiente transparente e flexível durante o período da falha. Portanto, é necessário obter um aumento na gestão das propriedades de tolerância a falhas para as aplicações hospedadas no ambiente de computação da nuvem. (JHAWAR et al., 2012).

Tchana et al. (2012) descrevem técnicas de virtualização, que são usadas em plataformas de nuvem para implementar particionamento de recursos. Esses recursos são apresentados em três camadas de infraestrutura de uma nuvem: a camada de recursos físicos (contendo os recursos da nuvem), a camada de virtualização (contendo as máquinas virtuais) e a camada de aplicações (contendo as aplicações das empresas). E de acordo com a arquitetura de uma nuvem, são identificados três tipos de falhas em uma plataforma de nuvem: falha de *hardware*, falha nas máquinas virtuais e falha do aplicativo.

A estratégia de tolerância a falhas inclui duas fases distintas para o gerenciamento de falhas: fase de detecção e fase de reparo. Uma das dificuldades para implementar tolerância a falhas em uma arquitetura de nuvem é definir a melhor forma de implementar as duas fases de gerenciamento de falhas, de acordo com os seus direitos de acesso na arquitetura da nuvem. Entretanto, é razoável deixar responsabilidade exclusiva de tolerância a falhas com um participante da nuvem sabendo que falhas de *hardware* só podem ser detectadas e reparadas pelo fornecedor de nuvem; falhas de máquinas virtuais podem ser detectadas pelos dois

participantes, mas apenas reparados pelo provedor de nuvem e falhas de aplicação só podem ser detectadas pelo cliente, mas podem ser reparados pelos dois participantes. (TCHANA et al., 2012).

3.5 Considerações Finais

Sistemas distribuídos são vulneráveis por diferentes tipos de falhas. Este capítulo apresentou as técnicas existentes para garantir o bom funcionamento do sistema mesmo na ocorrência de falhas. A computação em nuvem herda alguns dos desafios de sistemas distribuídos além de enfrenta muitos desafios próprios. Desafios como a alta disponibilidade, que são solucionadas com o desenvolvimento de mecanismos de tolerância a falhas. No capítulo seguinte é apresentado o desenvolvimento do mecanismo de tolerância a falha OpenStackTF, assim como os resultados coletados nos testes.

4 – Desenvolvimento do Mecanismo para Tolerância a Falhas e Resultados

Este trabalho fundamenta-se em uma pesquisa exploratória e foi alicerçada por meio de levantamento bibliográfico, constituído principalmente de livros e artigos científicos.

Este capítulo apresenta os materiais e métodos para o desenvolvimento do mecanismo de tolerância a falhas OpenStackTF. Posteriormente será demonstrado como o mesmo foi implementado, os testes desenvolvidos e os resultados finais obtidos.

4.1 Considerações iniciais

Como um ambiente de sistema distribuído, a computação em nuvem está sujeita a falhas, e técnicas de tolerância a falhas devem ser utilizadas para aumentar a disponibilidade desses ambientes. De acordo com esse contexto e conforme apresentado nos capítulos anteriores foram realizadas pesquisas das nuvens *open source* disponíveis, e os três projetos mais evidenciados foram o Eucalyptus, OpenNebula e OpenStack. Em seguida foi criado um ambiente com os três gestores, nesse ambiente foram utilizados 4 computadores com as mesmas configurações, sendo que em um computador foi instalado o sistema operacional *Linux Desktop*, para ser instalado os gestores das nuvens, e as outras três máquinas com o sistema operacional *Ubuntu Linux Server*.

Após realizar testes nos três gestores, foi escolhido o OpenStack, pelos fatores:

- Comunidade envolvida no projeto, bem como o grande número de fabricantes apoiando e suportando o projeto;
- Adoção em larga escala no mercado de nuvem, e pela utilização em empresas como HP, Rackspace, eBay, NASA, PayPal dentre outras;
- *Dashboard* de administração que fornece as ferramentas necessárias para provisionamento e gerenciamento dos recursos computacionais.

Durante as fases de pesquisa no OpenStack, foram realizados vários testes no ambiente, para tentar identificar deficiências, e foram encontrados os seguintes pontos: (BACHIEGA et al. 2013).

- **Tolerância a falhas**: Não tem tolerância a falha, tanto para o gestor quanto para o *node*. Nesse caso, se ocorrer alguma falha com o gestor, os *nodes* continuam funcionando normalmente, mas se o *node* apresentar algum problema, a execução daquela instância é perdida, dependendo assim de uma intervenção humana, tendo que carregar uma instância de VM;
- **Algoritmo de Escalonamento**: Possui dois algoritmos para escalonamento de instâncias, o primeiro é o ROUNDROBIN (intercalado) e o segundo GREEDY (aloca a instância no primeiro nó encontrado). O gestor poderia determinar de uma maneira melhor e técnica quem são seus nodes e realizar o escalonamento de acordo com o consumo da aplicação ou escalonar caso perceba que um node esteja excedendo quantidade utilizada de memória e CPU;
- **Distâncias Geográficas**: No caso das nuvens estarem distribuídas geograficamente, o gestor deveria ter a opção de alternar o escalonamento das instâncias de VM de acordo com a localização e em qual tem mais acessos. E um dos problemas da virtualização de uma aplicação é que ela não está preparada para ser acessada através de uma rede;
- **Consumo Energético**: Para os gestores, uma grande contribuição para a questão de energia elétrica seria se os nós que estão inoperantes poderiam estar desligados ou colocados em baixo consumo de energia;
- **Proteção de Dados**: A política e controle de *backup* dos gestores apresentam algumas falhas. Outro problema é que na maioria das nuvens *open souce*, o gerenciamento de

armazenamento é de terceiros, e nesse caso torna-se dispendioso para a rede utilizar essas soluções.

Para essa pesquisa foi escolhido uma vulnerabilidade que é considerada crítica para o funcionamento da nuvem. O ponto escolhido foi o problema de tolerância a falha de *hardware* nos *nodes*, e foi criado o mecanismo de tolerância a falha intitulado de OpenStackTF, que será detalhado na seção 4.3.2.

4.2 Trabalhos Relacionados

Nuvem pagas como VMware, Amazon e Citrix implementam em suas distribuições soluções de tolerância a falhas.

Segundo a VMware (2014), o produto vSphere, fornece disponibilidade para todo um ambiente virtual de uma empresa, minimizando o tempo de inatividade não planejado por meio da reinicialização da máquina virtual, proporcionando um nível de alta disponibilidade. A VMware disponibiliza 5 produtos de recursos de alta disponibilidade como:

- ***High Availability***: A *High Availability* (HA) proporciona continuidade para aplicativos executados em máquinas virtuais. Em caso de falha do servidor físico, as máquinas virtuais afetadas são reiniciadas automaticamente em outros servidores com capacidade extra. Em caso de falha do sistema operacional, o vSphere HA reinicia as máquinas virtuais afetadas no mesmo servidor físico.
- ***Data Protection:*** O VMware vSphere *Data Protection* (VDP) é a solução de backup e recuperação que oferece backups eficientes em disco e recuperação confiável.
- ***App HA***: O vSphere App HA é um recurso que complementa a função do vSphere HA com monitoramento no nível do aplicativo e solução de problemas automatizada.
- ***Fault Tolerance***: O VMware vSphere *Fault Tolerance* (FT) fornece disponibilidade contínua para os aplicativos, protegendo-os contra falhas de hardware ao criar uma instância de sombra em tempo real de uma máquina virtual que é um equivalente virtual da instância primária. Com failover imediato entre duas instâncias, o FT elimina até mesmo a menor possibilidade de perda ou interrupção de dados.
- **Replicação**: O vSphere *Replication* permite replicar máquinas virtuais ligadas pela rede de um host do vSphere para outro sem replicação nativa no array de armazenamento.

As vantagens exclusivas incluem requisitos de largura de banda reduzidos, nenhum bloqueio de armazenamento e a capacidade de criar configurações de recuperação de desastres flexíveis.

De acordo com Amazon (2014), a AWS (*Amazon Web Services*) é uma ferramenta que permite criar sistemas tolerantes a falhas, e que sejam confiáveis, e com pouca intervenção humana. Os serviços *Amazon Elastic Compute Cloud* (EC2) e o A*mazon Elastic Block Store* (EBS), fornecem recursos como de alta disponibilidade.

Segundo Citrix (2014), o XenServer é uma solução de infraestrutura virtual completa, com interface de gerenciamento, recursos de migração ao vivo, e ferramentas para converter cargas de trabalho a partir de um ambiente físico para virtual. É possível gerenciar máquinas virtuais que podem ser executadas a partir de uma interface de gerenciamento. Permite que as máquinas virtuais ativas possam ser transferidas para um novo host físico, sem interrupções das suas aplicações gerando inatividade.

Entretanto, soluções *open source* como o OpenStack, não dispõe desses recursos, embora seja amplamente utilizada desta forma, é importante o estudo de solução que possam contribuir com essa deficiência.

4.3 Materiais

Para o desenvolvimento da pesquisa foi necessária a montagem de dois ambientes. O primeiro era composto de instalações originais do *open source;* ambiente no qual foram realizadas pesquisas para detectar as falhas existentes. No segundo foi implementado o mecanismo OpenStackTF, e realizado os testes.

4.3.1 Ambiente Inicial

Para o ambiente inicial de pesquisa foram utilizados 5 computadores da marca Lenovo ThinkCentre M58 3.0GHz 4GB 500GB, todos com o sistema operacional Linux Ubuntu

Server versão 12.04.2 i386 e dois switches de 8 portas. Todos os computadores foram configurados com duas placas de rede, uma placa para rede privada e a outra para acesso externo (SSH[25]), todos os equipamentos foram configurados de acordo com o Quadro 5.

Após a instalação do sistema operacional Linux, foram instalados em todos os computadores o OpenStack versão Folson 2012.2.1, conforme Apêndice C. Vale ressaltar que foi utilizada a versão disponível no site da OpenStack sem nenhuma configuração adicional, ou seja, livre de qualquer solução tolerante a falha.

Quadro 5 – Configuração dos computadores.

