UNIVERSIDADE FEDERAL DA PARAÍBA

CENTRO DE CIÊNCIAS APLICADAS A EDUCAÇÃO

DEPARTAMENTO DE CIÊNCIAS EXATAS

BACHARELADO EM SISTEMAS DE INFORMAÇÃO

ANÁLISE DE VIABILIDADE DO USO DE NUVENS COMPUTACIONAIS OPORTUNISTAS

KAWE RAMON BORGYDARK DE OLIVEIRA

Orientador: Prof. MSc. MARCUS WILLIAMS AQUINO DE CARVALHO

RIO TINTO - PB

2013

KAWE RAMON BORGYDARK DE OLIVEIRA

ANÁLISE DE VIABILIDADE DO USO DE NUVENS COMPUTACIONAIS OPORTUNISTAS

Monografia apresentada para obtenção do título de Bacharel à banca examinadora no Curso de Bacharelado em Sistemas de Informação do Centro de Ciências Aplicadas e Educação (CCAE), Campus IV da Universidade Federal da Paraíba.
Orientador: Prof. MSc. MARCUS WILLIAMS AQUINO DE CARVALHO.

RIO TINTO - PB
2013

O48a Oliveira, Kawe Ramon Borgydark de.
Análise de viabilidade do uso de nuvens computacionais oportunista / Kawe Ramon Borgydark de Oliveira. – Rio Tinto: [s.n.], 2013.
73f.: il. –
Orientador: Marcus Williams Aquino de Carvalho.
Monografia (Graduação) – UFPB/CCAE.

1. Informática. 2. Computação em nuvem. 3. Computação Utilitária. 4. Computação em Grade. I. Título.

UFPB/BS-CCAE CDU:004(043.2)

KAWE RAMON BORGYDARK DE OLIVEIRA

ANÁLISE DE VIABILIDADE DO USO DE NUVENS COMPUTACIONAIS OPORTUNISTAS

Trabalho de Conclusão de Curso submetido ao Curso de Bacharelado em Sistemas de Informação da Universidade Federal da Paraíba, Campus IV, como parte dos requisitos necessários para obtenção do grau de BACHAREL EM SISTEMAS DE INFORMAÇÃO.

Assinatura do autor:_____

APROVADO POR:

_____

Orientador: Prof. Msc. MARCUS WILLIAMS AQUINO DE CARVALHO
Universidade Federal da Paraíba – Campus IV

_____

Prof. MSc. RODRIGO DE ALMEIDA VILAR DE MIRANDA
Universidade Federal da Paraíba – Campus IV

_____

Prof. MSc. YURI DE ALMEIDA MALHEIROS BARBOSA
Universidade Federal da Paraíba – Campus IV

RIO TINTO - PB
2013

Aos amigos, família, colegas e professores, minha eterna gratidão por compartilhar comigo seus conhecimentos.

AGRADECIMENTOS

Primeiramente quero agradecer a meu pai, pela grande ajuda durante todo o curso, além dos conselhos e a motivação pra seguir em frente. Quero agradecer também a meus familiares e amigos que me incentivaram ao longo dessa jornada. Agradeço ao meu orientador que me ajudou a compreender os assuntos necessários para construção desse trabalho.

RESUMO

A computação em nuvem é uma tecnologia que cada vez mais está ganhado mercado, usada por grandes e pequenas empresas, e que é capaz de prover acesso elástico e flexível a ambientes de redes e provedores de recursos computacionais, atendendo às necessidades de usuários que precisam de aplicações que executam em ambientes de grande escala. Também possibilita o melhor uso do desempenho dos recursos computacionais, evitando a ociosidade. Neste trabalho é proposta uma abordagem de nuvem oportunista que permite o uso de recursos ociosos de uma infraestrutura computacional disponível, possibilitando a criação de um ambiente de rede mais rico, capaz de fornecer muito mais recursos de *hardware* e *software* ao se comparar com um ambiente comum.

Palavras chave: Computação em nuvem, Software, Hardware, Eucalyptus, OpenStack, Computação Utilitária, Computação em Grade, SaaS, IaaS, PaaS.

ABSTRACT

Cloud computing is a technology which is increasingly gaining market. Used by large and small companies, it is able to provide elastic and flexible access to network environments and computational resources providers, meeting the users' necessities who needs applications which run on large-scale environments. It also enables the best use of the performance of computational resources, avoiding laziness. On this essay, it is proposed a cloud opportunistic approach, which allows the use of idle resources in an available computing infra structure, enabling the creation of a richer network environment, able to provide more resources for hardware and software compared to a common environment.

Keywords: Cloud Computing, Software, Hardware, Eucalyptus, OpenStack, Utility Computing, Grid Computing, SaaS, IaaS, PaaS.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE SIGLAS

TI	Tecnologia da Informação
CPU	*Central Processor Unit* (Unidade Central de Processamento)
NIST	*National Institute of Standards and Tenology* (Instituto Nacional de Padrões e Tecnologia)
SaaS	*Software as a Service* (Software como um Serviço)
IaaS	*Infrastructure as a Service* (Infraestrutura como um Serviço)
PaaS	*Plataform as a Service* (Plataforma como um Serviço)
SLA	*Service Level Agreement* (Nílve de Acordo de Serviço)
QoS	*Quality of Service (*Qualidade do Serviço*)*
VM	*Virtual Machine (*Máquina Virtual*)*
DNS	*Domain Name System (*Sistema de Domínio de Nomes*)*
VLAN	*Virtual Local Area Network (*Área de rede Virtual*)*
NTP	*Network Time Protocol (*Protocolo de Tempo de Rede*)*

SUMÁRIO

1 INTRODUÇÃO

Com a necessidade de flexibilidade e elasticidade em recursos computacionais, a computação em nuvem (*Cloud Computing*) possibilita a criação de um cenário típico de implantação de nuvens privadas e públicas. Essas e diversas vantagens que esse tipo de abordagem permite, superam o uso de métodos comuns usados antigamente como computação em grade. De acordo com GuiyiWei et al (2009), a computação em nuvem é uma evolução natural para dados e centros de computação com a gestão automatizada de sistemas, balanceamento de carga de trabalho, e as tecnologias de virtualização.

O uso de máquinas é sempre acompanhado com despesas de manutenção, suporte e instalação, pois computadores possuem componentes físicos e lógicos que precisam de atualização periodicamente. Com o objetivo de minimizar esse tipo de desperdício e também de capital com renovações de licenças de software, é proposto o uso de uma nuvem privada não suscetível a perda de informações e acessível em qualquer ponto de rede. Nesse trabalho será implantado o conceito de nuvem oportunista que visa o melhor uso dos recursos computacionais acessíveis. Esse tipo de abordagem, usa apenas uma pequena parte do processamento do computador quando o usuário estiver acessando-o, não comprometendo as atividades do mesmo.

Um exemplo de ambiente que pode usufruir dos benefícios da nuvem computacional oportunista é o de uma universidade. Uma infraestrutura de computação na nuvem pode ser montada usando recursos já existentes em laboratórios, além de máquinas de professores e funcionários. Desta forma, essa abordagem tem a finalidade de não atrapalhar o trabalho do aluno ou professor que esteja utilizando o computador, para isso foram desenvolvidas diversas técnicas que fazem uma leitura no computador e informa a sua porcentagem de uso.

A estrutura desse trabalho se inicia com a definição do problema a ser estudado. Depois definimos os objetivos gerais e específicos, é caracterizada a fundamentação teórica usada como base para a construção desse projeto. Após isso, é comentado sobre as aplicações escolhidas para a construção da nuvem, assim como suas respectivas instalações e configurações. Logo depois, é feita uma discussão de viabilidade do projeto ser implantado no ambiente de rede que temos na universidade. Finalmente, é apresentado a conclusão e trabalhos futuros.

1.1 DEFINIÇÃO DO PROBLEMA

Nesse trabalho, abordaremos a análise de viabilidade da instalação de uma nuvem computacional em máquinas *desktop* de uma universidade, usando máquinas não dedicadas para fazer experimentos, simulações, rodar programas com necessidade de alto processamento, etc. Para isso, essas máquinas serão utilizadas apenas quando estiverem ociosas. Assim, estaremos utilizando o conceito de nuvem oportunista, que visa utilizar o computador da melhor forma possível, fazendo apenas uso de recursos necessários quando possível, de modo que não atrapalhe os usuários.

1.1.1 Objetivos Gerais

A principal meta desse trabalho é avaliar se é viável a implantação de uma nuvem computacional oportunista em máquinas *desktop* de uma universidade, onde todos os alunos e professores poderão utilizá-la para realização de projetos e experimentos. Para que isso aconteça, é preciso estudar quais ferramentas são apropriadas, levantar requisitos sobre a nuvem, além de instalar e configurar uma aplicação que gerencie toda a nuvem e suas máquinas virtuais.

1.1.2 Objetivos Específicos

- Estudar ferramentas de gerência de nuvem, pesquisando quais as mais famosas e usadas pela comunidade acadêmica.
- Instalar, configurar e implantar uma nuvem que possa ser capaz de aumentar a capacidade de processamento de aplicações e ferramentas de simulação de sistemas, obtendo um melhor resultado comparando-se em usar apenas uma máquina.
- Fazer uma estimativa da capacidade computacional de uma possível nuvem implantada nos laboratório de computação da UFPB – Campus IV, calculando quantas máquinas virtuais podem ser levantadas em todo o laboratório, além do total de recursos disponíveis para a nuvem (ex: CPU, memória e disco).
- Analisar qual o horário que as máquinas menos são utilizadas. Com o término dessa atividade, esperamos encontrar os horários mais vagos no laboratório para que seja possível fazer uso dos computadores.

2 FUNDAMENTAÇÃO TEÓRICA

Nesta sessão, é explicada a base teórica para construção desse projeto. A computação em nuvem é um modelo de computação novo, mas que têm algumas de suas características semelhantes a alguns modelos mais antigos como computação em grade, nesta parte do trabalho entenderemos o porquê disso.

2.1 COMPUTAÇÃO EM NUVEM (*Cloud Computing*)

Atualmente, esse é um dos paradigmas de computação mais utilizados e falados na comunidade científica e no mercado de computação, no qual a infraestrutura e recursos de *hardware* e *software* são oferecidos como serviço. Esse termo já se tornou global e atende a uma grande massa, que começa com simples usuários finais que hospedam seus arquivos, até grandes empresas que terceirizam sua administração de TI. Segundo Taurion (2009), o termo computação em nuvem surgiu em 2006 em uma palestra de Eric Schmidt da Google, sobre como sua empresa gerenciava seus *datacenters.* Neste tipo de modelo, uma das suas principais funções é a entrega personalizada de uma infraestrutura de TI, *software* e aplicações como serviços, onde pode ser definido como nuvem pública que os usuários pagam por tempo de uso de recursos computacionais (consumo de tempo de CPU, espaço em disco, transferência de dados, memória), ou pode ser determinado como nuvem privada, onde se busca uma implementação de servidores internos, permitindo uma arquitetura com mais segurança, gerenciamento e configuração, porém exige-se um investimento inicial bem maior do que a escolha da nuvem pública.

De acordo com a NIST (*National Institute of Standards and Technology*), a computação em nuvem pode ser definida como "um modelo que permite acesso à rede de forma onipresente, conveniente e sob demanda a recursos computacionais compartilhados e configuráveis (ex: redes, armazenamento, aplicações e serviços) que podem ser rapidamente provisionados e liberados com um esforço mínimo de gestão ou interação com o provedor de serviços. Segundo esta definição, este tipo de abordagem nos traz uma série de vantagens que chamam a atenção para seu uso, pois é simples e barato". Usar computação em nuvem, diversas vantagens são alcançáveis, como essas descritas a seguir:

- **Compatibilidade com Sistema Operacional, *Software ou Hardware:*** Através de sua extrema compatibilidade, usuários não precisarão se adaptar para usar, pois as

aplicações deverão dar suporte a todos os programas necessários com atualização automática, onde o sistema será capaz de detectar quais programas dependem daqueles atualizados e gerar uma cascata de instalação para todos os *softwares.*

- **Maior produtividade no desenvolvimento de aplicações:** De acordo com a necessidade do negócio recursos podem ser adicionados linearmente. A implementação de uma nova aplicação não precisa esperar por *upgrades*. Assim a infraestrutura responde conforme à demanda, as variações do volume de negócios permitem acelerar o desenvolvimento.

- **Oportunidade de novo rumo de trabalho para vendedores e desenvolvedores:** Enquanto aplicações são vendidas para *hardwares* específicos, a nuvem consegue englobar a maioria dos usuários. O desenvolvedor que escolhe seu ambiente de hospedagem de dados, além de economizar tempo e dinheiro no desenvolvimento, teste, atualização, correção e implantação do sistema.

- **Criar novos negócios:** Uma empresa inovadora tem muito mais vantagem em cima de organizações que sempre encaram os problemas da mesma forma, uma maneira inteligente seria incrementar ou criar produtos que se encaixam na necessidade do negócio.

Na internet o agente que fornece serviços de computação para usuários finais, são chamados de Provedor de Serviços (PS). Os usuários finais são chamados de Usuário de Serviços (US). Cada usuário possui sua própria infraestrutura de acordo com sua necessidade. Na abordagem de computação em nuvem essa estrutura é determinada como serviço por um Provedor de Infraestrutura (PI). Esses três tipos diferentes de agentes PS, US e PI são agente básicos desse tipo de modelo como mostra a Figura 1:

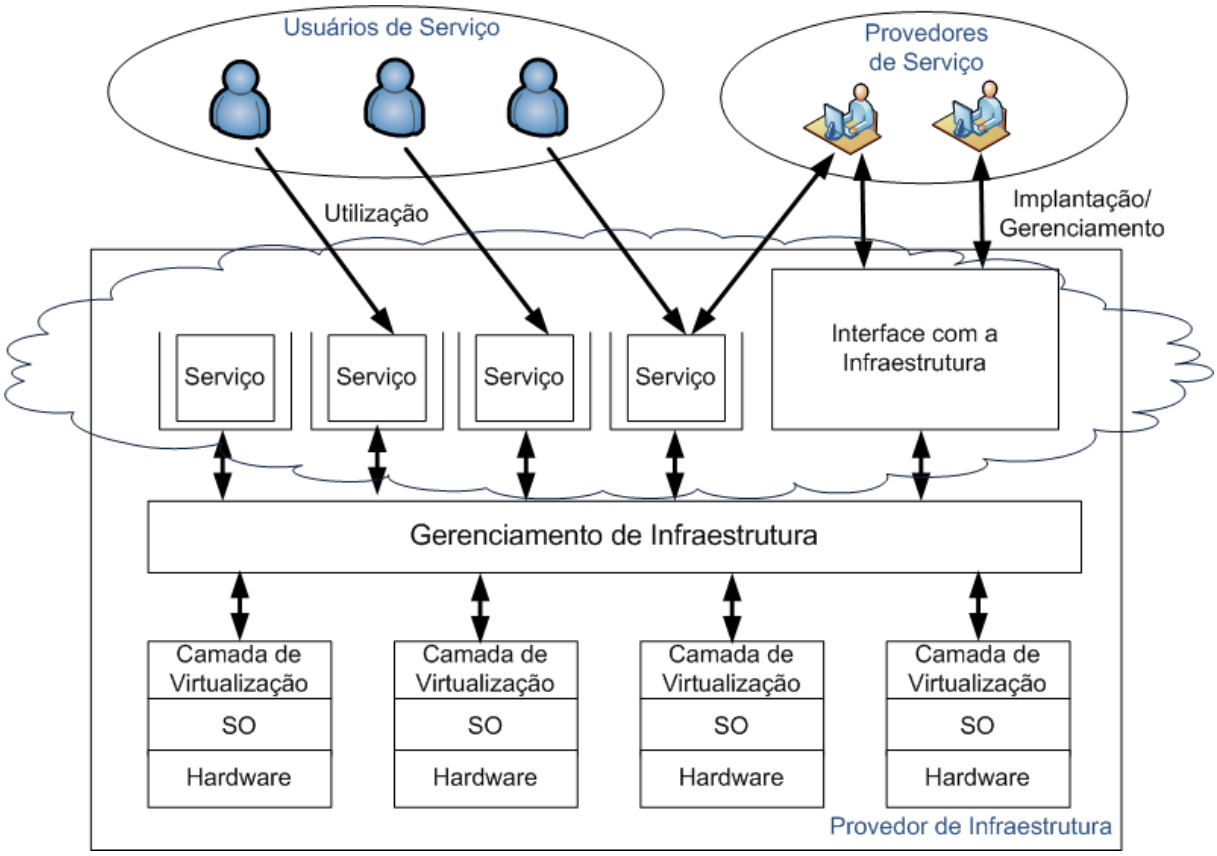

Figura 1 - Agentes de um Sistema de Computação em Nuvem. (Vaquero L.M. 2008)

2.2 ANATOMIA DA COMPUTAÇÃO EM NUVEM

O uso de virtualização em nuvens criou um novo tipo de arquitetura, ela é dividida em três camadas: aplicação, plataforma e infraestrutura. Esse tipo de estrutura define um novo modelo de desenvolvimento de aplicações e encapsulam recursos sob demanda, e seus serviços podem ser vendidos em base no "pago por uso". Os cenários de uma arquitetura desse tipo de abstração divide-se nas seguintes definições:

2.2.1 Software como Serviço (*Software as a Service - SaaS*)

O SaaS envolve diversas características, como recursos humanos, contabilidade, e-mail e permite ao cliente a execução de aplicações na nuvem, em alternativa à execução local dessas aplicações, junto com aplicativos que são providos por empresas como Salesforce, NetSuite e Google. Nesta camada há acordos de nível de serviço (*Service Level Agreement - SLA*) pode ser feita dinamicamente por um provedor de infraestrutura para seus clientes, dependendo da demanda que o mesmo precisar.

2.2.2 Plataforma como Serviço (*Plataform as a Service - PaaS*)

É uma camada abstraída que é responsável pelo encapsulamento de um ambiente de desenvolvimento de serviços. Segundo (Cezar Taurion 2009) a *PaaS* pode ser definida da seguinte forma "uma plataforma para criar e operar aplicações, incluindo ferramentas de desenvolvimento, administração e gerenciamento, além de serviços de *runtime*, tudo na modalidade *SaaS".* Os recursos são apresentados ao cliente de uma forma transparente, pode oferecer suporte as fases de desenvolvimento e testes, mas pode ser específica, como gerência de configuração. Um exemplo desse tipo de serviço é o *Google App Engine* que é uma plataforma de hospedagem e desenvolvimento de aplicativos.

2.2.3 Infraestrutura como Serviço (*Infrastructure as a Service - IaaS*)

É a camada de mais baixo nível encontrada em uma arquitetura de *cloud computing*, e é responsável por distribuir serviços padronizados de capacidade computacional e armazenamento através da rede. É uma evolução que não requer nenhum compromisso de longo prazo e permite aos usuários provisionar recursos sob demanda. O provedor de IaaS faz um pouco de gestão para manter o centro operacional de dados, os usuários devem implantar e gerenciar os serviços de *software.* Escala de largura de banda, memória e armazenamento são recursos que geralmente são incluídos no orçamento desse tipo de serviço.

2.3 TIPOS DE NUVENS

A computação em nuvem pode ser implementada de três formas: pública, privada ou híbrida. Dependendo dos tipos de dados, com o que ela está trabalhando e os diferentes níveis de segurança e gerenciamento requerido.

2.3.1 Nuvem Pública

Esse tipo de prática oferece um ótimo nível de eficiência em recursos compartilhados, no entanto, elas são mais vulneráveis do que nuvens privadas. Neste modelo não podem ser aplicadas restrições de acesso quanto ao gerenciamento de redes ou técnicas de autenticação e autorização. As nuvens públicas são aquelas que são executadas por terceiros. As aplicações

de diversos usuários ficam misturadas nos sistemas de armazenamento, o que pode parecer ineficiente a princípio. Porém, se a implementação de uma nuvem pública considera questões fundamentais, como desempenho e segurança, a existência de outras aplicações sendo executadas na mesma nuvem permanece transparente tanto para os prestadores de serviços como para os usuários. (Fernando Seabra, 2009). Para a escolha desse tipo de nuvem, as seguintes características são essenciais: Padronização de carga de trabalho para aplicações que são usadas por muitas pessoas, aplicação de *SaaS* feita por um vendedor que tenha implementado estratégia de segurança e a criação de projetos colaborativos, e a terceirização de serviços, ou seja, qualquer organização pode alugar uma nuvem a outra, economizando o uso de vários recursos.

2.3.2 Nuvem Privada

São construídas por um único usuário (empresa ou instituição) que queira ter sua própria gerência. É bem apropriado para aplicações permanentes que demandam níveis específicos de segurança, qualidade de serviço e localização de dados. Neste modelo, o cliente possui o *data center*, armazenamento, rede, *hardware*, e decide quais são os usuários que tem permissão para acessar seu sistema.

2.3.3 Nuvem Híbrida

É uma combinação das duas nuvens citadas anteriormente, permite fazer uma junção das características da nuvem pública com a privada. Com isso, os níveis de serviço podem ser mantidos mesmo que haja muito uso dos recursos. É importante destacar que nuvens híbridas podem ser complexas, devido a maneira de determinar a forma com que as aplicações são diferenciadas entre nuvens públicas e privadas. Fazer processamento de dados de uma aplicação que tenha muitos dados, pode perder desempenho, pois terá que trafegar esses dados de uma rede privada para uma pública, e isso pode ser muito caro.

2.4 COMPARAÇÕES COM MODELOS RELACIONADOS

A computação em nuvem é uma abordagem nova e antes dela foram construídos alguns modelos que freqüentemente são confundidos com esse tipo de abstração, a seguir é caracterizado alguns deles.

2.4.1 Computação em Grade (*Grid Computing*)

É um sistema que compartilha recursos de forma descentralizada, que usa protocolos e interfaces que garante um QoS de nível elevado. Algumas características como redução de custos e aumento da flexibilidade são considerados objetivos similares aos dois modelos. Segundo (Vaquero et al. 2009). Alguns pontos mais importantes que diferenciam são apresentados a seguir.

- **Alocação dos Recursos:** Enquanto a computação em grid realiza um compartilhamento dos recursos entre usuários, a computação em nuvem só aloca um recurso a um determinado usuário caso ele queira usá-lo; logo, isso sugere a ideia de que o recurso é totalmente dedicado àquele usuário; além disso, não ocorre propriamente um compartilhamento de recursos na computação em nuvem, devido ao isolamento realizado através de virtualização.

- **Plataformas e dependências:** As nuvens permitem que os usuários usem *softwares* independentes de um determinado domínio; ou seja, os *softwares* rodam em ambientes customizados, e não padronizados; os *grids*, ao contrário, só aceitam aplicações que sejam executáveis em seu sistema.

- **Escalabilidade:** Tanto os *grids* como as nuvens lidam com as questões de escalabilidade; nos *grids*, o usuário manualmente habilita a escalabilidade através do aumento do número de nós utilizados; nas nuvens, por outro lado, escalabilidade é automática, requerendo uma reconfiguração dinâmica.

2.4.2 Computação Utilitária (*Utility Computing*)

Esse tipo de modelo se refere a habilidade de mediar o serviço e carga para o uso exato dos consumidores. Podemos comparar essa definição com o fornecimento de água que temos em casa, no qual pagamos a quantidade que usamos. Apesar da *cloud computing* também utilizar esse mesmo tipo de cobrança, pode crescer dinamicamente a sua capacidade,

habilitando a possibilidade de alocar recursos sob demanda, em tempo real, com isso, aplicações não se limitam a capacidade de uma infraestrutura fixa.

2.5 VIRTUALIZAÇÃO

A virtualização é considerada um fundamento para a computação em nuvem. É capaz de auxiliar em um ambiente de trabalho que permita diversas plataformas de sistemas operacionais sem a necessidade de aumentar a estrutura de *hardware*, com isso, cada aplicação desenvolvida pode ter suas bibliotecas instaladas no sistema operacional apropriado para ela e pode ter sua máquina virtual própria, que executam uma infraestrutura de *hardware* comum, compartilhando todos os seus recursos.

As máquinas virtuais, por emularem um ambiente computacional sobre outro impõem algumas restrições de implementação e de desempenho. É aqui que entra o desenvolvimento dos produtos de *software* para a virtualização. Basicamente, as máquinas virtuais podem ser implementadas como uma aplicação de um sistema operacional e executarem em modo usuário, ou serem uma camada de *software* posicionada entre o *hardware* da máquina e o sistema operacional. A primeira opção é o que se denomina de máquina virtual de processo e a segunda de monitor de máquina virtual ou *hypervisor*. (Smith e Nair, 2005). Para uma utilização mais eficiente, ou seja, mais trabalho realizado pelo mesmo *hardware*, é preciso fazer uma boa distribuição de seus recursos (memória, disco, processador, etc) entre as diferentes aplicações utilizadas. Cada programa tem a percepção que todos os recursos da máquina estão disponíveis só para ele, quando na verdade está sendo compartilhado com vários outros. Assim, obtendo mais transparência sobre recursos físicos para usuários finais, isso é feito criando interfaces gráficas que é uma camada de mais alto nível do sistema. Uma definição bastante encontrada em livros, é a de uma camada de software inserida entre o hardware e as aplicações que executam tarefas para os usuários e cujo objetivo é tornar a utilização do computador, ao mesmo tempo, mais eficiente e conveniente. (Silberchatz, 2001).

3 ESTUDO DE APLICAÇÕES DE GERÊNCIA DE NUVEM

Para fazer a construção da nuvem, foi feito um levantamento de ferramentas que são capazes de fazer isso. Depois de uma pesquisa, foi concluído que o *Eucalyptus* e o *OpenStack*, são as duas principais ferramentas no mercado atual. Nas próximas seções explicaremos cada uma detalhadamente.

3.1 EUCALYPTUS

O Eucalyptus é uma ferramenta que possibilita a criação de nuvens privadas e públicas usando o sistema operacional Linux. Com uma arquitetura bastante modular, o *Eucalyptus* possui componentes internos que consistem em serviços Web, tornando-os fáceis de serem modificados e expandidos. Um fator muito importante que motiva o uso dessa ferramenta é a ausência de plataformas abertas para o estudo científico de Computação em Nuvem, já que a maioria das soluções são proprietárias e não são propícias à realização de experimentos. O usuário desse arcabouço se comunica com o sistema fazendo chamadas à API Amazon EC2, que já se posiciona com o padrão de nuvem no mercado. Também pode ser definido como uma infraestrutura de *software,* que faz uso de vários recursos computacionais que está ao alcance de pesquisadores, como *clusters* e *workstation farms,* e foi construída com código aberto. Possui como meta duas características principais: uma delas é a capacidade de ser implantado em ambientes de infraestrutura de TI que não estão sob controle de seus criadores; a outra se refere à extensibilidade - como essa ferramenta é modularizada, permite fácil manutenção de um componente ou até sua substituição. No início de 2009, começou o processo de comercialização do *Eucalyptus* como uma empresa de código aberto. Logo após, em 23 de abril, o Ubuntu lançou o Ubuntu *Enterprise Cloud* utilizando o mais novo suporte para EBS (*Elastic Block Storage*) dessa ferramenta. A partir dai, esse projeto ficou muito conhecido e atualmente usado por grandes empresas como: AWS (*Amazon Web Services*), *Dell, HP, Intel, redhat* e CERN (Organização Nuclear Européia para Pesquisa Nuclear). Outra motivação para o uso foi a ideia de incluir o módulo de controlador de nó feito pela equipe do *Ourgrid* que usa os recursos da nuvem de forma mais oportunista, ou seja, aproveitando melhor os recursos dos computadores.