Computador	**Rede Privada (eth0)**	**Rede Pública (eth1)**
Controller	192.168.100.200	10.10.100.200
*node*01	192.168.100.201	10.10.100.201
*node*02	192.168.100.202	10.10.100.202
*node*03	192.168.100.203	10.10.100.203
*node*04	192.168.100.204	10.10.100.204

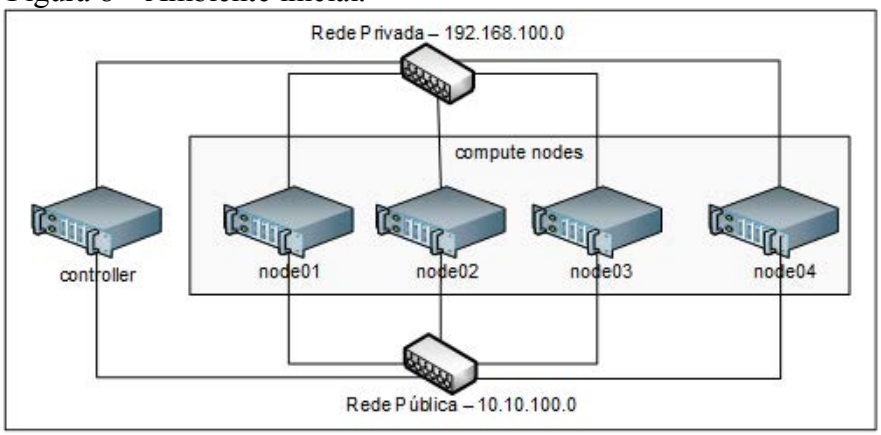
Fonte: Elaborado pelo autor.

A Figura 8 mostra a topologia da rede, onde o computador intitulado de *controller* é o gestor da nuvem, que tem o papel de gerenciar a configuração de rede das máquinas hospedeiras (*nodes*). Ele executa operações como alocação de endereços IP, implementação de grupos de segurança e configuração de redes para os nodes. É responsável por escalonar as máquinas virtuais nos *nodes*. As máquinas virtuais são requisitadas pelo usuário, e são identificadas no OpenStack como instância, que pode ser utilizada pelo usuário até o momento de descarte, ou caso ocorra algum problema com o *node* que a máquina virtual esteja alocada.

Analisando a Figura 8 é possível apontar algumas deficiências existentes; se o *controller* falhar, todo o ambiente fica comprometido deixando assim a nuvem fora de operação até que seja detectado e resolvido o problema. Outro problema a destacar é, se algum *switch* falhar todo o ambiente será prejudicado. Os *nodes* também podem apresentar falhas e prejudicar o usuário que estava utilizando-o, pois os *nodes* são responsáveis em alocar as máquinas virtuais dos usuários e, caso alguma falha ocorra nos *nodes*, as máquinas virtuais devem ser descartadas.

[25] SSH (*Secure Shell*): é um protocolo/programa, que permitem a conexão com outro computador que permite executar comandos de uma unidade remota.

Figura 8 – Ambiente inicial.

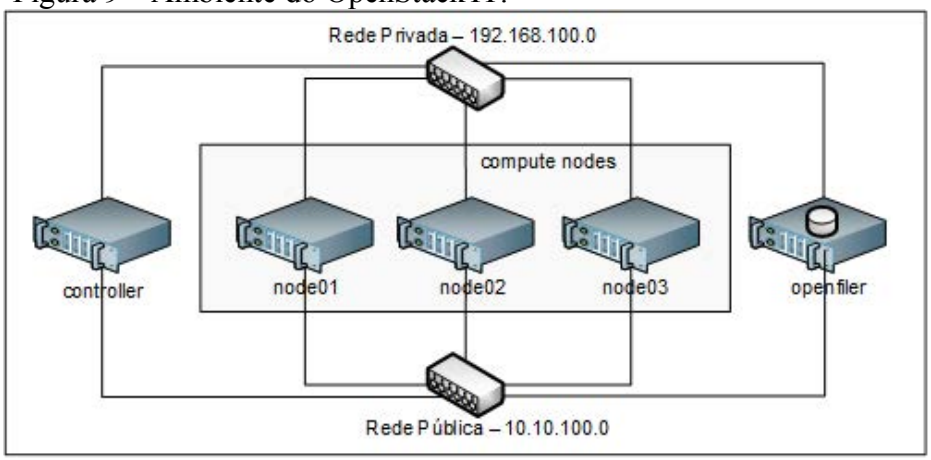


Fonte: Elaborado pelo autor.

4.3.2 Ambiente do Mecanismo OpenStackTF

Para o ambiente de desenvolvimento do OpenStackTF foram utilizados os mesmos 5 computadores do ambiente inicial com apenas uma mudança. Conforme a Figura 9 demonstra, nesse ambiente um *node* foi substituído pelo OpenFiler, para desempenhar o papel de *storage* das instâncias que são iniciadas pelo OpenStack.

Figura 9 – Ambiente do OpenStackTF.



Fonte: Elaborada pelo autor.

Para a implantação do OpenStackTF foi utilizado um *node* com o sistema operacional Linux OpenFiler, que é *apliance open source* para *storage* NAS (*Network Attached Storage*) e SAN (*Storage Area Network*). Esta distribuição fornece acesso aos dispositivos de armazenamento de dados via rede construído sobre a meta distribuição Linux *rPath* sendo distribuído como Linux *stand-alone*.

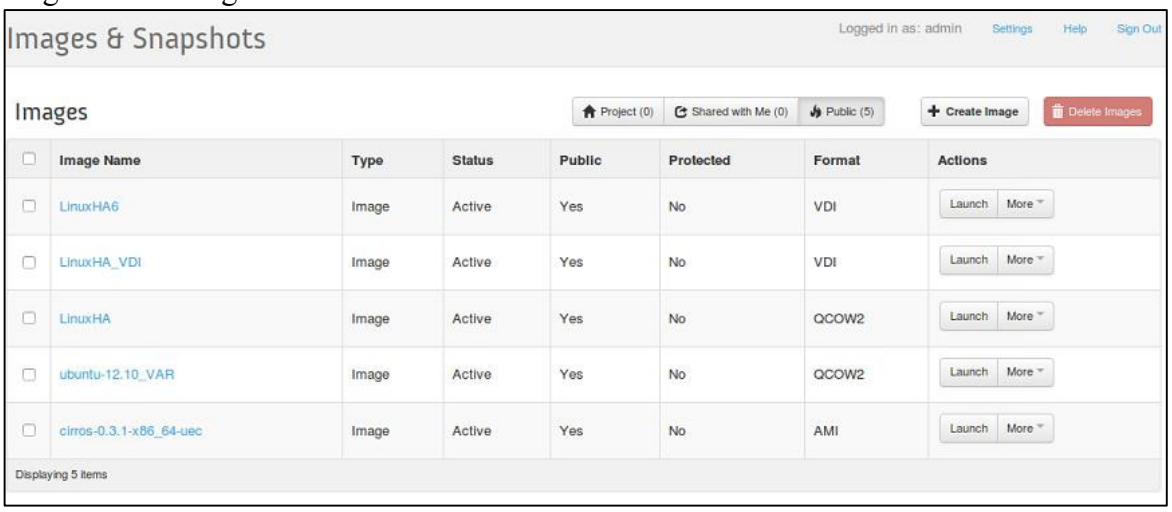
Foi escolhida a distribuição OpenFiler, pois possui suporte para particionamento baseado em volume iSCSI (*Internet Small Computer System Interface*) e gerenciamento das configurações via *web*. Outro fator que justifica a escolha do iSCSI foram os testes realizados com outras ferramentas que não funcionaram no OpenStack, no caso o DRBD (*Distributed Replicated Block Device*), NFS (*Network File System*) e RSync.

O iSCSI é usado para conectar dispositivos de armazenamento através de uma rede por meio de TCP/IP. Ele pode ser usado através de uma rede local (LAN), de uma rede de longa distância (WAN) ou da Internet. Os dispositivos iSCSI incluem discos, fitas, CDs e outros dispositivos de armazenamento em outro computador em rede ao qual se pode conectar. Algumas vezes esses dispositivos de armazenamento fazem parte de uma rede chamada rede de área de armazenamento (SAN). Na relação entre o computador e o dispositivo de armazenamento, o computador é chamado de iniciador (*Initiator*) por iniciar a conexão com o dispositivo que é chamado de destino (*Target*). O apêndice D apresenta a configuração realizada nessa pesquisa no OpenFiler. (MICROSOFT, 2014).

4.4 Métodos

Para simular a requisição de uma máquina virtual por um usuário, foi utilizado o Horizon (OpenStack *Dashboard*), módulo que consiste de uma interface gráfica *web* para os usuários e administradores de um sistema que utilizam o OpenStack. Através dessa interface um usuário pode criar uma máquina virtual e gerenciar o OpenStack. A Figura 10 mostra uma tela do Horizon, que ilustra as imagens disponíveis para serem utilizadas como máquinas virtuais. Todas as imagens foram criadas pelo autor dessa pesquisa. Para iniciar uma máquina virtual deve-se clicar no botão *Launch* da coluna *Actions*.

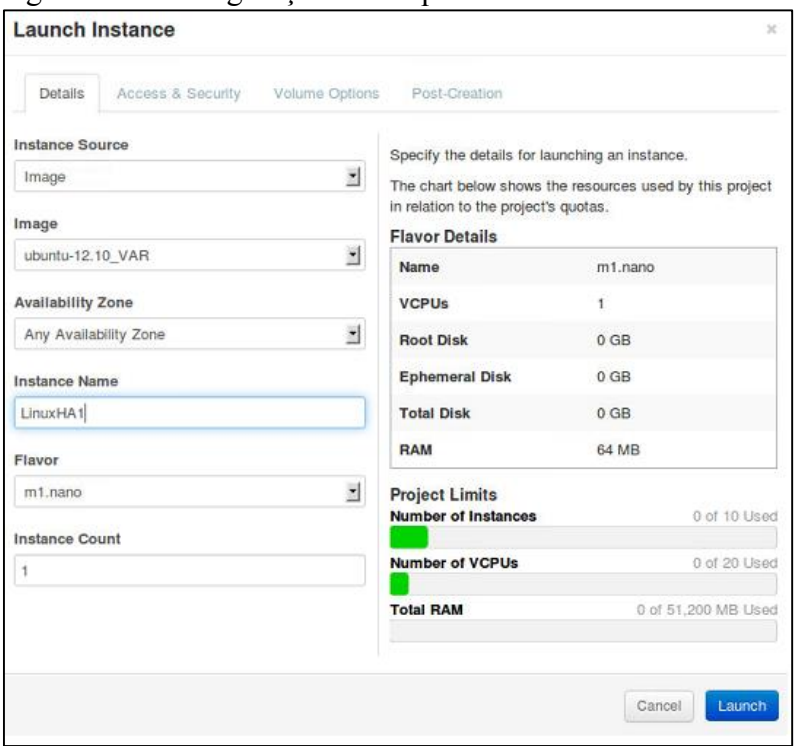
Figura 10 – Imagens ativas no ambiente.



Fonte: Elaborado pelo autor.

Após escolher a imagem que deseja ser iniciada, foi necessário entrar com algumas configurações conforme mostra a Figura 11, como nomear uma máquina virtual pelo campo *Instance Name*, escolher a imagem que será criada (*Image*) e a configuração da instância (*Flavor*). Ainda na Figura 11 é possível verificar qual a limitação do ambiente, como o número de instâncias que podem ser criadas, número de processamentos virtuais e total de memória RAM disponível.

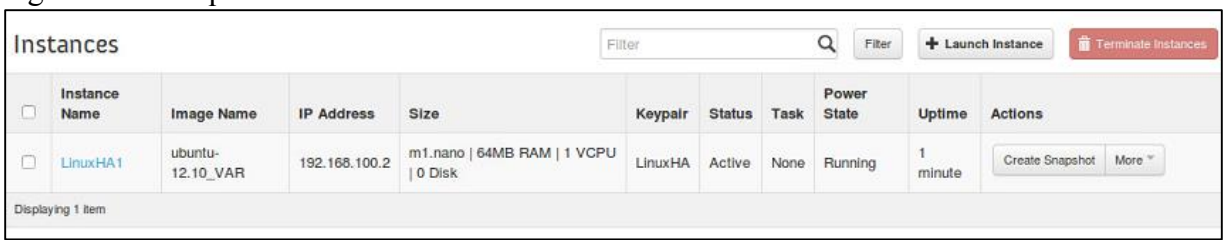
Figura 11 – Configuração da máquina virtual.



Fonte: Elaborado pelo autor.

Conforme ilustra a Figura 12, uma máquina virtual foi iniciada e recebeu o IP virtual 192.168.100.2, que foi alocado pelo *controller*. É possível identificar o status da máquina e o total de tempo que permanece ligada.

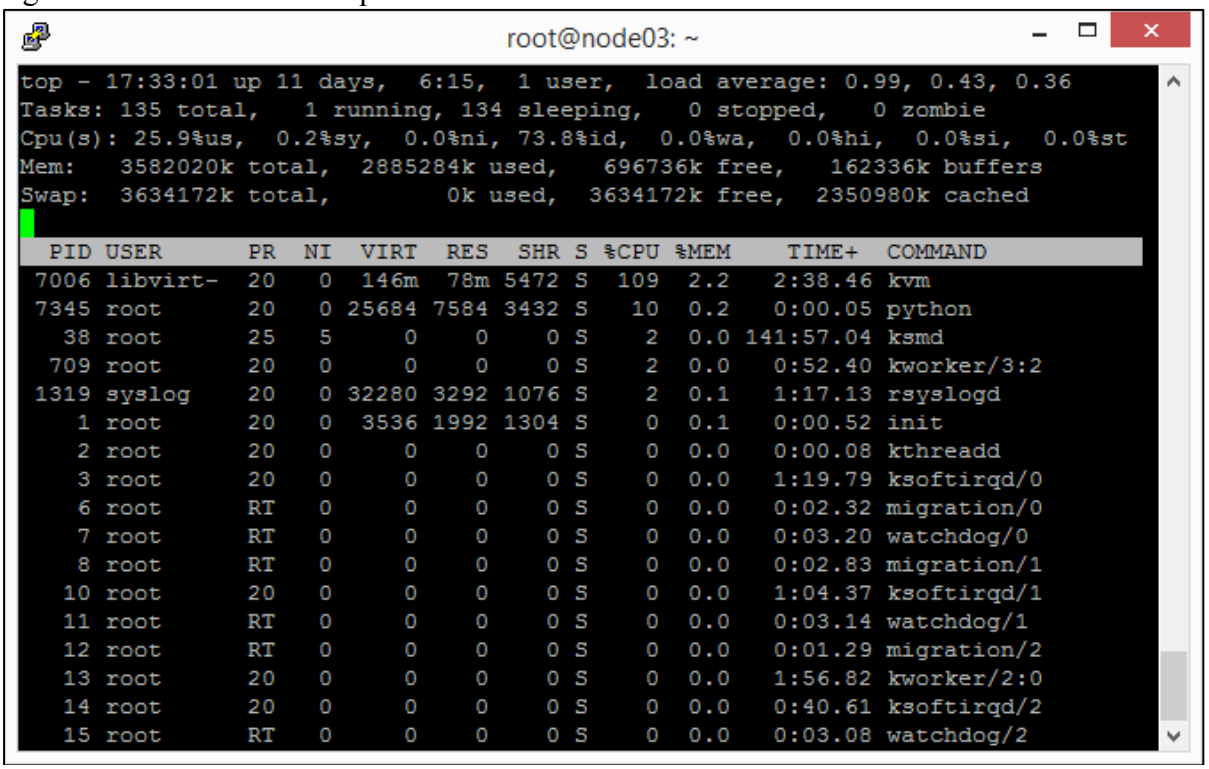
Figura 12 – Máquina virtual ativa.



Fonte: Elaborado pelo autor.

De acordo com a Figura 13, quando uma máquina virtual está ativa um processo é iniciado no *node* que foi alocado, esse processo pode ser identificado com o nome de kvm na coluna COMMAND.

Figura 13 – Processo da máquina virtual.



Fonte: Elaborado pelo autor.

A partir do momento que a máquina virtual é iniciada no *node*, a mesma está vulnerável as falhas. Caso o *node* apresente algum tipo de falha de *hardware*, a máquina

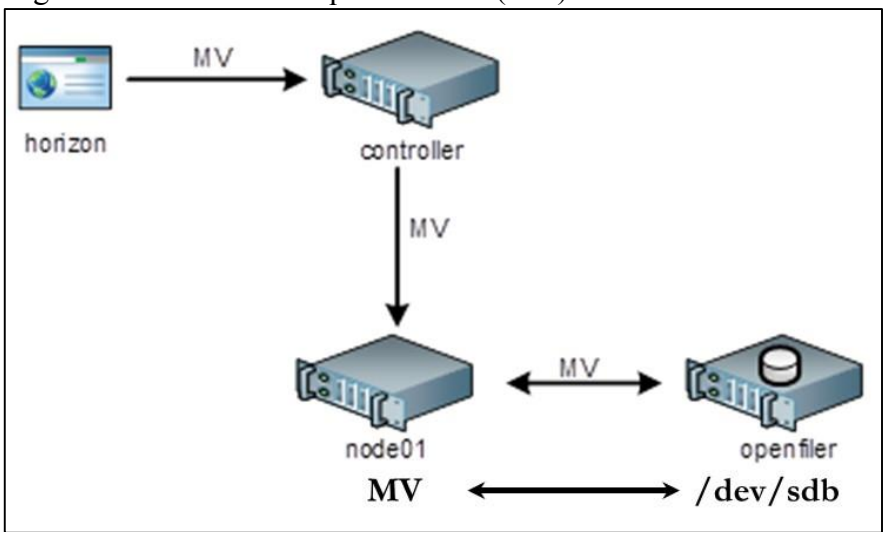
virtual não pode ser aproveitada e deve ser descartada pelo administrador da nuvem, fazendo com que o usuário perca sua máquina virtual.

Para solucionar esse problema foi implementado o mecanismo OpenStackTF. Este mecanismo criou um ambiente tolerante a falhas nas máquinas virtuais instanciadas nos *node*. É importante ressaltar a validade do mecanismo caso a falha for classificada como transiente ou intermitente.

4.5 Desenvolvimento do Mecanismo de Tolerância a Falhas - OpenStackTF

A Figura 14 mostra o fluxo de uma máquina virtual quando é instanciada pelo *horizon*. Assim que a máquina virtual é instanciada em um *node* disponível, o protocolo iSCSI que está configurado e ativo no *node*, replica no OpenFiler criando uma redundância da máquina virtual em um diretório disponível para o *node*, no exemplo da Figura 14, a máquina virtual instanciada pelo *node01*, será armazenada no diretório **/dev/sdb** do OpenFiler. Essa máquina virtual que foi enviada para o OpenFiler, é atualizada constantemente pelo iSCSI, até que uma falha ocorra no *node*. Caso algum erro ocorra no *node*, a máquina virtual fica em espera no OpenFiler até que o *node* se recupere da falha.

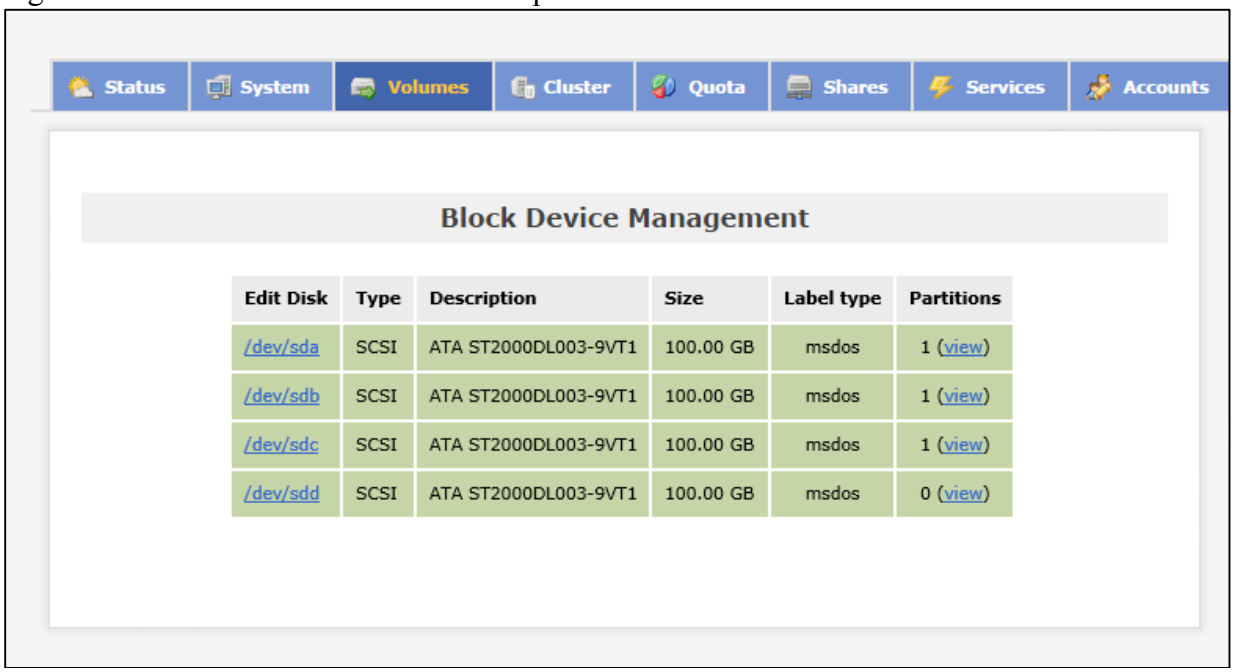
Figura 14 - Fluxo da Máquina Virtual (MV) no ambiente.



Fonte: Elaborada pelo autor.

A Figura 15 mostra a configuração de discos realizada no OpenFiler. Na configuração é possível visualizar que foram criadas 3 partições (sdb, sdc e sdd) de 100GB cada, além da partição (sda) utilizada para instalação do OpenFiler. Essas partições foram necessárias pois cada *node* replicará suas máquinas virtuais em uma partição.