3.1.1 Arquitetura do Eucalyptus

Alguns recursos como pequenos *clusters*, conjunto de *workstations* e várias máquinas virtuais ou desktop são disponibilizados para os usuários. De acordo com (G. C. Drive, 2009) hosts internos somente podem se comunicar com seu respectivo nó *front-end* ou com os demais hosts internos através de uma rede privada. Os nós *front-end* são escolhidos por administradores da rede e são responsáveis por rotear as informações públicas para os hosts internos. Esses tipos de host são definidos porque os endereços de IP são escassos e o acesso público a toda ramificação dos hosts da nuvem interna pode ser perigosa. A nuvem do *Eucalyptus* é separada em quatro componentes. A seguir é detalhado cada um deles.

- **Controlador de Nó (*Node Controller*):** Tem a responsabilidade de executar os recursos físicos de uma máquina virtual. Portanto, é possível alocar vários *Node Controllers* em máquinas virtuais e apenas um numa máquina física. Segundo (Alanis et al, 2013) esse componente tem como principais funções controlar atividades da VM, incluindo a execução, monitoramento, finalização das instâncias, gerência do ponto da rede virtual, coleta de dados relacionados à disponibilidade dos recursos e entrega de relatórios para o C*luster Controller.*

- **Controlador de Cluster (*Cluster Controller*):** É responsável por rotear pacotes entre redes virtualizadas externa (pública) e a rede interna(privada). Esse componente gerencia um conjunto de *Node Controllers.* Para realizar suas operações, são armazenadas informações sobre máquinas virtuais de um *Cluster* para fazer o escalonamento das requisições de criação de instâncias de máquinas virtuais entre os *Node Controllers* utilizados.

- **Controlador de Armazenamento (*Storage Controller - Walrus*):** Componente que realiza comando de *post/get* no serviço de armazenamento que está implementado na interface da Amazon's S3, com a funcionalidade para salvar e recuperar imagens de máquinas virtuais e dados de usuários.

- **Controlador de Nuvem (*Cloud Controller*):** Tem a função de processar requerimentos de administradores e usuários. De acordo com (Patricia et al, 2013), esse componente pode tomar decisões de programação, processos de autenticação, monitoramento de disponibilidade de recursos em diversos componentes da infraestrutura, incluindo nós e controladores de *cluster*.

Na figura 2, é apresentado a comunicações dos componentes do *Eucalyptus*.

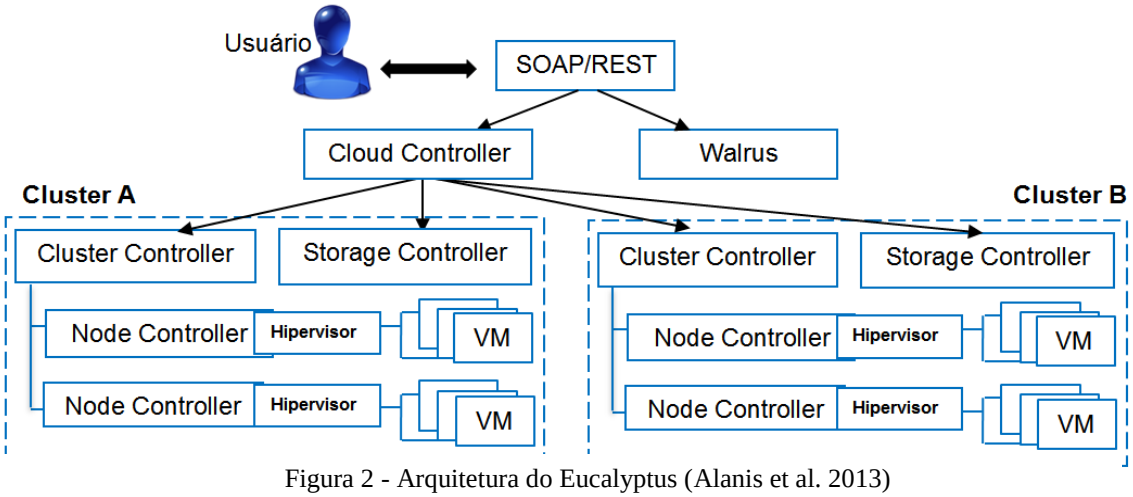

Figura 2 - Arquitetura do Eucalyptus (Alanis et al. 2013)

3.2 OPENSTACK

O projeto *Openstack* tem como objetivo criar uma plataforma de nuvem computacional de código aberto para nuvens públicas e privadas, tendo como meta a escalabilidade sem complexidade. Esse projeto começou através do trabalho de duas organizações: *Rackspace Hosting* (grande firma de hospedagem dos Estados Unidos) e a NASA. Elas decidiram juntar forças para liberar seus projetos de armazenamento baseado em objetos em uma nuvem interna e o código base para computação em nuvem como um projeto de código aberto. Como consta na sua descrição, "o *Openstack* foi projetado para prover flexibilidade para o design da sua nuvem, sem requisitos de *hardware* ou *software* proprietário e com capacidade de integração com sistemas e tecnologias herdadas de terceiros" (OpenStack, 2013). A primeira versão chamada de "Austin" foi entregue ao público em novembro de 2010. A fundação do *OpenStack* já atraiu mais de 7.000 membros individuais de 100 países, 850 organizações diferentes e já arrecadou mais de US $ 10 milhões. Tem como objetivo servir desenvolvedores, usuários e todos relacionados, fornecendo um conjunto de recursos compartilhados para crescer com a nuvem pública e privada, disponibilizando a tecnologia para vendedores e ajudando os desenvolvedores na produção de um melhor *software* de nuvem na indústria.

3.2.1 Comunidade

A comunidade do *Openstack* é muito grande. Ela foi criada pelos usuários finais com a participação ativa de vendedores de muitos outros projetos de código aberto. Atualmente, a comunidade do *Openstack* possui um comitê com 5600 usuários, 850 organizações, 550 desenvolvedores, 5 diretores, 8 gerentes de projeto e já foram feitos 300 mil downloads (Openstack, 2013).

3.2.2 Arquitetura do Openstack

O *OpenStack* se divide em três componentes principais, que são eles: *Nova*, *Glance e Swift.* Mais detalhes sobre esses componentes são dados a seguir.

3.2.2.1 Componente *Nova*

O objetivo desse componente é prover um *framework* para gerenciamento de instâncias de máquinas virtuais de larga escala. Semelhante a funcionalidade e escopo do serviço da *Amazon's EC2,* através desse componente é possível criar, gerenciar e parar servidores virtuais baseado no próprio sistema de imagens do usuário, através de uma API programável. Recursos do computador podem ser acessados pela API por desenvolvedores da nuvem e pela interface web usada por administradores e usuários. Os administradores da nuvem freqüentemente implantam o *nova* usando um dos muitos *hypervisores* suportados no ambiente de virtualização. O KVM e XenServer são as escolhas mais populares e recomendadas na maioria dos casos. Há duas partes essenciais da arquitetura: a pesquisa de mensagem e o banco de dados. Essas duas partes facilitam a sincronização de tarefas complexas através da passagem de mensagem e compartilhamento de informações, como mostra a figura 3.

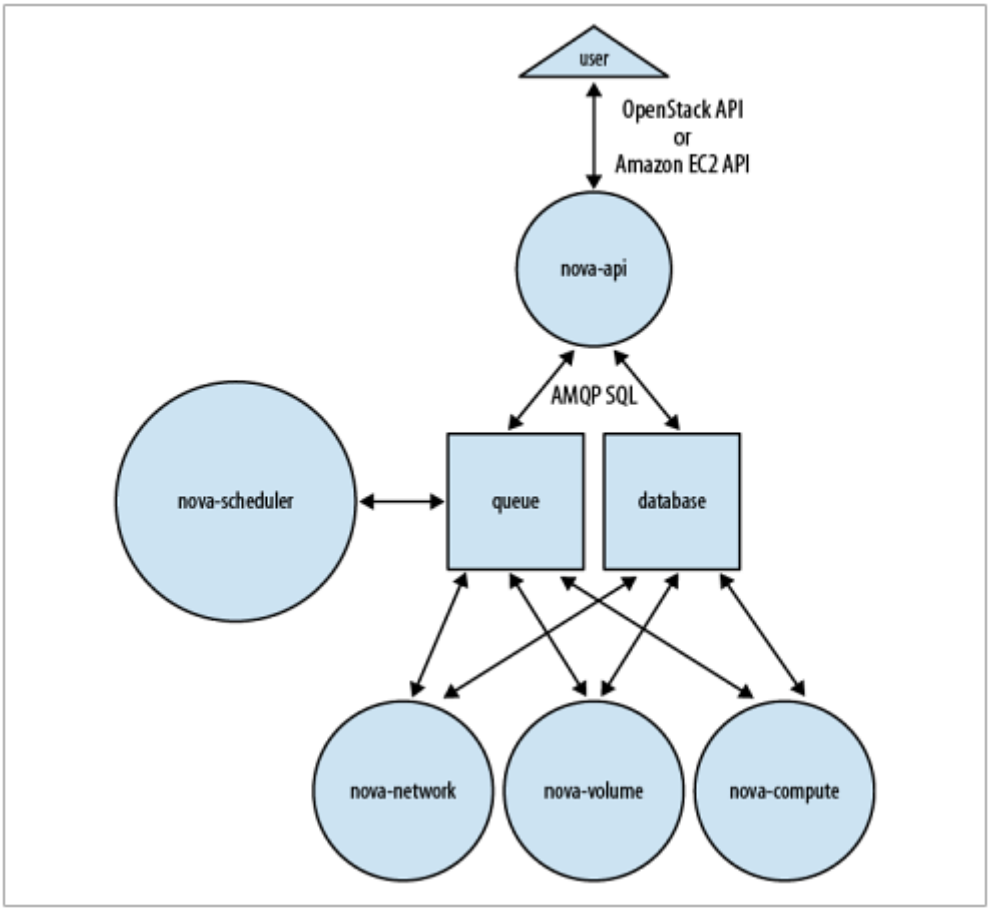

Figura 3 - Arquitetura Lógica do Nova. (Pepple K. 20011)

De acordo com (Pepple K. 2011) esse diagrama pode ser detalhado da seguinte maneira:

- Usuários finais que querem usar o *Nova* para criar instâncias podem chamar a *nova-api* como a API do *OpenStack* ou requisitando a API da *EC2*.

- *Daemons* do *Nova* trocam informações através de uma consulta (ações) e banco de dados para transportar requisições dessa API.

De acordo com a documentação do *OpenStack*, alguns casos de uso referentes ao componente *Nova* são descritos a seguir são populares:

- **Provedores de Serviços:** Oferecendo uma plataforma IaaS ou serviços de mais alto nível;

- **Provedores de Serviços:** Disponibilizando uma plataforma *IaaS* ou serviços de mais alto nível;

- **Departamentos de TI:** Atuando como prestadores de serviços da nuvem para unidades de negócios e equipes de projeto.

- **Processamento de grande volume de dados:** Ferramentas como *Hadoop* (Projeto da Apache para processamento de altas cargas de dados).

Acreditamos que este tipo de ambiente também é adequado para o uso em laboratórios de universidades, para que atividades de ensino e pesquisa possam usufruir de tal infraestrutura, usando máquinas desktop já existentes quando estiverem ociosas.

3.2.2.2 Componente *Glance*

A arquitetura do *glance* é dividida em três partes: *glance-api, glance-registry* e *image store. Glance-api* aceita chamadas à API, como na *nova-api*, e atualmente os metadados da imagem são inseridas no *image store.* O *glance-registry* armazena e recupera metadados de imagens. O banco de dados do *glance* contém apenas duas tabelas: *Image* e *Image Property*. A tabela *Image* representa a imagem no *datastore* (formato de disco, tamanho, etc), enquanto a tabela de *Image Property representa* os metadados da imagem. Ele também aceita requisições que vem da API e se comunica com outros componentes (*glance-registry* e *image store*) para facilitar a pesquisa, recuperação, envio, ou exclusão de imagens. A porta padrão desse componente é a 9292. De acordo com a documentação (*Openstack,* 2013), o projeto *glance* provê serviços para descoberta, registro e recuperação de imagens de máquinas virtuais. Ele oferece uma API RESTful que permite consultas de metadados das imagens de VMs, além da recuperação da imagem atual. O *glance* foi escrito com as seguintes diretrizes:

- Componentes baseados na arquitetura: Comportamentos adicionados rapidamente;

- Altamente disponível: Escala para cargas de trabalho muito alta;

- Tolerante a falhas: Processos isolados para evitar falhas em cascata;

- Recuperável: As falhas devem ser fácil de diagnosticar, "depurar" e corrigir;

- Padronização Aberta: O usuário pode ser referência na documentação

Podemos entender melhor a arquitetura do *glance* na figura 4.