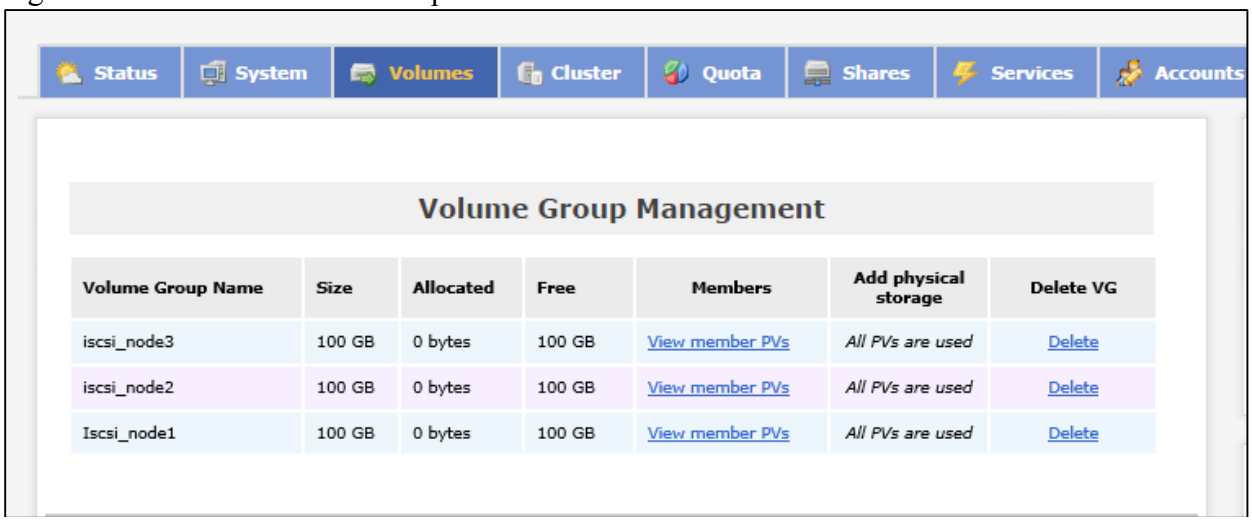
Figura 15 - Gerenciamento de disco no OpenFiler.



Fonte: Elaborada pelo autor.

A Figura 16 mostra os 3 volumes criados para cada *node* (iscsi_*node*1, iscsi_*node*2, iscsi_*node*3). Esses volumes foram criados pois a partir deles o protocolo iSCSI irá montar as máquinas virtuais no momento que são iniciadas nos *nodes*.

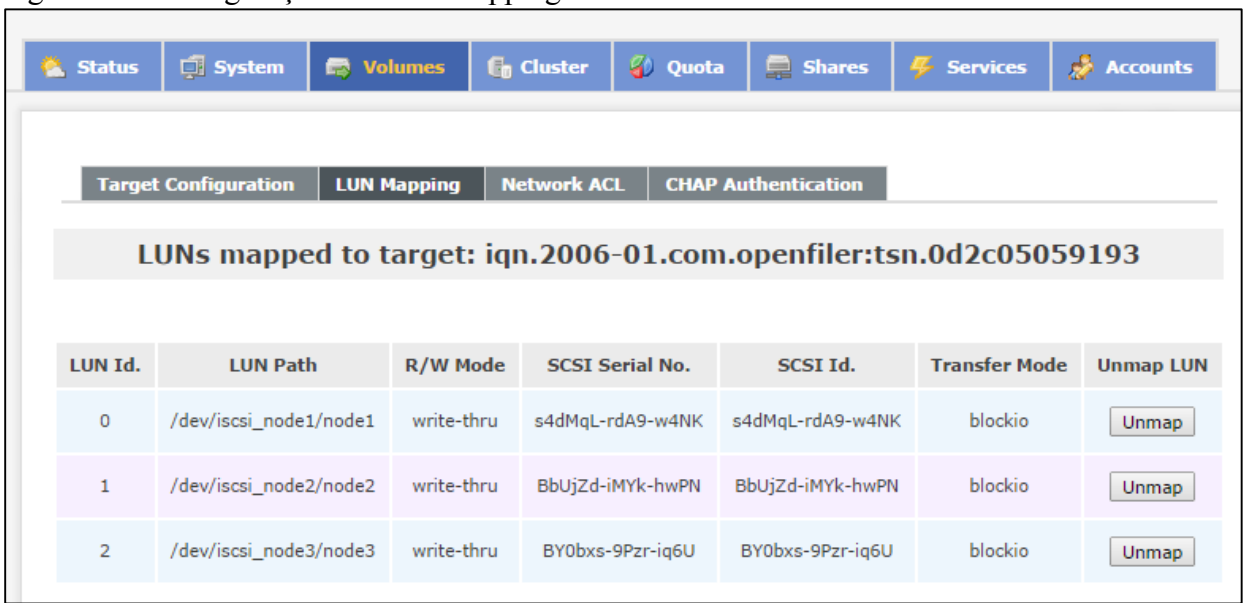
Figura 16 - Volume iSCSI criado para cada *node*.



Fonte: Elaborada pelo autor.

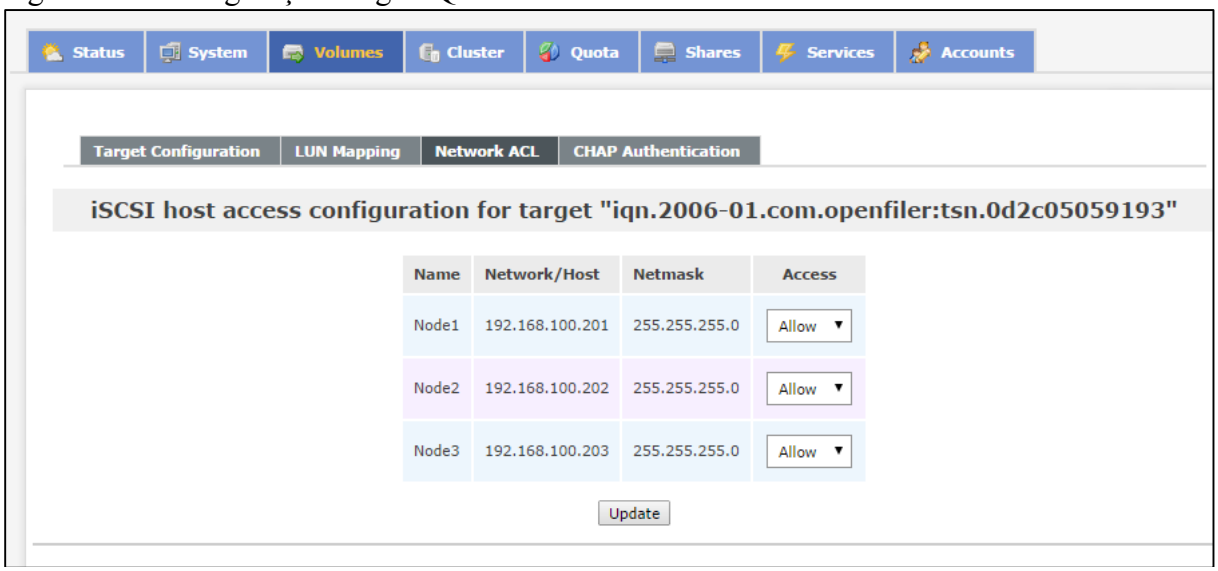
As Figuras 17 e 18, mostram as configurações do *LUN Mapping* e *Target IQN*. O *LUN Mapping* é necessário pois será o caminho lógico que o *node* irá enxergar na rede. O *Target IQN* funciona como um identificador de recursos entre o *node* e o OpenFiler.

Figura 17 - Configuração de LUN Mapping.



Fonte: Elaborada pelo autor.

Figura 18 - Configuração Target IQN.



Fonte: Elaborada pelo autor.

Vale ressaltar que o ambiente de teste do OpenStackTF tem limitações de *hardware*, mas o mesmo ambiente pode ser reproduzido em ambientes com mais recursos de *hardware*.

4.6 Testes e Resultados

Foram realizados testes com base no Tanenbaum e Steen (2007) que descrevem um ambiente tolerante a falhas, considerando que um ambiente tolerante a falha deve implementar disponibilidade, confiabilidade, segurança e capacidade de manutenção.

Os testes realizados foram intensificados nos *nodes*, pois como apresentado anteriormente, um usuário cria uma máquina virtual que será alocada em um *node* e caso o *node* apresente algum problema, a máquina virtual que está ligada é perdida e deve ser descartada pelo administrador da nuvem. Essa falha faz com que o usuário perca a sua máquina virtual, tendo que criar uma nova.

Para simular uma falha no *node*, foram realizados três tipos de testes. Um teste, retirando o cabo de rede do *node*, um teste reiniciando o node e o outro desligando o *node*. Esses testes de falhas são classificados como transiente ou intermitente. Para os dois tipos de testes realizados, foi utilizado a ferramenta *ping*[26] para testar se realmente o *node* não respondia e se estava apresentando a falha esperada. Além da utilização da ferramenta *ping* ter

sido utilizada em dois computadores o autor da pesquisa certificou-se pessoalmente que a máquina estava fora da rede ou desligada.

Os testes de simulação das falhas foram cronometrados para verificar quanto tempo uma máquina virtual volta a ficar ativa para o usuário. Para realizar esse controle de tempo, foi utilizado o próprio relógio do sistema operacional do *controller*. O controle do tempo foi realizado da seguinte forma:

- **Teste 1**: Para caso de falha na rede, o cabo de rede foi retirado e esperou-se a confirmação pelo *ping* que a máquina não respondia. Em seguida foi colocado o cabo de rede e iniciado o cronômetro. Quando o status da máquina virtual ficou "Active", o cronômetro foi cessado.
- **Teste 2**: Para o caso da máquina reiniciar, foi utilizado o comando "*shutdown -r now"*. No momento que o comando foi executado, iniciou-se o cronômetro e este foi cessado no momento que o status da máquina virtual ficou "Active".
- **Teste 3**: Para o caso da máquina desligar abruptamente, foi retirado o cabo de energia e recolocado logo em seguida. No momento que foi recolocado, iniciou-se o cronômetro e este foi cessado no momento que o status da máquina virtual ficou "Active".

Para assegurar a eficiência e a equidade na coleta de dados, foram feitos 3 testes e realizada uma média dos tempos.

Nas seções seguintes são apresentados dois cenários de testes. No primeiro cenário foram realizados os testes das falhas no ambiente inicial; para em seguida realizar o mesmo teste com as configurações ativas. O teste foi realizado somente com uma máquina virtual, para verificar a eficiência da solução. Já no segundo cenário, foram realizados testes com o maior número de máquinas possíveis. O segundo cenário foi realizado para verificar se a solução seria eficaz como no primeiro cenário.