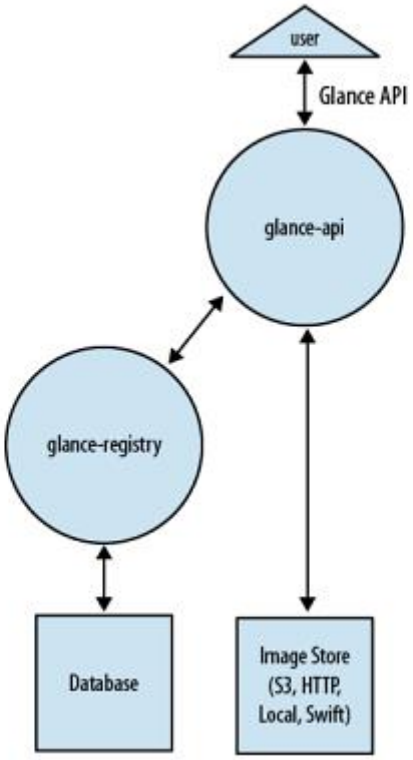

Figura 4 - Arquitetura Lógica do Glance. (Pepple K. 2011)

3.2.2.3 Componente *Swift*

Swift é o projeto mais velho e provavelmente o mais maduro do *OpenStack.* É a tecnologia subjacente que dá poder aos serviços do *Rackspace's Cloud Files* (projeto de computação em nuvem da *Rackspace* que oferece serviços como *PaaS,* armazenamento em nuvem, servidor virtual privado e etc.), esse componente só interage com o *Nova* de forma tangencial. O *Swift* é configurável de acordo com o número de cópias dos componentes que são escritas, bem como a forma que as zonas são configuradas. Cópias são réplicas dos dados armazenados supervisionados pelos auditores, que checam o conteúdo de bit a bit dos dados. A zonas são espaços isolados no disco, onde cada uma é totalmente independente da outra, deixando-a livre de complicações no caso de uma interrupção em alguma zona.

Esse componente faz o balanço da escrita dos objetos com o armazenamento dos objetos nos servidores, distribuindo a escrita e leitura. Isso é ilustrado na figura 5:

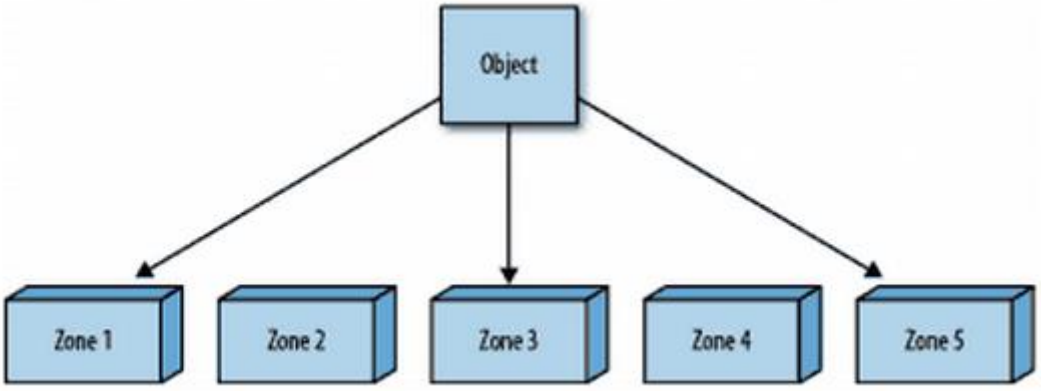

Figura 5 - Swift - Réplicas e Zonas. (Pepple K. 2011)

A arquitetura do *Swift* pode ser dividida em duas partes lógicas: apresentação e recursos. Ele aceita que usuários finais façam processos de requisições via *swift-proxy*, dai passam para o objeto apropriado, conta ou processos para conclusão. O *Swift* é altamente disponível, distribuído e eventualmente consistente no armazenamento de objetos. As organizações podem usá-lo para armazenar grandes quantidades de dados de forma eficiente, segura e barata, (*Openstack, 2013*). Os principais componentes como *data store, swift-proxy, swift-container, swift-object, swift-account* e as interações são ilustrados na figura 6.

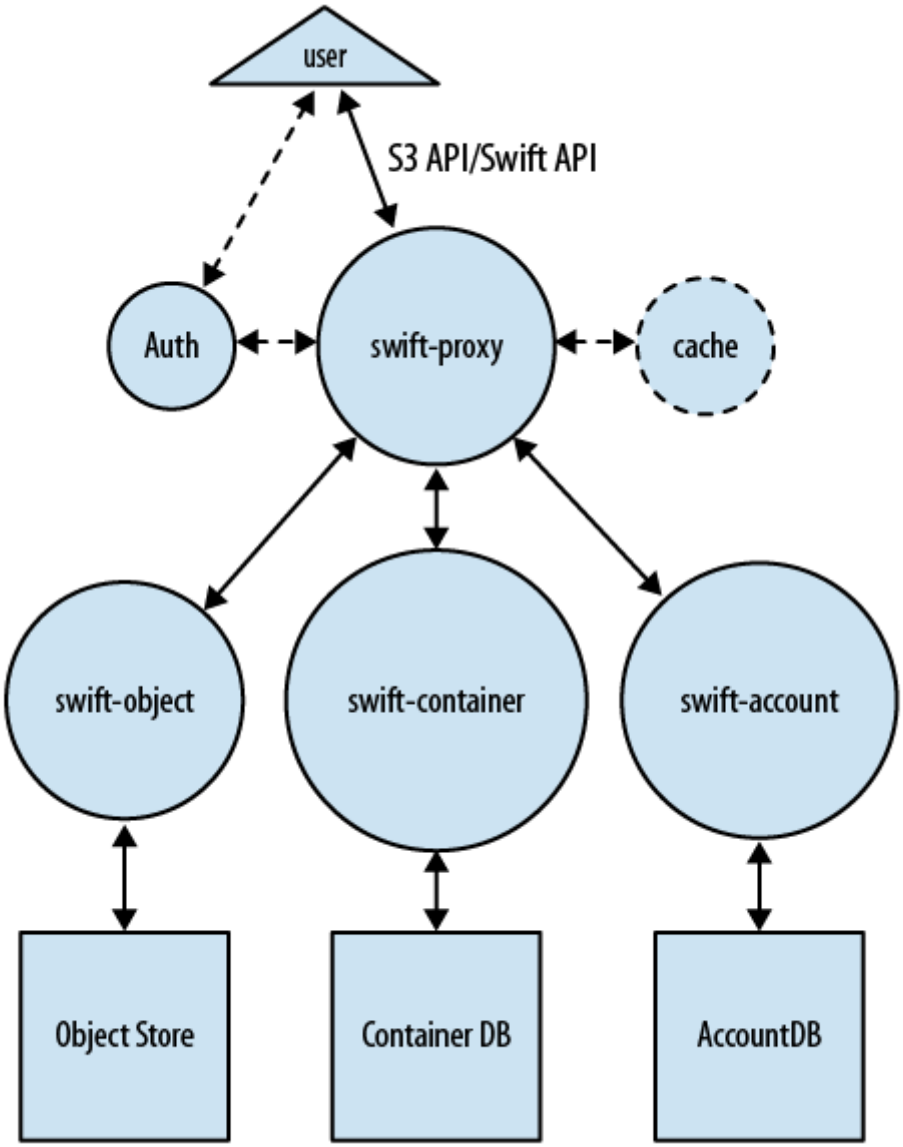

Figura 6 - Arquitetura Lógica do Swift. (Pepple K, 2011)

Swift possui vários recursos que se comunicam entre si, segue um detalhamento sobre os serviços mais importantes desse componente (OpenStack, 2013):

- *Swift-Proxy:* Responsável por fazer toda a ligação da arquitetura. Um grande número de falhas são tratadas no *proxy*. Por exemplo, se um servidor está indisponível para um objeto PUT, ele perguntará para o *ring* (explicado a seguir) por um servidor *handoff* (transição de uma unidade móvel) e a rota. Quando objetos são transmitidos de ou para o servidor de objetos, eles são transmitidos diretamente através do *proxy* de ou para o usuário.

- *Ring:* Mapeia os nomes das entidades que estão no disco e suas respectivas localizações físicas. Cada parte do *swift* como *account*, *containers* e *objects*, possuem um

ring. Quando algum componente precisa realizar operações com outro, ele precisa requisitar o *ring* correto que determina a localização no *cluster* que ele está à procura.

- *Swift-Object:* Servidor simples de armazenamento que tem opções básicas, como recuperação, persistência e exclusão de objetos em um dispositivo local. Os objetos são salvos como arquivos binários com metadados armazenados.

- *Swift-Container:* Responsável por listar todas os objetos armazenados.

- *Swift-Account:* Lista os *containers* alocados.

- *Replication:* Tem como principal função manter o sistema em funcionamento diante de falhas.

- *Updaters:* Processa atualizações que falharam inicialmente, como pode acontecer em filas de processos com alta carga.

- *Auditors:* Checa a integridade dos outros serviços. Se algum bit de um objeto estiver corrompido o mesmo entra em quarentena e é substituído por outro.

3.2.2.4 Outros Recursos

Alguns outros recursos que fazem parte da arquitetura do *OpenStack* são descritos a seguir:

- **OpenStack Networking (Q*uantum*):** Possui a capacidade de gerenciamento de IPs e API dirigida a rede. Permite que os usuários criem suas próprias redes, além de disponibilizar a capacidade de designar IP estático, *floating* ou dinâmico para instâncias de VM.

- **OpenStack DashBoard (*Horizon*):** Possui uma interface web para gerenciamento do sistema feito por administradores e usuários, como gerenciamento de imagens de VM, de ciclo de vida de uma instância e de armazenamento.

- **OpenStack Identity (*Keystone*):** Serviço de gerenciamento centralizado para contas de usuários como autenticação e sistema de controle de acesso. Provê o acesso a registros de serviços implantados pelo *OpenStack* no *data center* e nos pontos finais de comunicação.

4 INSTALAÇÃO E CONFIGURAÇÃO DAS APLICAÇÕES

Nesse capítulo vamos fazer o processo de instalação e configuração das ferramentas escolhidas e explicadas anteriormente. Cada uma tem suas particularidades, mas alguns serviços são semelhantes.

4.1 EUCALYPTUS

Para fazer a instalação do *Eucalyptus*, é preciso configurar vários recursos que a mesma necessita para funcionar corretamente. Serão descritos aqui os pontos relevantes de configuração. A instalação e configuração desse aplicativo serão apresentadas da forma mais detalhada possível, pois está diretamente ligada à materialização do ambiente.

Todos os softwares e sistemas utilizados nesse trabalho são de licenças livres e possuem código aberto, não constituindo seu uso infração de distribuição não autorizada ou violação de *copyright.* Os passos para instalação do *Eucalyptus* foram baseados na documentação do mesmo e materiais citados na referência.

4.1.1 Requisitos para uso

A seguir, os requisitos para o uso dessa ferramenta são detalhados (Alanis et al, 2013):

- **Máquina Física:** Todos os componentes devem ser instalados em máquinas físicas, e não em máquinas virtuais.

- **CPUs:** É recomendado que cada máquina da nuvem do *Eucalyptus* contenha um processador Intel ou AMD com o mínimo de dois cores de 2GHz.

- **Sistema Operacional:** O *Eucalyptus* suporta as seguintes distribuições do Linux: CentOS 5 (5.6 e maior), CentOS 6, RHEL 5 (5.6 e maior), RHEL 6, Ubuntu 10.04 LTS e Ubuntu 12.04 LTS.

- **Relógio da Máquina:** Cada máquina usada como componente e cada relógio das máquinas devem ser sincronizados (como exemplo o uso do NTP (*Network Transport Protocol*). Esses relógios devem ser sincronizados todo o tempo e não apenas na instalação.

4.1.1.1 Requisitos de Memória e Armazenamento

Para que a ferramenta tenha um bom desempenho na execução de suas tarefas é recomendado usar as seguintes configurações:

- Cada máquina na rede precisa no mínimo de 30 GB de espaço.

- Para as máquinas que irão executar os componentes de Walrus e SC(*Storage Controller*) é preciso de 100 GB.

- É recomendado um espaço livre de 50-100 GB no disco para cada host NC. Quanto maior o espaço livre no disco maior o número de VMs.

- Cada máquina na rede precisa de no mínimo 4 GB de memória RAM. Mas é recomendado que mais memória seja armazenada em cache.

4.1.1.2 Requisitos de Rede

Em termos de conexão com a rede, são dadas as seguintes recomendações:

- Todas as máquinas necessitam ter conectividade com a internet e uma rede interna bem configurada.

- Todos os componentes do *Eucalyptus* devem ter pelo menos um NIC (Controlador de Interface de Rede - *Network Interface Card*) que é um componente físico dentro do computador que coordena toda a comunicação entre a rede e o computador.

- Dependendo de algumas configurações, o *Eucalyptus* requer que a avaliação de dois conjuntos de endereços de IP. A primeira faixa de IP é privada, a segunda é pública,

para serem roteados para usuários finais e instâncias de VM. Ambos devem ser únicos para o Eucalyptus, não podendo ser usado por outros componentes ou aplicações da rede.

4.1.2 Instalação e Configuração de Dependências

A seguir é descrito algumas aplicações que são necessárias para o funcionamento do *Eucalyptus*.

4.1.2.1 Hypervisor

Também chamado de gerenciador de máquina virtual, o *hypervisor* é um programa que permite o compartilhamento de múltiplos sistemas operacionais em apenas um host. Tem como principal função controlar o processamento e recursos do host, alocando apenas o necessário para cada SO ligado e tendo certeza que cada VM não tenha a capacidade de romper outra. Segundo (MENEZES D. 2008), O *hypervisor*, ou Monitor de Máquina Virtual (VMM), é uma camada de *software* entre o *hardware* e o sistema operacional. O VMM é responsável por fornecer ao sistema operacional visitante a abstração da máquina virtual. É o *hypervisor* que controla o acesso dos sistemas operacionais visitantes aos dispositivos de *hardware*.