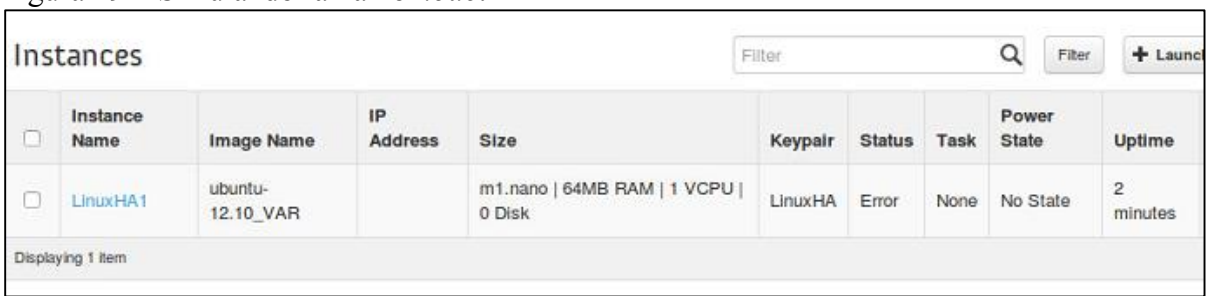
[26] Ping é um utilitário para testar a conectividade entre equipamentos. (CISCO, 2014).

4.7 Primeiro Cenário de Teste

Para essa etapa foi levado em consideração somente se a máquina virtual não é perdida após um *node* se recuperar de uma falha.

A Figura 19 mostra que a máquina virtual está com *status* de Erro. Para que o *status* de Erro aparecesse, foi simulado uma falha no *node*. Essa simulação foi realizada de três formas: retirando fisicamente o cabo de rede, reiniciando o *node* e desligando-o. Após o node se recuperar da falha a máquina virtual não pôde ser reaproveitada e foi necessário apagar a instância.

Figura 19 – Simulando falha no *node*.

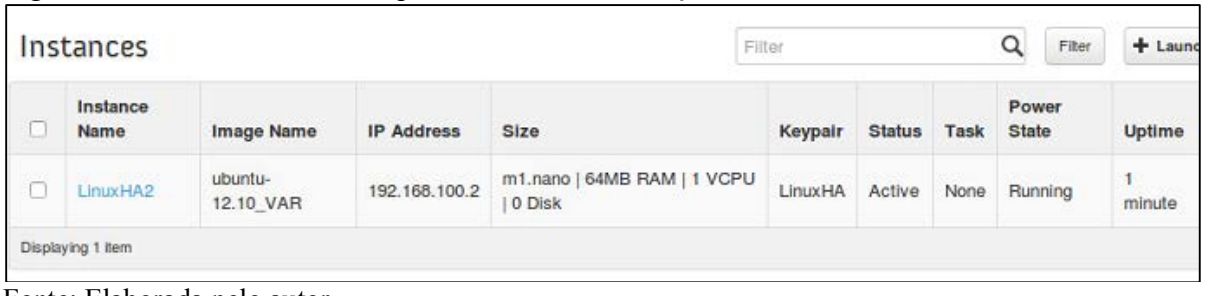


Fonte: Elaborada pelo autor.

Em seguida foi realizada a configuração de tolerância a falhas utilizando o OpenFiler, conforme descrito anteriormente e de acordo com o Apêndice D.

De acordo com a Figura 20, foi iniciada uma nova máquina virtual, só que dessa vez com o OpenStackTF ativo.

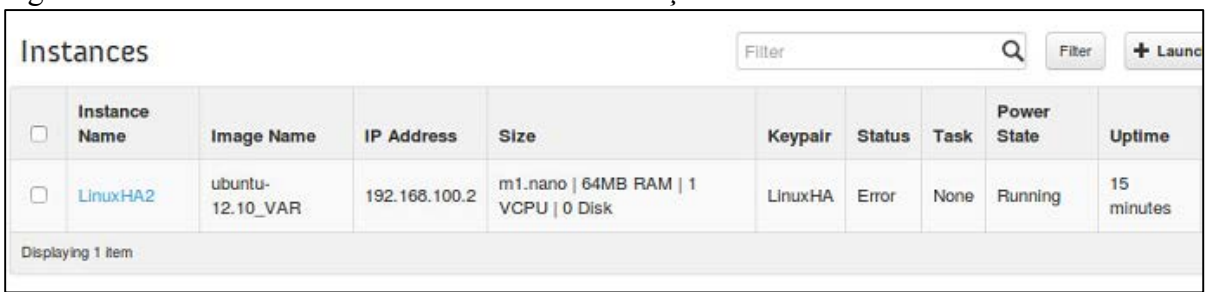
Figura 20 – Iniciando uma máquina virtual com solução de tolerância a falha.



Fonte: Elaborada pelo autor.

Conforme a Figura 21 o status da máquina apresentou Erro, pois foi retirado o cabo de rede e reiniciado e desligado o *node*.

Figura 21 – Simulando uma falha no *node* com solução de tolerância a falha.



Fonte: Elaborada pelo autor.

Após recolocar o cabo de rede ou o *node* ter reiniciado, o status da máquina voltou a ficar Ativa. Nesse caso não houve a necessidade de terminar a instância conforme mostra a Figura 22.

Figura 22 – Máquina virtual ativa após falha no *node*.



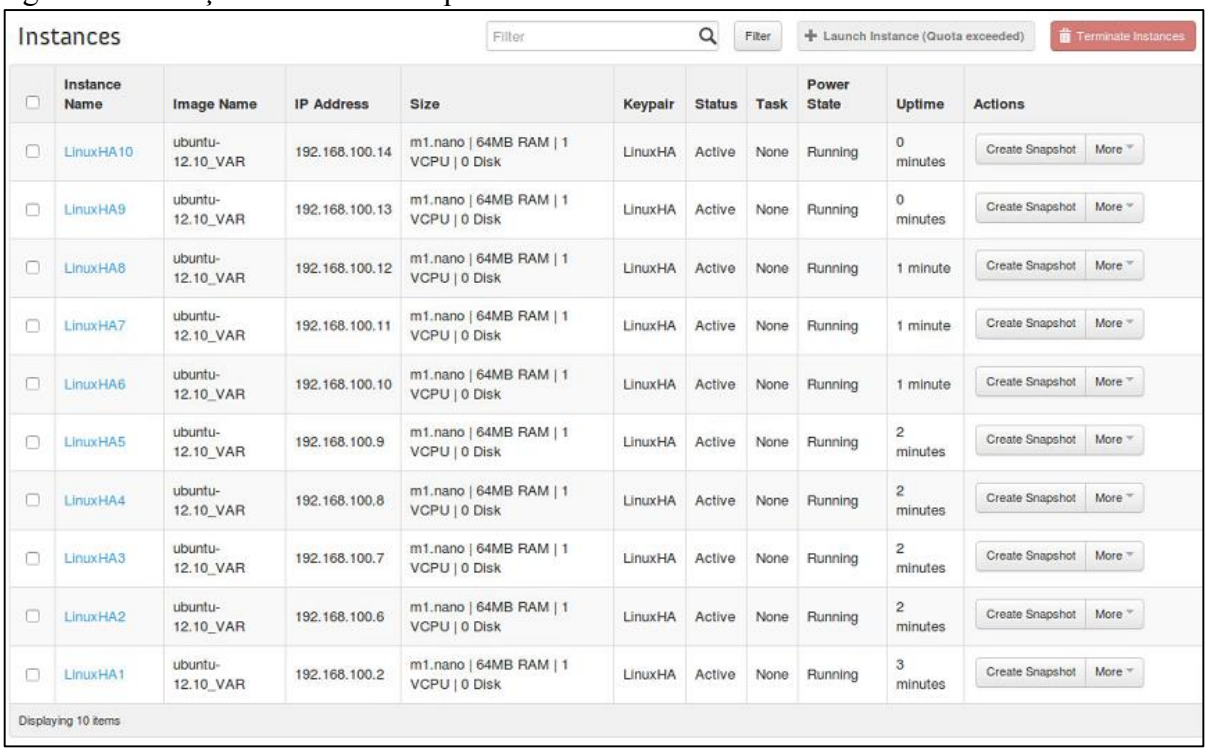
Fonte: Elaborada pelo autor.

Em todos os testes realizados, o comportamento observado foi o mesmo, a instancia voltou a ficar ativa após o *node* se recuperar da falha.

4.8 Segundo Cenário de Teste

O segundo cenário de testes foi realizado de forma similar ao primeiro, mas executando um número máximo de máquinas virtuais disponíveis para o ambiente de pesquisa. A Figura 23 apresenta o lançamento de 10 máquinas virtuais.

Figura 23 – Lançamento de 10 máquinas virtuais.



Instances

Filter | Launch Instance (Quota exceeded) | Terminate Instances

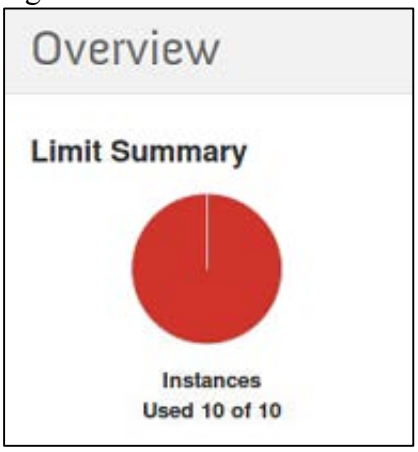
Instance Name	Image Name	IP Address	Size	Keypair	Status	Task	Power State	Uptime	Actions
LinuxHA10	ubuntu-12.10_VAR	192.168.100.14	m1.nano | 64MB RAM | 1 VCPU | 0 Disk	LinuxHA	Active	None	Running	0 minutes	Create Snapshot | More
LinuxHA9	ubuntu-12.10_VAR	192.168.100.13	m1.nano | 64MB RAM | 1 VCPU | 0 Disk	LinuxHA	Active	None	Running	0 minutes	Create Snapshot | More
LinuxHA8	ubuntu-12.10_VAR	192.168.100.12	m1.nano | 64MB RAM | 1 VCPU | 0 Disk	LinuxHA	Active	None	Running	1 minute	Create Snapshot | More
LinuxHA7	ubuntu-12.10_VAR	192.168.100.11	m1.nano | 64MB RAM | 1 VCPU | 0 Disk	LinuxHA	Active	None	Running	1 minute	Create Snapshot | More
LinuxHA6	ubuntu-12.10_VAR	192.168.100.10	m1.nano | 64MB RAM | 1 VCPU | 0 Disk	LinuxHA	Active	None	Running	1 minute	Create Snapshot | More
LinuxHA5	ubuntu-12.10_VAR	192.168.100.9	m1.nano | 64MB RAM | 1 VCPU | 0 Disk	LinuxHA	Active	None	Running	2 minutes	Create Snapshot | More
LinuxHA4	ubuntu-12.10_VAR	192.168.100.8	m1.nano | 64MB RAM | 1 VCPU | 0 Disk	LinuxHA	Active	None	Running	2 minutes	Create Snapshot | More
LinuxHA3	ubuntu-12.10_VAR	192.168.100.7	m1.nano | 64MB RAM | 1 VCPU | 0 Disk	LinuxHA	Active	None	Running	2 minutes	Create Snapshot | More
LinuxHA2	ubuntu-12.10_VAR	192.168.100.6	m1.nano | 64MB RAM | 1 VCPU | 0 Disk	LinuxHA	Active	None	Running	2 minutes	Create Snapshot | More
LinuxHA1	ubuntu-12.10_VAR	192.168.100.2	m1.nano | 64MB RAM | 1 VCPU | 0 Disk	LinuxHA	Active	None	Running	3 minutes	Create Snapshot | More

Displaying 10 items

Fonte: Elaborada pelo autor.