Nas versões do CentOS 5 e RHEL 5 a instalação deve usar o *Xen* como *hypervisor* e configuradas em cada host que irá executar o NC(*Node Controller*). Nas versões CentOS 6, RHEL 6, Ubuntu 10.04 LTS e Ubuntu 12.04 LTS deve ser instalado o KVM e configurado em cada host NC. Nesse trabalho escolhemos o Ubuntu 12.04 LTS como SO, já que é uma distribuição do Linux bastante popular, com comunidade bastante ativa, e é a versão instalada nos laboratórios de computação da UFPB - Campus IV. Usaremos como *hypervisor* o KVM, que é relativamente um novo projeto de código aberto, que é baseado na aquisição do Red Hat's da Qumranet em 2008. Se tornou o principal pacote de virtualização do Ubuntu, Fedora e outras distribuições do Linux. Para fazer a instalação do KVM no Ubuntu basta executar o comando abaixo:

```
$ apt-get install kvm
```

4.1.2.2 DHCP

Precisamos do DHCP no E*ucalyptus* pois, ele irá atribuir faixa de IPs para as máquinas virtuais e instâncias que criarmos. O Protocolo de Configuração Dinâmica de Host, é um protocolo designado pelas RFCs 2131 e 2132 de serviço TCP/IP que tem como principal função a configuração dinâmica de host, como atribuição de endereços IP, Máscara de sub-rede, Gateway Padrão, endereço IP de um ou mais servidores DNS, endereços IP de um ou mais servidores WINS e sufixos de pesquisa do DNS. Este protocolo substitui o antigo BOOTP que se tornou limitado para exigências atuais.

Se a versão do DHCP da máquina onde irá executar os serviços de core for 4, é necessário adicionar a seguinte propriedade no arquivo /etc/apparmod.d/usr.sbin.dhcp :

```
capability dac_override
```

Essa configuração a mais é usada para mudar os privilégios de acesso, mudando o proprietário de "dhcp:dhcp" para "root:root".

4.1.2.3 Bridge

O dispositivo de *bridge* filtra dados trafegados no limite da rede, diminui a quantidade de tráfego em LAN, dividindo-o em dois segmentos. É preciso configurar uma *ethernet bridge* em todas as máquinas de *Node Controller (NC). Essa bridge* conecta seu adaptador de rede local a rede de *cluster*. Em uma operação normal, Ncs irão levantar máquinas virtuais quando as instâncias são inicializadas. Para fazer a configuração correta é necessário saber o nome da *bridge* do sistema. Nomes comuns no Linux são apresentados na tabela 1.

Nome da Bridge	Hypervisor
Xenbr0	Xen. 3.0x
eth0	Xen 3.2 ou acima
br0	KVM

Tabela 1 - Bridges do Ubuntu

Para verificar quais são as *bridges* que podem ser utilizadas é usado o seguinte comando:

```
$ brctl show
```

Normalmente no Ubuntu a saída para esse comando é:

bridge name	bridge id	STP enable	interfaces
br0	8000.000000000000	no	
virbr0	8000.000000000000	yes	

O pacote *bridge-utils* do Linux possui utilidades para configurar a *Ethernet bridge* no Linux. Ele pode ser usado para conectar vários dispositivos *Ethernet* juntos. A conexão é totalmente transparente um *host* pode se comunicar com outro *host* que está conectado a outro dispositivo *Ethernet.*

Instalação do pacote *bridge-utils:*

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

Para que não seja necessário ter que configurar o IP dinâmico que a máquina recebe toda vez que o computador seja inicializado, é uma boa prática configurar a interface de rede para ter um IP estático. Para configurar da interface de rede para usar IP estático, é preciso editar o arquivo /etc/network/interfaces. A configuração ficará semelhante a essa:

```
auto eth0
iface eth0 inet manual

auto br0
     iface br0 inet static
     address 192.168.0.12
     netmask 255.255.255.0
     network 192.168.0.1
     broadcast 192.168.0.255
     gateway 192.168.0.1
     bridge_ports eth0
     bridge_fd 9
        bridge_hello 2
     bridge_maxage 12
        bridge_stp off
```

As configurações de IPs usadas acima servem para comunicação da rede externa (pública) e serve como exemplo usado no laboratório de informática, isso depende diretamente do ambiente de rede que se rá configurado. Para que essa configuração seja executada é preciso reiniciar a interface de rede com o seguinte comando em um terminal do Linux:

```
$ /etc/init.d/networking restart
```

4.1.2.4 NTP

O NTP (*Network Time Protocol*) é um protocolo usado para sincronizar o relógios de computadores em uma rede. Desenvolvido por Dabid Mills da Universidade de Delaware, o NTP é agora um padrão da internet. O *Eucalyptus* requer que cada máquina tenha NTP inicializado e configurado para rodar automaticamente na inicialização. O primeiro passo é baixar e instalar o NTP nas máquinas que irão executar o *Eucalyptus*, para isso, é preciso rodar esse comando:

```
$ apt-get install openntpd
```

A seguir é preciso configurar e adicionar os servidores NTP escrevendo o texto a seguir no arquivo /etc/openntpd/ntpd.conf:

```
server 0.pool.ntp.org
server 1.pool.ntp.org
server 2.pool.ntp.org
```

O passo a seguir é configurar o NTP para rodar na inicialização do sistema:

```
$ chkconfig openntpd on
```

Agora precisamos iniciar o NTP com o seguinte comando:

```
$ service openntpd start
```

E sincronizar o servidor:

```
$ ntpdate -u pool.ntp.org
```

Sincronizamos o relógio do sistema, para quando reiniciarmos a máquina, não sair da sincronia:

```
$ hwclock –systohc
```

4.1.2.5 MTA

Todos os hosts que irão rodar o controlador de nuvem têm como opção rodar o *Mail Transport Agent (MTA)* na porta 25. O *Eucalyptus* usa o MTA pra transmitir e retransmitir mensagens para endereços de usuários da nuvem. Como não reflete no funcionamento da nuvem, essa configuração não foi implementada.

4.1.3 Instalação dos Componentes

Antes de instalar o *Eucalyptus* precisamos baixar e adicionar a chave de segurança para a lista de chaves confiáveis com os seguintes passos:

1 - Baixar a chave do endereço: http://www.eucalyptus.com/resources/security/keys

2 - Adicionar a chave pública: apt-key add c1240596-eucalyptus-release-key.pub

Agora precisamos criar um arquivo chamado eucalyptus.conf que terá variáveis que indicam as configurações que o usuário escolheu, esse arquivo fica no diretório /etc/apt/sources.list.d com o conteúdo a seguir:

```
deb    http://downloads.eucalyptus.com/software/eucalyptus/3.1/ubuntu
precise main
```

Em todas as máquinas que irão rodar o *Eucalyptus* ou o Euca2ools que é uma ferramenta de linha de comando feita para interagir com o AWS (*Amazon Web Services*), precisamos criar um arquivo chamado euca2ools.list no diretório /etc/apt/sources.list.d com o seguinte conteúdo:

```
deb http://downloads.eucalyptus.com/software/euca2ools/2.1/ubuntu
precise main
```

Agora atualizamos a máquina, para que a mesma reconheça os novos arquivos para instalar com o comando:

```
$ apt-get update
```

Por fim, instalamos o pacotes e dependências no servidor do E*ucalyptus*:

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

Em cada máquina que irá executar o *Node Controller* instalamos o pacote a seguir:

```
$ apt-get install eucalyptus-nc
```

4.1.3.1 Configuração do Eucalyptus

A partir da versão 1.5 o *Eucalyptus* inclui um subsistema de rede na VM altamente configurável que pode ser adaptado para variar de acordo com o ambiente. Existe quatro modos de alto nível de rede, cada um tem seus próprios parâmetros de configurações, recursos, benefícios em alguns casos, possuem restrições em sua configuração de rede local. O administrador precisa escolher um desses antes que comece a usar o *Eucalyptus* no *front-end* modificando o arquivo "eucalyptus.conf" que fica no diretório "/etc/eucalyptus" do Ubuntu. A seguir uma breve descrição dos modos que podemos escolher, e as configurações que precisam ser modificas no servidor CC (*Cloud Controller*):

- **VNET_MODE**

Está é a configuração que irá manipular como os IPs serão atribuídos às máquinas virtuais e como a rede irá se comportar. Existem quatro tipos que podem ser usados dependendo da necessidade do ambiente:

1 - **SYSTEM MODE**

Este é o modo mais simples em termos de configurações, o *Eucalyptus* praticamente fica fora do caminho da rede na VM. Nele o *Cloud Controller* irá gerar e atribuir um endereço MAC aleatoriamente às instâncias de VM paralelamente com a requisição do NC para levantar as instâncias. As máquinas virtuais obtêm endereço de IP diretamente do DHCP da rede que em estão implantadas.

2 - **STATIC MODE**

Neste modo o E*ucalyptus* irá gerenciar a atribuição de IPs para as VMs (máquina virtual). O administrador configura o *Eucalytus* com um "mapa" contendo um par contendo um endereço MAC e o endereço IP no Controlador de Nuvem.

3 - **MANAGED MODE**

Este modo é o mais rico oferecido pelo *Eucalytpus*. Nele o administrador define uma rede (geralmente privada e roteada) a partir da qual serão atribuídos IPs para as instâncias de VM. O CC irá montar um servidor DHCP com mapeamentos estáticos que levantará e alocará os IPs corretos no tempo em que os NCs requisitarem instâncias. (CSS Corp, 2010).

4 - **MANAGED-NOVLAN MODE**

Esse modo é semelhante ao anterior com a ausência do isolamento de rede para a VM, este modo pode ser usado por administradores que querem que os IPs das instâncias sejam atribuídos dinamicamente. Esse modo foi escolhido nesse projeto, pois, com ele a rede criada para o *Eucalytpus* estará interligada com a rede do ambiente, assim, teremos um maior controle das máquinas virtuais levantadas.

- **VNET_SUBNET**

Uma *subnet* é uma grupo lógico de dispositivos de rede conectados. Um nó em uma *subnet* tende a ser localizado nas proximidades físicas entre si em uma LAN. O E*ucalyptus* usa a *subnet* para atribuir IPs privados. Um exemplo de *subnet* que pode ser atribuída é: 192.168.0.0.

- **VNET_NETMASK**

Netmask é uma máscara de 32 bits usada para dividir um endereço de IP dentro de uma *subnet* e especifica a rede de hosts disponíveis.

- **VNET_DNS**

A tecnologia DNS (*Domain Name Server*) permite que hosts em uma rede TCP/IP sejam endereçados por nome, inclui um protocolo de rede e uma distribuição de dados para pesquisa de nome de hosts e endereços. Nessa configuração o usuário deve inserir o endereço do seu DNS.

- **VNET_ADDRSPERNET**

Usado para controlar quantas instâncias de uma VM podem ser simultaneamente parte de um usuário individual, chamado de grupo de segurança pela Amazon EC2. Por padrão, essa configuração é preenchida com o valor "32".

- **VNET_PUBLICIPS**

Aqui o usuário deve inserir a faixa de IPs públicos disponíveis na rede que serão atribuídos às instâncias de VM.

- **VNET_LOCALIP**

Endereço de IP local da interface onde o CC se comunica com o CLC.

- **VNET_DHCPUSER**

Caminho para o binário do DHCP, normalmente é: /usr/sbin/dhcpd3.

- **VNET_PRIVINTERFACE**

Interface que indica a rede privada onde os NCs irão ser executados. Para o modo Managed-NOVLAN, é recomendado usar uma interface de *bridge.* Exemplo: br0.

- **VNET_PUBINTERFACE**

Interface que indica a rede pública. Exemplo: eth0.

Agora que o servidor de nuvem foi configurado, iniciamos as configurações nas máquinas que irão executar os NCs (*Node Controllers*), e mudamos as seguintes opções no mesmo arquivo que editamos anteriormente:

- **VNET_MODE**

Aqui escolhemos qual será o modo que o E*ucalyptus* irá executar, inserimos o mesmo modo que foi editado no servidor que é o "MANAGED-NOVLAN".

- **VNET_BRIDGE**

Nome da *bridge* que está conectada à Internet.

Esse processo precisa ser repetido em todas as máquinas que irão executar o NC.

4.1.4 Iniciando o Eucalyptus

Para iniciar os serviços do CLC (*Cloud Controller*), Walrus e SC (*Storage Controller*), basta executar os seguintes comandos:

```
$ /usr/sbin/euca_conf --initialize
$ service eucalyptus-cloud start
```

Para iniciar os serviços do CC (*Cluster Controller*) executamos o seguinte comando:

```
$ service eucalyptus-cc start
```

E para iniciar os serviços do NC (*Node Controller*) é preciso executar o comando a seguir:

```
$ service eucalyptus-nc start
```

4.1.5 Gerando Credenciais

Para interagir com a nuvem é necessário ter credenciais que é uma forma de segurança na interação do usuário com o *Eucalyptus*.

Devemos executar os comandos a seguir para gerar as credenciais de administrador:

```
$ /usr/sbin/euca_conf --get-credentials admin.zip
$ unzip admin.zip
$ source eucarc
```

Após esse passo precisamos editar o "eucarc" que é um arquivo de configuração que define o valor correto para cada variável de ambiente associada ao *Eucalyptus*. Nesse arquivo é preciso modificar os seguintes parâmetros:

```
1 - EC2_URL : http://<host_ip>:8773/services/Eucalyptus
2 - S3_URL : http://<host_ip>:8773/services/Walrus
```

Onde "<host_ip>" é o IP atribuído a máquina.

4.1.6 Registrando os Componentes

Essa ferramenta implementa um protocolo de segurança para registrar os componentes separadamente para que todo o sistema não possa ser enganado incluindo um usuário ou administrador não autorizado. Só é preciso registrar os componentes uma vez, logo após o E*ucalyptus* ser iniciado depois que instalado. A seguir, são detalhados os parâmetros para registrar qualquer componente, exceto o NC:

- (--register-xyz) : O componente a ser registrado;
- (--partition) : A partição que o componente pertencerá;
- (--component) : O nome associado ao componente. Usado para identificar o componente de uma forma fácil para o humano;
- (--host) : O endereço IP do serviço a ser registrado.