A Figura 24 mostra um gráfico disponível na interface, identificando que não foi possível lançar mais máquinas virtuais. A limitação de 10 máquinas virtuais é devido ao *hardware* disponível para implantação do OpenStackTF.

Figura 24 – Limite máximo de instancias atingidas.



Fonte: Elaborada pelo autor.

Após ativar as 10 máquinas virtuais no ambiente, foram realizados os mesmos procedimentos do primeiro cenário. Todas as máquinas virtuais obtiveram resultado de sucesso após o *node* retornar da falha.

4.9 Avaliação dos Resultados

Após a implantação e testes realizados no OpenStackTF, pôde-se avaliar que a solução encontrada resolveu um ponto falho no OpenStack: a reativação das máquinas virtuais instanciadas nos *nodes*. Desta forma os usuários não serão mais prejudicados caso uma falha de *hardware* ocorra no *node* em que a sua máquina virtual estava alocada. Já os administradores da nuvem que implementarem o OpenStackTF também serão beneficiados pois não precisarão recriar novas máquinas virtuais para os usuários caso a falha seja nos *nodes*.

Conforme resultados, os requisitos de confiabilidade apresentados por Tanenbaum e Steen (2007) sendo eles: disponibilidade, confiabilidade, segurança e capacidade de manutenção.

- O requisito de disponibilidade foi atingido; pois, após a falha no *node,* a máquina virtual ficou disponível para o usuário.
- O requisito de confiabilidade foi atingido; pois, o *node* apresentou uma falha, porém a máquina virtual se recuperou e ficou ativa, dando ao usuário a confiabilidade de não perder a máquina virtual.
- O requisito de segurança foi atingido; pois, o *node* ficou inoperante por algum tempo e mesmo assim voltou a funcionar normalmente. Além disso as máquinas virtuais ativas no momento da falha não foram danificadas.
- Já o requisito de capacidade de manutenção foi atingido; pois, o *node* que apresentou uma falha foi recuperado.

A Tabela 1 apresenta os resultados obtidos com os testes realizados no OpenStackTF. Estes resultados mostram o tempo em segundos que cada cenário levou para ativar as máquinas virtuais após os três testes de falhas. Por meio desta tabela é possível verificar também que quanto mais máquinas virtuais precisarem ser reativadas, maior será o tempo que o ambiente levará para ficar totalmente ativo.

Tabela 1 – Comparativos de tempos.

	Cenário 1	**Cenário 2**
Teste 1	1 segundo	3 segundos
Teste 2	35 segundos	90 segundos
Teste 3	55 segundos	115 segundos

Fonte: Elaborada pelo autor.

4.10 Considerações finais

Este capítulo apresentou inicialmente os trabalhos relacionados, assim como os materiais e métodos utilizados para o desenvolvimento do mecanismo OpenStackTF.

Foram apresentados dois cenários de testes no mecanismo implementado. Em ambos o mecanismo proposto é viável, pois oferece ao usuário uma alta disponibilidade em suas máquinas virtuais, criando uma transparência de tolerância a falhas. Este recurso permite ao administrador da nuvem evitar preocupações desnecessárias com os nodes, proporcionando maior disponibilidade de tempo para resolução de intercorrências comuns a esse ambiente.

5 – Conclusões e Trabalhos Futuros

Neste capítulo, são apresentadas as contribuições e as conclusões, assim como as limitações encontradas durante o desenvolvimento do mecanismo, e sugestões de trabalhos futuros para continuidade desta pesquisa.

5.1 Conclusões e Contribuições

As dificuldades encontradas por um usuário para assegurar a alta disponibilidade dos serviços de tecnologia principalmente em ambientes *open source*, foi um fator preponderante para o desenvolvimento do mecanismo OpenStackTF.

Pesquisas realizadas durante o desenvolvimento mostraram a inexistência de soluções gratuitas no OpenStack.

A proposta deste trabalho foi a implementação do mecanismo de tolerância a falha no OpenStack. Os resultados obtidos apontaram êxito na recuperação das máquinas virtuais instanciadas nos *nodes* com falhas.

Esta pesquisa foi relevante aos usuários e administradores de nuvem que pretendem implementar uma nuvem privada *open source*. Para que o administrador seja capaz de reproduzir o mecanismo proposto é necessário, primeiramente, possuir o OpenStack e posteriormente implementar o OpenStakTF.

Embora o mecanismo de tolerância a falha tenha alcançado o objetivo proposto, a ferramenta não esgota todos os assuntos relacionados por se tratar de um protótipo.

Esse mecanismo será disponibilizado por meio de fóruns de discussão da comunidade OpenStack com o intuído de alavancar as pesquisas nesta área.

5.2 Limitações

Durante o desenvolvimento do mecanismo OpenStackTF, a limitação encontrada, foi a quantidade de *hardware* disponível para criação dos testes. No período de testes, foi possível criar somente 10 máquinas virtuais nos 3 *nodes*, porém essa solução apresenta nível de significância em ambientes mais robustos.

5.3 Trabalhos Futuros

Com o desenvolvimento desse trabalho, algumas opções de continuidade de pesquisa surgiram e são apresentados nessa seção.

Uma possibilidade de trabalho futuro na arquitetura proposta refere-se a migração da máquina virtual do *node* com falha para outro *node* disponível; devendo ser avaliado o tempo que uma máquina virtual levará para migrar de um *node* a outro.

Outra melhoria é a possibilidade das máquinas virtuais não ficarem inoperantes durante uma falha no *node*. Neste caso a máquina virtual poderia ser instanciada simultaneamente em dois *nodes*. Nesta situação, o tempo de uma máquina virtual subir em outro *node* é zero, pois já estará ativa em outro *node*.

Outro ponto, é a implementação de uma redundância no *controller* da nuvem, pois se o *controller* apresentar algum tipo de falha irreversível, a nuvem ficará fora de operação até que seja corrigido a falha do *controller*.

REFERÊNCIAS

AMAZON, **Centro de Arquitetura da AWS**, Acesso em: <http://aws.amazon.com/pt/architecture/>, 15 de setembro de 2014.

ANDERSON, T.; LEE, P. A. **Fault tolerance -principles and practice**. Englewood Cliffs, Prentice-Hall, 1981.

BACHIEGA, N. G. ; MARTINS, H. P. ; SPOLON, R. ; CAVENAGHI, M. A. ; LOBATO, R. S. ; MANACERO Jr., A.. **Open Source *Cloud Computing*: Characteristics and an Overview**. Em: The 19th International Conference on Parallel and Distributed Processing Techniques and Applications (PDPTA 2013), p. 243-248, 2013.

BHADAURIA, R.; CHAKI, R.. **A Survey on Security Issues in Cloud Computing**. The paper is communicated to the IEEE Communications Surveys and Tutorials. 2011.

BREITMAN, K.; VITERBO, J.. **Congresso Internacional *Software* Livre e Governo Eletrônico** (3. : 2010 : Brasília) Amãpytuna: computação em nuvem: serviços livres para a sociedade do conhecimento. Brasília : FUNAG, 2010.

BUYYA, R.; RANJAN, R.; CALHEIROS, R. N. **Intercloud: Utility-oriented federation of cloud computing environments for scaling of application services**. Proceedings of the 10th International Conference on Algorithms and Architectures for Parallel Processing (ICA3PP 2010, Busan, South Korea, May 21-23), LNCS, Springer, Germany, 2010., v. abs/1003.3920, 2010.

CEARLEY, D. et al – **Hype Cycle for Application Development** – Gartner Group report number G00147982 – Relatório técnico do grupo gartner. Acesso em: <http://www.gartner.com/>, 1 de fevereiro de 2012.

CHAUHAN, V. K.; BANSAL, K.; ALAPPANAVAR, P. **Exposing cloud computing as a failure**. International journal of engineering science and technology [0975-5462] Vikas Ano:2012 Vol:4 Nr:4

CISCO, **Understanding the Ping and Traceroute Commands**, Acesso em: <http://www.cisco.com/c/en/us/support/docs/ios-nx-os-software/ios-software-releases-121-mainline/12778-ping-traceroute.html#ping_com>, 15 de setembro de 2014.

CITRIX, Acesso em: <http://www.citrix.com.br/>, 15 de setembro de 2014.

CRISTIAN, F. **Understanding Fault-Tolerant Distributed Systems**. Commun. ACM, (34)2:56-78, fevereiro 1991.

DANTAS, M. **Computação Distribuída de Alto Desempenho, Redes, Clusters e Grids Computacionais**. Axcel Books do Brasil Editora, 2005.

FOSTER, I.; YONG Z.; RAICU, I.; SHIYONG L., **Cloud Computing and Grid Computing 360-Degree Compared**, Grid Computing Environments Workshop, 2008. GCE '08 , vol., no., pp.1,10, 12-16 Nov. 2008

FURHT, B. **Cloud computing fundamentals**. In: FURHT, B.; ESCALANTE, A., eds. Handbook of Cloud Computing, cap. 1, Boston, MA: Springer US, p. 3–19, 2010.

GIL, A. C. **Como Elaborar Projetos de Pesquisa**. 4ª ed. São Paulo: Atlas, 2002. 176 p.

GILDER, G. **The Information Factories**, Wired Magazine, Outubro, 2006.

HADZILACOS, V. TOUEG, S. **Fault-Tolerant Broadcast and Related Problems**. In Mullender, S. (ed.), Distributed Systems, Wokingham: Addison-Wesley, 2. Ed., pp. 97-145. 1993.