Para registrar o componente NC é preciso apenas o nome do componente e o endereço IP como parâmetro.

A seguir é mostrado um exemplo de registros de componentes:

```
    $    /usr/sbin/euca_conf    --register-cloud    --partition
eucalyptus --host 10.10.2.35 --component clc-kawe
    $ /usr/sbin/euca_conf --register-walrus --partition walrus
--host 10.10.2.35 --component walrus-kawe
    $   /usr/sbin/euca_conf   --register-cluster   --partition
cluster01 --host 10.10.2.35 --component cc-kawe
    $ /usr/sbin/euca_conf --register-sc --partition cluster01 -
-host 10.10.2.35 --component sc-kawe
```

```
    $    /usr/sbin/euca_conf    --register-nodes    "10.10.2.33
10.10.2.34"
```

A seguir é detalhado como remover os registros dos componentes:

```
    $   /usr/sbin/euca_conf   --deregister-cloud   --partition
eucalyptus --host 10.10.2.35 --component clc-kawe
    $   /usr/sbin/euca_conf   --deregister-walrus   --partition
walrus --host 10.10.2.35 --component walrus-kawe
    $   /usr/sbin/euca_conf   --deregister-cluster   --partition
cluster01 --host 10.10.2.35 --component cc-kawe
    $ /usr/sbin/euca_conf --deregister-sc --partition cluster01
--host 10.10.2.35 --component sc-kawe
    $   /usr/sbin/euca_conf   --deregister-nodes   "10.10.2.33
10.10.2.34"
```

4.1.7 Configuração de Redirecionamento de IP

Para que a segurança do sistema melhore, é uma boa prática usar os serviços do *iptables* que possui estabilidade, rapidez, eficiência e fácil administração. O *iptables* é usado para configurar, manter e inspecionar os pacotes de filtragem de regras do IPv4 no *kernel* do Linux. (Ubuntu, 2010). Então, para usá-lo basta executar os comandos a seguir na máquina em que o *core* do *Eucalyptus* está sendo processado.

```
$ iptables --table nat --append POSTROUTING --out-interface
   eth0 -j MASQUERADE

$ iptables --append FORWARD --in-interface eth0 -j ACCEPT
```

4.1.8 Verificação de Inicialização

O Eucalyptus permite verificar se todos os seus serviços estão sendo processados corretamente, para analisar com detalhes é só ler os arquivos de logs que ficam no diretório /var/log/eucalyptus. A seguir, é detalhado em quais portas os serviços são executados:

- A "escuta" do CLC (*Cloud Controller*) é feita nas porta 8443 e na 8773;
- O Walrus usa a porta 8773;
- O SC (*Storage Controller*) está na porta 8773;

- O CC (*Cluster Controller*) usa a porta 8774;
- O NC (*Node Controller*) é "escutado" na porta 8775;

Para saber o estado dos serviços basta executar esse comando:

```
$ euca-describe-services
```

A saída deve ser semelhante a essa:

```
SERVICE storage cluster01 sc-kawe ENABLED 19
http://150.165.85.80:8773/services/Storage arn:euca:eucalyptus:cluster01:storage:sc-
kawe/
SERVICE walrus walrus walrus-kawe ENABLED 19
http://150.165.85.80:8773/services/Walrus arn:euca:bootstrap:walrus:walrus:walrus-
kawe/
SERVICE eucalyptus eucalyptus 150.165.85.80 ENABLED 19
http://150.165.85.80:8773/services/Eucalyptus
arn:euca:eucalyptus:::150.165.85.80/
SERVICE bootstrap bootstrap 150.165.85.80 ENABLED 19
http://150.165.85.80:8773/services/Empyrean
arn:euca:bootstrap:::150.165.85.80/
SERVICE dns eucalyptus 150.165.85.80 ENABLED 19
http://150.165.85.80:8773/services/Dns arn:euca:eucalyptus::dns:150.165.85.80/
SERVICE cluster cluster01 cc-kawe ENABLED 19
http://150.165.85.80:8774/axis2/services/EucalyptusCCarn:euc
a:eucalyptus:cluster01:cluster:cc-kawe/
```

Como podemos ver acima, todos os serviços estão com o status "ENABLED", isso significa que estão executando normalmente. O comando a seguir mostra um relatório sobre o registro de um nó:

```
$ euca-describe-availability-zones
```

A saída para esse comando nos informa quantas VMS podemos instanciar, isso depende das configuração da máquina em que o NC (*Node Controller*) está executando. As informações para este comando é semelhante ao exemplo abaixo:

```
AVAILABILITYZONE cluster01 150.165.85.80
arn:euca:eucalyptus:cluster01:cluster:cc-kawe/
```

```
AVAILABILITYZONE      |- vm types     free / max   cpu   ram  disk
AVAILABILITYZONE      |- m1.small    0004 / 0004   1    128    2
AVAILABILITYZONE      |- c1.medium   0004 / 0004   1    256    5
AVAILABILITYZONE      |- m1.large    0002 / 0002   2    512   10
AVAILABILITYZONE      |- m1.xlarge   0002 / 0002   2   1024   20
AVAILABILITYZONE      |- c1.xlarge   0000 / 0000   4   2048   20
```

Como podemos ver na saída do comando acima, a máquina tem a possibilidade de instanciar 4 VMs do tipo "m1.small", 4 "c1.medim", 2 "m1.large" e 2 do tipo "m1.xlarge". O *Eucalyptus* usa como referência, os tipos de instâncias disponibilizadas na Amazon.

4.1.9 Gerenciamento de Imagens

Uma imagem é um retrato (estado atual) do sistema de arquivo raiz, servindo de base para as instâncias. Quando uma máquina virtual é iniciada, o usuário escolhe uma imagem da máquina que usará como modelo. Uma nova máquina virtual é então uma instância de uma imagem que contém suas próprias cópias de tudo. As instâncias continuam executando até o usuário terminá-la ou pará-la. Se uma instância falhar, o usuário pode lançar uma nova pela mesma imagem. Ainda pode criar múltiplas instâncias de uma única imagem de máquina, e cada instância será independente da outra. Um tipo de instância define o que ela possui de recursos de *hardware*, incluindo a quantidade de memória, espaço de disco e CPU alocada. Imagens são apoiadas por armazenamento persistente ou por armazenamento da instância, se a imagem usar a persistência, significa que o dispositivo raiz é um retrato e aparece como um volume persistente quando a instância é lançada de uma imagem de máquina. Uma máquina que é apoiada por um armazenamento de instância significa que o dispositivo raiz é armazenado no *Walrus.* Geralmente quando usamos o termo "imagem" quer dizer que é um arquivo de sistema raiz, que foi empacotado, transferido para o servidor e registrado com o *Eucalyptus*, tal imagem é conhecida como EMI (*Eucalyptus Machine Image).* No entanto, existe outros tipos de imagens que servem como suporte para a EMI. Elas podem ser do kernel (EKI - *Eucalyptus Kernel Image*) ou do *ramdisk* (ERI - *Eucalyptus* Ramdisk Image). Essas contêm módulos do *kernel* necessários para propor funcionalidades para a imagem. Um conjunto de ERIs e EKIs são usadas por vários EMIs. Uma vez carregada dentro de uma nuvem *Eucalyptus*, A ERI e EKI são referidas por uma imagem e o usuário não tem interação com elas diretamente.

4.1.9.1 Adicionando Imagens

Umas das maneira de inserir uma imagem é adicionando uma imagem baseada em outra. Se o usuário já tem uma imagem alocada ele pode criar outra com as mesmas características da antiga. O *Eucalyptus* tem imagens disponíveis para começar da correta, você pode encontrar links na página principal da ferramenta, e ainda pode obter imagens prontas da página de imagens máquinas do *Eucalytus.* Para ativar uma imagem como uma entidade executável é preciso fazer os seguintes passos:

- Empacotar uma imagem de disco raiz e o par *kernel/ramdisk;*
- Enviar o pacote da imagem para o armazenamento do *Walrus;*
- Registrar os dados com o *Eucalyptus;*

Para fazer isso na prática, primeiro é necessário baixar a imagem do link http://emis.eucalyptus.com/ e descompactar o arquivo *tar* e depois executar os seguintes comandos em um terminal do Linux:

- euca-bundle-image -i <kernel_file> --kernel true;
- euca-upload-bundle -b <kernel_bucket> -m /tmp/<kernel_file>.manifest.xml;
- euca-register <kernel_bucket>/<kernel_file>.manifest.xml
- euca-bundle-image -i <ramdisk_file> --ramdisk true;
- euca-upload-bundle -b <kernel_bucket>/<kernel_file>.manifest.xml;

4.1.9.2 Verificando uma Imagem

Para exibir um relatório de imagens disponíveis na nuvem do *Eucalyptus*, basta executar o seguinte comando:

```
$ euca-describe-images
```

Se aparecer uma lista semelhante com a que segue a abaixo, a imagem foi instalada com sucesso:

IMAGE emi-B3C63807 02_rf_bucket/euca-debian-2011.07.02-x86_64.img.manifest.xml 996156983939 available public i386 machine eki-58653BBB eri-A4E03BD1 instance-store
IMAGE eri-A4E03BD1 02_rd_bucket/initrd.img-2.6.28-11-generic.manifest.xml 000000000001 available public i386 ramdisk instance-store
IMAGE eki-58653BBB 02_kernel_bucket/vmlinuz-2.6.28-11-generic.manifest.xml 000000000001 available public i386 kernel

Note que o ID da imagem é emi-B3C63807.

4.1.10 Usando Instâncias

Uma instância é uma máquina virtual. Uma VM é essencialmente operações de um computador que contém processos do sistema, aplicações, acessibilidade à rede e *drivers* de discos. Através do *Eucalyptus* é possível executar instâncias baseadas no Linux ou Windows.

4.1.10.1 Criando Par de Chaves

O *Eucalyptus* usa criptografia de pares de chaves para acessar as instâncias. Antes de executar uma instância, é necessário criar um par de chaves. Após criar um par de chaves é gerado duas chaves: a pública que é salva dentro do *Eucalyptus* e a chave privada que é um conjunto de caracteres gerado para o usuário. (*Eucalyptus, 2013*). Para isso, é preciso executar os comandos a seguir:

```
$ euca-add-keypair<keypair_name> > <keypair_name>.private
```

Onde <keypair_name> é um nome único para seu par de chaves, como no exemplo abaixo:

```
$ euca-add-keypair kawe-keypair > kawe-keypair.private
```

A chave privada é salva no em um arquivo no diretório local. O próximo passo é mudar a permissão para acessar o arquivo de chave criado com o seguinte comando:

```
$ chmod 0600 kawe-keypair.private
```

Consulte o sistema para ver a chave pública:

```
$ euca-describe-keypairs
```

Esse comando retorna uma saída semelhante a essa:

KEYPAIR kawe-keypair
ad:0d:fc:6a:00:a7:e7:b2:bc:67:8e:31:12:22:c1:8a:77:8c:f9:c4

4.1.10.2 Executando uma Instância

Para iniciar uma instância, só é preciso do ID da imagem que está executando e do nome do par de chaves criado pelo usuário como o exemplo a seguir:

```
$ euca-run-instances emi-EC1410C1 –k kawe-keypair
```

Com o comando a seguir é possível verificar o status da instância criada:

```
$ euca-describe-instances
```

A saída comum para esse comando é:

RESERVATION r-460007BE kawe kawe-default
INSTANCE i-2F930625 emi-EC1410C1 192.168.10.187 10.1.0.107
running kawe-keypair
2010-03-29T23:08:45.962Z eki-822C1344 eri-BFA91429

4.1.10.3 Acessando uma Instância

Para usar uma instância o usuário deve entrar via ssh com um IP atribuído a ela. Você pode ter acesso ao IP usando o comando "euca-describe-instances". Para logar em uma instância basta usar o IP da mesma e a chave privada, como segue o exemplo a seguir:

```
$ ssh -i kawe-keypair.private root@192.168.7.30
```

4.1.10.4 Reiniciando uma Instância

Reiniciar uma instância preserva o sistema de arquivo raiz de provocar falhas. Segue a seguir um exemplo de como fazer isso:

```
$ euca-reboot-instance <instance_id>
```

Onde <instance_id> é o ID atribuído a instância quando criada.

4.2 OPENSTACK

Aqui é descrito como instalar e configurar o O*penStack* no Ubuntu 12.04 LTS, com um controlador de *Cluster* e um Nó cliente. Veremos os requisitos, instalação e configurações que para o funcionamento dessa ferramenta. Como a comunidade do *OpenStack* é bastante movimentada, têm vários *post* e dúvidas cadastradas, ajudando os usuários que iniciam o seu uso.

4.2.1 Pré-Requisitos

Para instalação do *OpenStack* é recomendado os requisitos a seguir.

4.2.2 Configurações de Rede

O *Openstack* precisa ser instalado em sua própria sub-rede sem DHCP. Cada nó é configurado com um endereço IP, máscara de sub-rede e *Gateway*. Para isso é preciso editar o arquivo /etc/network/interfaces como no exemplo a seguir:

```
auto eth0
iface eth0 inet static
     address 192.160.42.11
     netmask 255.255.255.0
     gateway 192.168.42.1
```

4.2.3 Instalação

O *OpenStack* requer um usuário com acesso *root* para executar. Todos os nós devem rodar com o mesmo nome de usuário. Escolhemos *stack* como nome de usuário nesse exemplo, que é o usuário padrão usado no *OpenStack*. Para fazer isso, segue os comandos:

```
$ groupadd stack
$ useradd -g stack -s /bin/bash -d /opt/stack -m stack
```

Esse usuário fará várias mudanças no sistema, que precisam ter privilégios de administrador (*root)* sem senha. Portanto, adicionamos o usuário *stack* na lista de usuários com permissão de administrador no sistema:

```
$ echo "stack ALL=(ALL) NOPASSWD: ALL" >> /etc/sudoers
```

Daqui em diante, faremos todos os processos logado com o usuário "stack". Para mudar de usuário, executamos o comando:

```
$ su stack
```

Para instalar a nuvem do *OpenStack,* usamos o procedimento do *DevStack* que é um *shell script* criado pela *Rackspace Cloud Builders* e mantido pela comunidade de desenvolvimento do *OpenStack,* documentado para construir ambientes completos de desenvolvimento da nuvem.