HASS, F. ***OpenStack* High Availability Guide**, Acesso em: <http://docs.*OpenStack*.org/trunk/*OpenStack*-ha/*OpenStack*-ha-guide-trunk.pdf>, 20 de Janeiro de 2013.

IBM. **Computação em nuvem**, Acesso em: <http://www.ibm.com/developerworks/br/cloud/index.html>, 20 de janeiro de 2013.

JHAWAR, R.; PIURI, V.; SANTAMBROGIO, M. **A comprehensive conceptual system-level approach to fault tolerance in Cloud Computing**, Systems Conference (SysCon), 2012 IEEE International, vol., no., pp.1-5, 19-22 March 2012.

KIRKLAND, Dustin. **UEC/PackageInstall**. Acesso em: <https://help.ubuntu.com/community/UEC/PackageInstall>. 20 de janeiro de 2013.

LABS., C12g. **Planning the Installation 3.0**. Acesso em: <http://opennebula.org/documentation:archives:rel3.0:plan>. 20 de janeiro de 2013.

LAKATOS, E. M.; MARCONI, M. A. **Metodologia do Trabalho Científico**. 3ª ed. São Paulo: Atlas, 1992. 214 p.

LIU, S., LIANG, Y., and BROOKS, M. **Eucalyptus: a web service-enablede infrastructure**. In CASCON ’07: Proceedings of the 2007 conference of the center for advanced studies on Collaborative research, pages 1–11, New York, NY, USA. ACM, 2007.

MACAPUNA, C. A. B.; ROTHENBERG C. E.; MAGALHÃES, M. F. **Controle distribuído em Redes de Data Center baseado em Gerenciadores de Rack e Serviços de Diretório/Topologia**. Anais / VIII Workshop em Clouds, Grids e Aplicações; Porto Alegre: SBC. 157 p. 2010.

MAHJOUB, M.; MDHAFFAR, A.; BEN HALIMA, R.; JMAIEL, M. **A comparative study of the current Cloud Computing technologies and offers**. First International Symposium on Network Cloud Computing and Applications, 2011.

MICROSOFT. **O que é Internet Small Computer System Interface (iSCSI)?** Acesso em: <http://windows.microsoft.com/pt-br/windows-vista/what-is-internet-small-computer-system-interface-iscsi>, 15 de junho de 2014.

MILLER, M. C**loud Computing: Web-Based Applications That Change the Way You Work and Collaborate Online**. Que Publishing Digital, 2008.

NEMETH, Evi; SNYDER, Garth; HEIN, Trent R.. **Manual completo do linux: guia do administrador**. 2 ed. São Paulo: Peaarson Prentice Hall, 2007.

NIST. **The NIST Definition of Cloud Computing**, NIST, 2011.

OPENSTACK. **What is *OpenStack*?** Acesso em: <http://docs.*OpenStack*.org/cactus/*OpenStack*-compute/admin/content/what-is-*OpenStack*.html>, 15 de setembro de 2014.

ROBERTS, John C.; AL-HAMDANI, **Wasim.Who can you trust in the cloud?: a review of security issues within cloud computing**. In Proceedings of the 2011 Information Security Curriculum Development Conference (InfoSecCD '11).ACM, New York, NY, USA, 15-19, 2011.

SABAHI, F. **Cloud computing security threats and responses.Communication *Software* and Networks (ICCSN)**, 2011 IEEE 3rd International Conference on May 2011.

SIWCZAK P. **Understanding your options: Deployment topologies for High Availability (HA) with *OpenStack*,** Acesso em: <http://www.mirantis.com/blog/117072/>, 10 de fevereiro de 2013.

SQLITE.ORG. **About SQLite**. Acesso em: <http://www.sqlite.org/about.html>, 20 de janeiro de 2013.

SUMTER, La'Quata. **Cloud computing: security risk**. In Proceedings of the 48th Annual Southeast Regional Conference (ACM SE '10).ACM, New York, NY, USA, Article 112 , 4 pages. 2010.

TANENBAUM, A.S.; STEEN, M. V. **Sistemas distribuídos: princípios e paradigmas**. 2 ed, São Paulo: Pearson Prentice Hall, 2007

TCHANA, A.; BROTO, L.; HAGIMONT, D. **Approaches to cloud computing fault tolerance**, Computer, Information and Telecommunication Systems (CITS), 2012 International Conference on, vol., no., pp.1-6, 14-16 May 2012

UNDHEIM, A.; CHILWAN, A.; HEEGAARD, P. **Differentiated Availability in Cloud Computing SLAs**. GRID '11 Proceedings of the 2011 IEEE/ACM 12th International Conference on Grid Computing Pages 129-136. 2011

VERAS, M. **Cloud computing: nova arquitetura da TI**. Rio de Janeiro: Brasport, 2012.

VMWARE, **Disponibilidade**, Acesso em: <http://www.vmware.com/>, 15 de Setembro de 2014.

WEBER, T.S., **Um roteiro para exploração dos conceitos básicos de tolerância a falhas**, Acesso em: <http://www.inf.ufrgs.br/~taisy/disciplinas/textos/Dependabilidade.pdf>, 20 de janeiro de 2013.

WU, L.; CHEN, Z.; SU, J.; LUO, X. **The based-role pmi model for access control in large scale netware system**. In: Computer Design and Applications (ICCDA), 2010 International Conference on, 2010, p. V2–81–V2–84.

ZHANG, Q.; CHENG, L.; BOUTABA, R. **Cloud computing: state-of-the-art and research challenges**. Journal of Internet Services and Applications, v. 1, n. 1, p. 7–18, 2010.

Apêndice A - Instalação do *Eucalyptus*

A instalação do *Eucalyptus* divide-se basicamente em duas partes, o *cloud controller* (gestor) e os *nodes* (nós). Após instalação do sistema operacional (Ubuntu 10.10 64bits *Eucalyptus Cloud*) deve-se atualizar o sistema nas duas máquinas:

```
sudo apt-get update
sudo apt-get dist-upgrade
```

Instalação *Controller*

```
sudo apt-get install eucalyptus-cloud eucalyptus-cc eucalyptus-walrus eucalyptus-sc
```

Nome do Cluster: cluster1
Range de IPs disponíveis para Rede Pública: 192.168.1.200-192.168.1.249

Instalação *Node*

```
sudo apt-get install eucalyptus-nc
```

A comunicação entre o *controller* e os *nodes* é feita pela interface de rede "br0":

```
auto br0
iface br0 inet static
        address 192.168.70.1
        network 192.168.70.0
        netmask 255.255.255.0
        broadcast 192.168.70.255
        bridge_ports eth0
        bridge_fd 9
        bridge_hello 2
        bridge_maxage 12
        bridge_stp off
```

Após, configurar o eucalyptus.conf com a interface de rede "br0":

```
sudo /etc/init.d/eucalyptus-nc restart
```

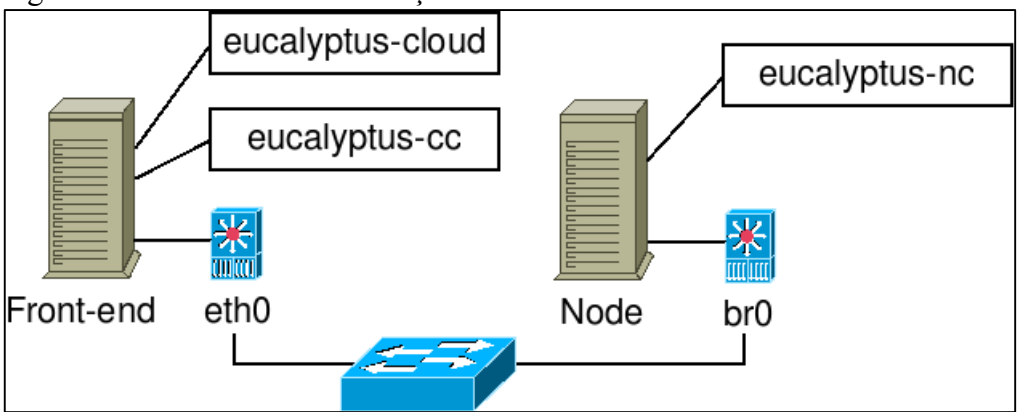
Definir uma senha para o usuário "eucalyptus" no *node controller:*

```
sudo passwd eucalyptus
```

No cloud *controller*, autorizar o usuário "eucalyptus" a acessar os *nodes* diretamente:

```
sudo -u eucalyptus ssh-copy-id -i ~eucalyptus/.ssh/id_rsa.pub eucalyptus@<IP_OF_NODE>
```

Figura 25 – Ambiente de Instalação.



Fonte: Kirkland (2012).

Em seguida, no *controller*, é necessário criar as credenciais de usuário para utilizar as instâncias de VM:

```
unzip -d ~/.euca mycreds.zip
mkdir -p ~/.euca
chmod 700 ~/.euca
cd ~/.euca
sudo euca_conf --get-credentials mycreds.zip
unzip mycreds.zip
ln -s ~/.euca/eucarc ~/.eucarc
cd -
```

Para validar seu cluster e verificar se a agregação de instâncias está funcionando:

```
. ~/.euca/eucarc
euca-describe-availability-zones verbose
```

Carregando uma instância de VM

No *cloud controller*, realizar as operações abaixo para carregar em uma instância a imagem do euca-ubuntu-9.04-x86_64.tar.gz:

```
tar zxvf euca-ubuntu-9.04-x86_64.tar.gz
euca-bundle-image -i euca-ubuntu-9.04-x86_64/kvm-kernel/vmlinuz-2.6.28-11-generic --kernel true
euca-upload-bundle -b ubuntu-kernel-bucket -m /tmp/vmlinuz-2.6.28-11-generic.manifest.xml
euca-register ubuntu-kernel-bucket/vmlinuz-2.6.28-11-generic.manifest.xml

euca-bundle-image -i euca-ubuntu-9.04-x86_64/kvm-kernel/initrd.img-2.6.28-11-generic --ramdisk true
euca-upload-bundle -b ubuntu-ramdisk-bucket -m /tmp/initrd.img-2.6.28-11-generic.manifest.xml
euca-register ubuntu-ramdisk-bucket/initrd.img-2.6.28-11-generic.manifest.xml

euca-bundle-image -i euca-ubuntu-9.04-x86_64/ubuntu.9-04.x86-64.img --kernel $EKI --ramdisk $ERI
euca-upload-bundle -b ubuntu-image-bucket -m /tmp/ubuntu.9-04.x86-64.img.manifest.xml
euca-register ubuntu-image-bucket/ubuntu.9-04.x86-64.img.manifest.xml
```

Para executar a instância:

```
euca-run-instances -k mykey --kernel <eki-XXXXXXXX> --ramdisk <eri-XXXXXXXX> <emi-XXXXXXXX>
```