O próximo passo é baixar o projeto do *DevStack* que está hospedado no *GitHub:*

```
$ git clone git://github.com/openstack-dev/devstack.git
$ cd devstack
```

Agora configuramos o usuário para acesso à chave ssh:

```
$ mkdir ~/.ssh
$ chmod 700 ~/.ssh
$ cd ~/.ssh
```

```
$ ssh-keygen -t rsa > authorized_keys
```

Até esse ponto, todos esses comandos descritos acima são em comum em todos os nós da rede. A partir daqui, existem diferenças entre o controlador de *cluster* e os nós clientes.

4.2.3.1 Configuração do Cluster Controller

Esse controlador executará todos os serviços do *OpenStack*. Agora configuramos as variáveis de rede no arquivo "localrc" dentro da pasta "devstack" criada quando o download foi realizado:

```
HOST_IP=192.168.42.11
FLAT_INTERFACE=eth0
FIXED_RANGE=10.4.128.0/20
FIXED_NETWORK_SIZE=4096
FLOATING_RANGE=192.168.42.128/25
MULTI_HOST=1
LOGFILE=/opt/stack/logs/stack.sh.log
ADMIN_PASSWORD=labstack
MYSQL_PASSWORD=supersecret
RABBIT_PASSWORD=supersecrete
SERVICE_PASSWORD=supersecrete
SERVICE_TOKEN=xyzpdqlazydog
```

Nesse tipo de configuração os 10 primeiros IPs são reservados como privados para a sub-rede. Para isso basta criar o arquivo "local.sh" para executar o seguinte comando:

```
for i in `seq 2 10`; do /opt/stack/nova/bin/nova-manage fixed
reserve 10.4.128.$i; done
```

Para levantar o *OpenStack* e configurar todas as suas dependências é preciso rodar o seguinte *script* que fica na pasta onde foi instalado o *DevStack*:

```
$ ./stack.sh
```

Esse *script* instala e configura várias combinações de componentes e serviços como: *Ceilometer, Cinder, Glance, Heat, Horizon, Keystone, Nova, Neutro* e *Swift.* É totalmente personalizado, definindo as variáveis de ambiente (DevStack, 2013). O usuário pode mudar uma configuração padrão fazendo uma exportação como no exemplo:

```
$ export DATABASE_PASSWORD=anothersecret
$ ./stack.sh
```

Pode também mudar o caminho onde será instalado:

```
$ DEST=${DEST:-/opt/stack}
```

Uma grande quantidade de processos serão executados na instalação. Para conferir com detalhes, é só abrir o arquivo de log que fica em "/opt/log/stack/stack.sh.log".

Agora configuramos todos os nós da rede que serão clientes do *Cluster Controller*. Para isso, editamos o arquivo "locarc" com o seguinte conteúdo:

```
HOST_IP=192.168.42.12
FLAT_INTERFACE=eth0
FIXED_RANGE=10.4.128.0/20
FIXED_NETWORK_SIZE=4096
FLOATING_RANGE=192.168.42.128/25
MULTI_HOST=1
LOGFILE=/opt/stack/logs/stack.sh.log
ADMIN_PASSWORD=labstack
MYSQL_PASSWORD=supersecret
RABBIT_PASSWORD=supersecrete
SERVICE_PASSWORD=supersecrete
SERVICE_TOKEN=xyzpdqlazydog
DATABASE_TYPE=mysql
SERVICE_HOST=192.168.42.11
MYSQL_HOST=192.168.42.11
RABBIT_HOST=192.168.42.11
GLANCE_HOSTPORT=192.168.42.11:9292
ENABLED_SERVICES=n-cpu,n-net,n-api,c-sch,c-api,c-vol
```

A variável "HOST_IP" muda para cada nó da rede. O comando a seguir serve para criar um usuário do *Openstack:*

```
export OS_TENANT_NAME=admin
export OS_USERNAME=admin
export OS_PASSWORD=admin_pass
export OS_AUTH_URL=<http://ip:5000/v2.0>
```

Onde <ip:5000> é o IP atribuído a máquina usando a porta 5000.

4.2.3.2 Gerando Credenciais

O *OpenStack* tem a funcionalidade de credencial da mesma forma que o *Eucalyptus* têm, para gerá-la basta executar o comando a seguir:

```
$ . openrc admin admin
```

4.2.3.3 Lista de Imagens Disponíveis

Essa ferramentas nos fornece uma lista de imagens que podem ser usadas para criar as instâncias, é possível ver essa lista com o comando a seguir:

```
$ nova image-list
```

Segue um exemplo de saída para esse comando:

ID	Name	Status
c225e762-ebec-4be4-aaa9-0d05435e0c9e	cirros-0.3.1-x86_64-uec	ACTIVE
2527bc42-a379-457c-8d6d-bc759e67b0c4	cirros-0.3.1-x86_64-uec-kernel	ACTIVE
75a80b17-54e5-4ec3-92f5-ffb57c1968f5	cirros-0.3.1-x86_64-uec-ramdisk	ACTIVE

Tabela 2 - Lista de Imagens do OpenStack

4.2.3.4 Tipos de Máquinas Virtuais

Antes de gerar uma instância, devemos escolher que tipo queremos, considerando a quantidade de recursos para serem usados, para isso executamos o seguinte comando:

```
$ nova flavor-list
```

Para saída desse comando, segue uma tabela de exemplo:

ID	Name	Memory_MB	Disk	Ephermeral	VCPUs	RXTX_Factor	Is_Public
1	m1.tiny	512	1	0	1	1.0	True
2	m1.small	2048	20	0	1	1.0	True
3	m1.medium	4096	40	0	2	1.0	True
4	m1.large	8192	80	0	4	1.0	True
42	m1.nano	64	0	0	1	1.0	True
5	m1.xlarge	16384	160	0	8	1.0	True
84	m1.micro	128	0	0	1	1.0	True

Tabela 3 - Tipos de VMs do OpenStack

4.2.3.5 Iniciando uma Instância

Para iniciar uma instância, é preciso escolher a imagem, tipo de VM e um nome para ela usando as informações mostradas anteriormente, o comando a seguir cria uma instância de exemplo:

```
$ nova boot --flavor=3 --image=2527bc42-a379-457c-8d6d-bc759e67b0c4 testserver
```

Segue abaixo a tabela referente ao comando:

Property	Value
OS-EXT-STS:task_state	scheduling
image	cirros-0.3.1-x86_64-uec-kernel
OS-EXT-STS:vm_state	building
OS-EXT-SRV-ATTR:instance_name	instance-00000001
OS-SRV-USG:launched_at	None
flavor	m1.medium
Id	e0a24d0f-c541-42e7-9fd6-17e17fc96a7f
security_groups	[{u'name': u'default'}]
user_id	fa3b1b802a294aa9830b957c84cb715a

OS-DCF:diskConfig	MANUAL
accessIPv4	
accessIPv6	
progress	0
OS-EXT-STS:power_state	0
OS-EXT-AZ:availability_zone	Nova
config_drive	
Status	BUILD
updated	2013-09-22T01:36:59Z
hostId	
OS-EXT-SRV-ATTR:host	None
OS-SRV-USG:terminated_at	None
key_name	None
OS-EXT-SRV-ATTR:hypervisor_hostname	None
Name	testserver
adminPass	n3xEWPQByN4Q
tenant_id	0979fadd8c604b97afc59efac56832fd
created	2013-09-22T01:36:58Z
os-extended-volumes:volumes_attached	[]
metadata	{}

Tabela 4 - Criação de uma instância no OpenStack

5 ANÁLISE DE VIABILIDADE

Neste capítulo é relatado como foi feita a decisão de escolher tais ferramentas para instalação da nuvem, os problemas encontrados, assim como as lições aprendidas no decorrer do trabalho, recomendações sobre o uso e a análise da viabilidade de instalação da nuvem em um ambiente de rede em um laboratório de computação de uma universidade.

5.1 COMPARAÇÃO DE RECURSOS DAS FERRAMENTAS

Aqui falaremos um pouco sobre as principais diferenças entre as duas ferramentas escolhidas para implementação da nuvem. Com base nas suas comunidades, componentes, instalações, configurações e outros fatores com o qual possamos diferenciá-las. De acordo com a mais nova pesquisa da IDC (*International Data Corporation*), até 2016 os gastos com hospedagem de nuvem privada no mundo, será de $24 bilhões. Assim, podemos ter uma noção de como esse mercado está crescendo rápido, tornando necessário saber qual a melhor ferramenta para implementar essa tecnologia.

O primeiro aspecto que podemos usar é a maturidade no mercado. O *OpenStack* baseia-se nas parcerias com a AT & T, HP e IBM, que são grandes empresas que se comprometeram em usá-lo como fundamento em suas ofertas de nuvem privada. Já o *Eucalyptus*, se confia em seus laços com a AWS (*Amazon Web Services*), tentando convencer as empresas a escolher o uso de nuvens híbridas.

Daniel Kranowski, consultor da LLC (companhia que faz algoritmos de negócios), falou na conferência do *JavaOne*, que a arquitetura do *Eucalyptus* é similar ao AWS, tem um nível médio de instalação, e uma interface administrativa limitada, que precisa um pouco da ajuda da linha de comando. Também falou que o *Eucalyptus* tem um modelo de segurança de chaves dividido nos cincos componentes, e que cada um precisa se registrar com os outros. Ele comentou sobre o *OpenStack*, dizendo que o mesmo possui uma arquitetura distribuída e fragmentada, de difícil instalação, e que fica impulsivo por causa dos vários CLIs (*Command-Line Interface*), tem um sistema de segurança forte baseado em *tokens* e no *Swift*, e diz que o sistema de armazenamento de objetos é o pivô da sua alta disponibilidade.

Na tabela a seguir, podemos acompanhar as contribuições de *commits* feitos por instituições parceiras desses dois projetos no período de 2009 à 2013. Está análise foi feita pela comunidade CY13-Q1 (artigo publicado com o título de "*Open Source IaaS Community Analysis*"). Podemos observar que o *Eucalyptus* é praticamente dominado por si próprio, tendo poucas contribuições de outras empresas. Já o *OpenStack*, recebe várias contribuições de grandes empresas.

Eucalyptus		OpenStack	
Domínio	%	Domínio	%
eucalyptus.com	85	review.openstack.org	48
gmail.com	13	redhat.com	9
fedoraproject.com	2	gmail.com	9
		linux.vnet.ibm.com	4
		openstack.org	3
		us.ibm.com	3
		hp.com	2
		nicira.com	2
		cloudscaling.com	1
		intel.com	1

Tabela 5 - Contribuições de Instituições (2009 -2013)

Outra comparação, o conjunto de dados de referência, segue na tabela 6.

	Eucalyptus	OpenStack
Mensagens Mensais	1064	2697
Taxa de Participação	3.63	3.15
População Trimestral	170	679
População Mensal	150	541

Tabela 6 - Conjunto de dados de referência

Como foi descrito antes, a comunidade do *OpenStack* é maior e mais movimentada em relação à do *Eucalyptus*. Com essas informações percebemos que apenas na taxa de participação o *Eucalyptus* ganha, deixando uma vantagem a mais para o *OpenStack.* As mensagens mensais representam o volume de discussões ocorridos, a taxa de participação, representa a média de respostas para uma pergunta, a população ativa do trimestre, representa a possibilidade de obter ajuda da comunidade a longo prazo, enquanto a população ativa mensal, representa a possibilidade de obter ajuda da comunidade a curto prazo.

Em relação à persistência de objetos, as duas ferramentas o fazem armazenamento imagens de disco, mas em relação à tolerância a falhas apenas na versão 3.0 do *Eucalyptus* foi inseriu o uso de DRBD (*Distributed Replicated Block Device*), que é um módulo inserido no núcle do Linux que disponibiliza armazenamento distribuído, utilizado em *clusters* de alta disponibilidade. Já o *OpenStack*, usa sincronização do *backend* (Código base da aplicação, que controla os componentes do sistema). Comparando a criação de imagens de máquinas virtuais, as duas possuem a funcionalidade do usuário criar e gerenciar suas próprias imagens de VMs, o *OpenStack* possui pouco suporte à API da Amazon, enquanto o *Eucalyptus* possui bastante comunicação com esse serviço. Sobre a interface, o *Eucalyptus* não controla instâncias de convidados, apenas pela interface da linha de comando. Enquanto o *OpenStack* disponibiliza mais recursos na interface direta com o usuário.

A decisão de uso dessas ferramentas foi feita após uma pesquisa sobre as melhores ferramentas para construção de uma nuvem computacional. Elas são as mais famosas no mercado e na comunidade acadêmica. Pois, possuem muitas funcionalidades que facilitam as tarefas do administrador e usuário. Além de ter componentes bem implementados e gerenciáveis, são flexíveis.

5.2 COMPARAÇÃO DO PROCESSO DE INSTALAÇÃO

Como vimos na sessão de instalação e configuração das ferramentas, para instalar o *Eucalyptus* é necessário um esforço maior do que instalar o *OpenStack*. Mesmo com a semelhança de alguns serviços nas duas ferramentas, o *OpenStack* é mais fácil de instalar e configurar, pois algumas funcionalidades são disponibilizadas na interface web.