Para acompanhar sua inicialização:

```
watch -n5 euca-describe-instances
```

Apêndice B - Instalação do *OpenNebula*

A instalação do *OpenNebula* não difere da instalação de outros gestores da nuvem, divide-se basicamente em duas partes, o *cloud controller* (gestor) e os *nodes* (nós). Após instalação do sistema operacional (Ubuntu 10.10 64bits Server) deve-se atualizar o sistema nas duas máquinas:

```
sudo apt-get update
sudo apt-get dist-upgrade
```

Instalação *Controller*

```
sudo apt-get install opennebula
```

Nome do Cluster: opennebula

Instalação *Node*

```
sudo apt-get install opennebula-node
```

A comunicação entre o *controller* e os *nodes* é feita pela interface de rede "br0":

```
auto br0
iface br0 inet static
        address 192.168.70.1
        network 192.168.70.0
        netmask 255.255.255.0
        broadcast 192.168.70.255
        bridge_ports eth0
        bridge_fd 9
        bridge_hello 2
        bridge_maxage 12
        bridge_stp off
```

Após, é necessário crias o sistema de chave pública para permitir comunicação direta entre nós e gestor:

```
sudo scp /var/lib/one/.ssh/id_rsa.pub oneadmin@node01:/var/lib/one/.ssh/authorized_keys
sudo scp /var/lib/one/.ssh/id_rsa.pub oneadmin@node02:/var/lib/one/.ssh/authorized_keys
```

```
sudo sh -c "cat /var/lib/one/.ssh/id_rsa.pub >> /var/lib/one/.ssh/authorized_keys"
```

É necessário criar o diretório que irá alocar as imagens de instâncias de VMs:

```
sudo mkdir /var/lib/one/images
sudo chown oneadmin /var/lib/one/images/
```

Em seguida, devem-se registrar os nós participantes do cluster:

```
onehost create node01 im_kvm vmm_kvm tm_ssh
onehost create node02 im_kvm vmm_kvm tm_ssh
```

Para que a instância de VM possa comunicar na rede, é necessário criar uma configuração de rede virtual para o cluster (vnet01.template):

```
NAME = "LAN"
TYPE = RANGED
BRIDGE = br0
NETWORK_SIZE = C
NETWORK_ADDRESS = 192.168.0.0
```

É necessário carregar o *template* de rede criado:

```
onevnet create vnet01.template
```

Após, é criado o *template* da instância de VM, informando todos os parâmetros necessários para a inicialização da máquina virtual (vm01.template):

```
NAME = vm01
CPU = 0.5
MEMORY = 512
OS = [ BOOT   = hd ]
DISK = [
  source   = "/var/lib/one/images/vm01.qcow2",
  target   = "hda",
  readonly = "no" ]
NIC = [ NETWORK="LAN" ]
GRAPHICS = [type="vnc",listen="127.0.0.1",port="-1"]
```

Finalmente, deve-se criar a instância de VM:

```
onevm submit vm01.template
```

Apêndice C - Instalação do *OpenStack*

Na instalação do *OpenStack* serão utilizados cinco computadores, cada um com uma função específica e todos com o sistema operacional (Ubuntu 12.04.2 i386 Server) devidamente atualizados:

```
sudo apt-get update
sudo apt-get upgrade
sudo apt-get install -y unzip
sudo apt-get install -y bridge-utils
brctl addbr br100
apt-get install iputils-arping -y
apt-get install arping -y
apt-get install git -y
```

Servidor Gestor

O gestor contém todos os serviços da nuvem, nova-compute, nova-api, nova-volume, nova-network, Glance Image Service e Swift. Configuração da rede, considerando:

- eth0: 192.168.0.0/24 (comunicação gestor-*node*)
- eth1: 10.10.100.0/24 (comunicação externa, acesso ssh e vnc)

```
auto lo
iface lo inet loopback
auto br100
iface br100 inet static
bridge_ports eth0
bridge_stp off
bridge_maxwait 0
bridge_fd 0
address 192.168.100.200
netmask 255.255.255.0
auto eth1
iface eth1 inet static
address 10.10.100.100.1
netmask 255.255.255.0
```

A instalação do servidor de NTP é importante para manter a sincronização de relógios entres os computadores participantes da nuvem.

```
#apt-get install -y ntp
#sed -i 's/server ntp.ubuntu.com/server ntp.ubuntu.com\nserver 127.127.1.0\nfudge 127.127.1.0 stratum
10/g' /etc/ntp.conf
```

Após, é necessário adicionar o usuário de comunicação Gestor/*Nodes*:

```
#useradd -U -G sudo -s /bin/bash -m stack
```

```
#echo "stack ALL=(ALL) NOPASSWD: ALL" >> /etc/sudoers
#mkdir ~/.ssh; chmod 700 ~/.ssh
#echo "ssh-rsa
AAAAB3NzaC1yc2EAAAADAQABAAABAQCyYjfgyPazTvGpd8OaAvtU2utL8W6gWC4JdRS1J95G
hNNfQd657yO6s1AH5KYQWktcE6FO/xNUC2reEXSGC7ezy+sGO1kj9Limv5vrvNHvF1+wts0Cmyx6
1D2nQw35/Qz8BvpdJANL7VwP/cFI/p3yhvx2lsnjFE3hN8xRB2LtLUopUSVdBwACOVUmH2G+2BW
MJDjVINd2DPqRIA4Zhy09KJ3O1Joabr0XpQL0yt/I9x8BVHdAx6l9U0tMg9dj5+tAjZvMAFfye3PJcY
wwsfJoFxC8w/SLtqlFX7Ehw++8RtvomvuipLdmWCy+T9hIkl+gHYE4cS3OIqXH7f49jdJf
jesse@spacey.local" > ~/.ssh/authorized_keys
```

Através do Git, é realizado o clone do script DevStack, responsável pela instalação automática dos pacotes necessários para a nuvem:

```
# mkdir /opt/stack/
#cd ~
#git clone git://github.com/OpenStack-dev/devstack.git
#cd devstack
```

Na sequência, adicionar o host no /etc/hosts:

```
192.168.100.200 controller
```

Configuração do localrc (em ~/devstack/):

```
HOST_IP=192.168.100.200
FLAT_NETWORK_BRIDGE=br100
FLAT_INTERFACE=eth1
FIXED_RANGE=10.10.100.192/26
FIXED_NETWORK_SIZE=62
PUBLIC_INTERFACE=eth0
FLOATING_RANGE=192.168.100.200/24
MULTI_HOST=1
LOGFILE=/opt/stack/logs/stack.sh.log
ADMIN_PASSWORD= cloudunesp
MYSQL_PASSWORD= cloudunesp
RABBIT_PASSWORD= cloudunesp
SERVICE_PASSWORD= cloudunesp
SERVICE_TOKEN= cloudunesp
```

Rodar o script ./stack.sh (em ~/devstack/):

```
#./stack.sh
```

Nodes

Para os *node*, é necessário repetir todos os passos do gestor, alterando o IP correspondente na interface de rede e o arquivo localrc (em ~/devstack/):

```
HOST_IP=192.168.100.200
FLAT_NETWORK_BRIDGE=br100
FLAT_INTERFACE=br100
VLAN_INTERFACE=br100
FIXED_RANGE=192.168.100.0/24
FIXED_NETWORK_SIZE=4096
FLOATING_RANGE=10.10.100.112/28
PUBLIC_INTERFACE=eth1
MULTI_HOST=1
```

```
LOGFILE=/opt/stack/logs/stack.sh.log
ADMIN_PASSWORD=cloudunesp
MYSQL_PASSWORD=cloudunesp
RABBIT_PASSWORD=cloudunesp
SERVICE_PASSWORD=cloudunesp
SERVICE_TOKEN=cloudunesp
MYSQL_HOST=192.168.100.200
RABBIT_HOST=192.168.100.200
GLANCE_HOSTPORT=192.168.100.200:9292
ENABLED_SERVICES=n-cpu,n-net
```

Rodar o script ./stack.sh (em ~/devstack/):

```
#./stack.sh
```

Para carregar os serviços no *node*:

```
#/opt/stack/nova/bin/nova-compute start &
#/opt/stack/nova/bin/nova-network start &
#nova-manage db sync
```

Acessar o gestor com root, executar o comando abaixo para checar a comunicação entre gestor e *node*:

```
#nova-manage service list
```

O comando acima deve retornar:

Binary	Host	Zone	Status	State	Updated_At
nova-cert	controller	nova	enabled	:-)	26/11/2012
nova-compute	controller	nova	enabled	:-)	26/11/2012
nova-scheduler	controller	nova	enabled	:-)	26/11/2012
nova-network	controller	nova	enabled	:-)	26/11/2012
nova-consoleauth	controller	nova	enabled	:-)	26/11/2012
nova-compute	node	nova	enabled	:-)	26/11/2012
nova-network	node	nova	enabled	:-)	26/11/2012

Como pode ser observado acima, o *node* apareceu em host com status ":-)". Para finalizar, acessar a interface gráfica Horizon e carregar as instâncias de máquinas virtuais.

Apêndice D - Instalação do *OpenFiler*

Na instalação do *OpenFiler* foi utilizado 1 computador como sendo o Alvo (Target) iSCSI e um iniciador (initiator) iSCSI, ambos rodando Ubuntu 12.04.2 i386 Server devidamente atualizados:

```
sudo apt-get update
sudo apt-get upgrade
apt-get install open-iscsi
```

Configuração do Initiator iSCSI

Abrir o arquivo de configuração open `/etc/iscsi/iscsid.conf...`

```
vi /etc/iscsi/iscsid.conf
```

... e configure *`node.startup`* to *`automatic`*:

```
[…]
node.startup = automatic
[…]
```

Reinicie o serviço do initiator:

```
/etc/init.d/open-iscsi restart
```

Em seguida conectar no target (`openfiler`):

```
iscsiadm -m discovery -t st -p 192.168.100.230
```

O resultado esperado deve ser:

```
root@openfiler:~# iscsiadm -m node
192.168.100.230:3260,1 iqn.2006-01.com.openfiler:tsn.cloud
root@openfiler:~#
```

Certifique-se de que o novo disco foi detectado usando dmesg:

```
dmesg | grep sd

[...]
 [ 2486.960577]  sdb: sdb1
[...]
```

Na saída acima sdb é o novo disco iSCSI. Em seguida, criar uma partição, formatar o sistema de arquivos e montar o novo disco iSCSI. Em um terminal digite:

```
sudo fdisk /dev/sdb
n
p
enter
w
```

Agora formatar o sistema de arquivos e montá-lo em / como um exemplo:

```
sudo mkfs.ext4 /dev/sdb1
sudo mount /dev/sdb1 /
```

Por fim, adicione uma entrada no `/etc/fstab` para montar a unidade iSCSI durante a inicialização:

```
/dev/sdb1       /       ext4    defaults,auto,_netdev 0 0
```