5.3 DESEMPENHO DA NUVEM NOS LABORATÓRIOS

Nesta seção faremos uma estimativa de qual seria o tamanho da nuvem criada no laboratório de informática do Departamento de Ciências Exatas da UFPB – Campus IV. Os dois principais laboratórios do departamento possuem 80 máquinas, com as mesmas configurações, que são detalhadas a seguir:

- CPU: AMD Phenom 9600B Quad-Core, 1.15 GHz;
- Memória: 2 GB RAM;

Com base nessas informações, o *Eucalyptus* verifica a quantidade e os tipos de instância de VMs que a máquina pode iniciar. Para isso, utilizamos o comando "euca-describe-availability-zones", que mostra a seguinte saída para um Node Controller instalado em uma das máquinas do laboratório:

Tipo de VM	Capacidade	CPU (cores)	RAM (MB)	Disco (GB)
m1.small	4	1	128	2
c1.medium	4	1	256	5
m1.large	2	2	512	10
m1.xlarge	1	2	1024	20

Tabela 7 - Tipos de VMs disponíveis

O tipo de máquina virtual define a quantidade de recursos alocados para uma instância como, o número de CPUs, tamanho da memória e o tamanho de armazenamento, podemos observar essas informações na tabela acima. No *Eucalyptus* existem cinco tipos de VMs pré-definidos, juntando todas as máquinas, podemos calcular o seguinte:

Tipo de VM	Capacidade (VMs)	Total de CPU (cores)	Total de RAM (MB)	Total de Disco (GB)
m1.small	320	320	40.960	640
c1.medium	320	320	81.920	1.600
m1.large	160	160	81.920	1.600
m1.xlarge	80	160	81.920	1.600

Tabela 8 - Total de VMs para todas as máquinas

Na tabela 8, podemos ter uma estimativa do tamanho da nuvem usando todas as máquinas disponíveis nos laboratórios, para diferentes tipos de instâncias. Por exemplo, o tipo de VM *m1.small* possui 128 MB de RAM, então se cada máquina tem a capacidade de executar 4 instâncias e sabemos que há 80 máquinas nos laboratórios, então teremos um total de 40.960 MB de RAM para serem usados pelas 320 instâncias.

5.3.1 Economia no uso da Nuvem

Com o uso dessa nuvem, teríamos vários gastos como energia, manutenção, funcionários de operação e problemas de redes que podem surgir com o alto processamento de carga feito pela nuvem. Aqui faremos uma análise de como seriam as despesas se a UFPB pagasse por uma nuvem pública de capacidade semelhante à da suposta nuvem estimada anteriormente. Escolhemos os serviços da *Amazon EC2* pelo seu poder no mercado para fazer a comparação. Na tabela a seguir são detalhados os tipos de instâncias que podemos alugar:

Tipo de VM	**CPU (cores)**	**RAM (GB)**	**Disco (GB)**	**Preço**
t1.micro	1	0,615	Somente EBS	$0.020
m1.small	1	1,7	1 x 160	$0.060
m1.medium	1	3,75	1 x 410	$0.120
m1.large	2	7,5	2 x 420	$0.240
c1.xlarge	8	7	4 x 420	$0.480

Tabela 9 - Tipos de Instâncias da Amazon EC2

Na tabela 9, é descrito os tipos VMs disponíveis para alugar. A instância *m1.large* tem capacidade para executar dois *cores* de CPU. Portanto, cada *core* usa 420 GB de armazenamento. O Amazon EBS (*Elastic Block Store*) fornece volumes de armazenamento em bloco para uso com instâncias do Amazon EC2. Os volumes do Amazon EBS são conectados à rede e persistem independentemente da vida útil de uma instância. (Amazon, 2013).

Para estimar o custo de se usar uma nuvem pública com capacidade equivalente à nuvem privada oportunista da universidade, nós identificamos quais os tipos de instância do Eucalyptus e da Amazon possuem configurações mais próximas, para possibilitar uma comparação de custo justa entre as nuvens diferentes. Observamos que a instância do tipo

medium do Eucalyptus se aproxima da instância do tipo *micro* da Amazon EC2, então usamos esse tipo de instância como base para a comparação. Na Tabela 10 apresentamos o custo para manter ativa uma instância *micro* por hora, dia, mês e ano.

Tempo	**Preço**
Hora	$0.020
Dia	$0.48
Mês	$14.4
Ano	$172.8

Tabela 10 - Preço para aluguel da instância micro na Amazon EC2

Considerando o tipo de instância *medium* no Eucalyptus como equivalente à *micro* na Amazon EC2, e que em cada máquina das 80 máquinas da universidade é possível levantar 4 instâncias do tipo *medium*, como descrito anteriormente, podemos estimar o valor estimado da nuvem privada da universidade, com base no custo equivalente na Amazon, como mostra a Tabela 11.

Tempo	**Preço (320 instâncias)**
Hora	$6.4
Dia	$153.6
Mês	$4.608
Ano	$55.296

Tabela 11 - Preço para aluguel das 320 instâncias

Sabemos que a nuvem não poderia estar em funcionamento o dia todo, pois outras atividades acadêmicas como construção de trabalhos e testes de sistemas que são feitas nos laboratórios devem ser priorizadas. Em um caso médio, as máquinas estariam disponíveis para nuvem durante 16 horas por dia, pois na maioria das vezes os alunos usam as máquinas das 8 às 12 horas e de 13 às 17 horas. Com base nessas informações de disponibilidade das máquinas para a nuvem por dia, as despesas ficariam da seguinte maneira:

Tempo (16 horas)	Preço (320 instâncias)
Dia	$102.4
Mês	$3.072
Ano	$36.864

Tabela 12 - Preço para aluguel das 320 instâncias (16 horas por dia)

Com base nesses dados, podemos concluir que há uma grande vantagem em se construir uma nuvem do início. Pois, além da UFPB economizar muito dinheiro, dará a oportunidade para a comunidade acadêmica obter um conhecimento muito rico, que se está precisando cada dia mais no mercado de computação. Outro motivo é a possibilidade dos alunos aprender como é o funcionamento da nuvem na prática, além do uso dela para realização de seus trabalhos, testes de sistemas e outros.

5.4 PROBLEMAS ENCONTRADOS

Nesta seção são descritas as dificuldades encontradas no decorrer da instalação e configuração da nuvem computacional, para servir de referência em futuras implantações desses sistemas. Tendo em vista que as ferramentas usadas são recentes, a quantidade de referências, livros, artigos e tutoriais diminuem, dificultando o conhecimento para aqueles que estão se familiarizando com este tipo de tecnologia. Para instalar essas ferramentas, é preciso uma grande quantidade de outros aplicativos e serviços para um bom funcionamento e, para isso, requer um estudo sobre tais ferramentas, aplicativos e serviços que devem ser instalados corretamente para que a nuvem funcione corretamente. Outro ponto importante, que vale entrar em questão, é que essas ferramentas foram feitas para serem implantadas em um ambiente de rede dedicado apenas ao funcionamento da nuvem, já que no caso desse trabalho, o ambiente já tem uma rede comum funcionando no laboratório. Assim, ficou mais complicado ainda para diferenciar a nuvem da rede implantada. Esse ambiente que queremos implantar a nuvem, não é muito comum para usuários dessas ferramentas, então fica difícil achar material em fóruns e documentações sobre como solucionar e adaptar a rede para que ela funcione na rede que está implantada atualmente, e na rede da nuvem, já que a documentação assume que o usuário usará a rede apenas para funcionamento da nuvem.

5.4.1 Conexão com a Internet

A UFPB Campus IV têm carência de uma boa conectividade com a internet, alunos e professores sempre reclamam da qualidade da conexão que temos nos laboratórios de informática do campus. Por causa da internet lenta no laboratório ou indisponível, muitas vezes não era possível instalar as ferramentas necessárias, o que dificultava muito a consulta a fóruns e documentações, além da demora para baixar os serviços necessários.

5.4.2 Interface de Rede

Na documentação do *Eucalyptus* é recomendado que a interface de rede do computador que irá executar o servidor da nuvem esteja com o endereço de IP estático e que tenha uma interface pública para atribuição de IPs para as máquinas virtuais. Na sessão 4.2.3 é detalhado as configurações que devem ser feitas na *bridge*. Fizemos a alteração do arquivo /etc/network/interfaces, mas quando a máquina é iniciada, a mesma não tem acesso a internet, então a solução foi deixar o arquivo com as configurações padrões, pois isso não interfere no funcionamento do *Eucalyptus.*

5.4.3 DHCP

O *Eucalyptus* tem seu próprio DHCP para atribuir endereço de IP para as instâncias que são criadas pelo usuário, então se recomenda que a rede em que as máquinas estão interligadas não possua nenhum DHCP ativo. Como a rede da Universidade também possui um servidor de DHCP, os dois entram em conflito no momento de criar uma instância. Com isso, o *Eucalyptus* não consegue atribuir um IP corretamente às instâncias criadas. As VMs iniciadas pelo Eucalyptus possuem endereço de MAC começando por *d0:0d.* Isso significa que podemos configurar o servidor DHCP da rede para ignorar as requisições vindas das VMs do Eucalyptus, filtrando pelos endereços de MAC que possuem este padrão. Para isto, é preciso adicionar a seguinte configuração no *dnsmasq*:

```
dhcp-host=d0:0d:*:*:*:*,ignore
```

O dnsmasq é um servidor leve designado para prover serviços de DNS e DHCP para pequenas redes. Ele pode atribuir nomes a máquinas locais que não estão no DNS global.

No *dhcp3-server* é necessário configurar para definir uma classe correspondente ao prefixo do endereço MAC, e então excluir a classe da *pool*. Geralmente o arquivo do servidor DHCP fica nesse diretório: /etc/dhcpd.conf, a configuração fica dessa forma:

```
class "ignoreclass"{
     match if (substring(hardware,1,2)=d0:0d);
}

subnet 192.168.0.0 netmask 255.255.255.0 {
     option domain-name-servers 192.168.0.1;
     pool{
          deny members of "ignoreclass";
          range 192.168.0.101 192.168.0.199;
     }
}
```

A classe "ignoreclass" foi criada para ignorar pedidos de instâncias. Logo após na linha "deny members of ignoreclass" é onde os endereços atribuídos são negados. Finalmente, na linha "range 192.168.0.101 192.168.0.199" é onde todos os outros endereços de MAC receberão seus endereços.

Como não temos permissão para alterar as configurações da rede dos laboratórios. Para fazer essa modificação no servidor DHCP da rede, o administrador da rede foi contactado e o mesmo fez as alterações descritas acima.

5.4.4 Virtualização na BIOS

O E*ucalyptus* precisa usar a tecnologia de virtualização para criar as instâncias de máquinas virtuais. Nos computadores da rede da universidade a opção de virtualização estava desativada na BIOS, então foi solicitado a permissão do administrador para habilitar o uso dessa ferramenta. Entrando no menu de configuração da BIOS, na seção de Segurança, existe a opção de suporte do hardware à virtualização, que deve ser ativada.

5.4.5 Servidor MySQL

Na instalação do *OpenStack* ocorreu uma falha ao tentar iniciar o servidor MySQL. Para resolver esse problema foi preciso reinstalá-lo com o seguinte comando:

```
$ apt-get --reinstall install mysql-server-5.5
```

5.5 DISCUSSÃO SOBRE A VIABILIDADE

Para construir um ambiente como foi proposto nesse trabalho é preciso uma equipe de desenvolvimento com bastante experiência em Linux, computação em nuvem, redes e arquitetura de computadores. Um bom conhecimento do ambiente de rede onde será instalada a nuvem é também uma vantagem. Notou-se que as ferramentas geralmente são adequadas para serem instaladas em ambientes que não estão previamente implantados, dando total liberdade para que as configurações do ambiente, principalmente da rede, seja completamente moldado para os requisitos das aplicações de gerência da nuvem. Foi solicitada ajuda de equipe com essas características, mas o problema não foi solucionado.

5.6 APRENDIZADO COM O PROJETO

Em todo o decorrer do trabalho foi necessário estudar vários conceitos, documentações, livros, tutoriais, dúvidas de grupos e etc. Onde surgiu a oportunidade de aprender bastante sobre redes, sistemas distribuídos, sistemas operacionais e computação em nuvem. Outra experiência obtida foi sobre o funcionamento do Linux. Tivemos que usar vários tipos de comandos e conceitos sobre seus processos para preparar todo o ambiente para configuração da nuvem.

5.7 RECOMENDAÇÕES

Com a experiência obtida com esse projeto, ficou bem claro que qualquer pessoa ou organização que queira implementar uma nuvem tem muitas vantagens para usufruir, como: economia, escalabilidade, disponibilidade e utilização de recursos ociosos. Mas por outro lado, o que mais preocupa é a questão da segurança e integridade dos dados que trafegam pela rede da nuvem. Portanto, é recomendado que a nuvem seja instalada por uma equipe experiente, com capacidade de assegurar que todas as informações que passem pela rede sejam bem criptografadas, com a maior segurança possível para que outras redes externas não possam violar a integridade das informações trafegadas pela rede. E para a construção da nuvem usar o *OpenStack* como ferramenta, já que é uma ferramenta que têm uma comunidade

bastante movimentada, grandes empresas cada vez mais contribuindo e que está se destacando atualmente.

6 CONCLUSÃO E TRABALHOS FUTUROS

Finalmente nesse capítulo, é feita a conclusão do projeto, assim como a recomendação de trabalhos que possam ser feitos com base na experiência obtida neste.

6.1 CONCLUSÃO

Como foi visto em todo o projeto, a computação em nuvem nos fornece grandes vantagens que podem ser o diferencial em alguns tipos de organização. Mas, para atingir essa meta, é preciso um esforço para implantação de uma nuvem computacional de boa qualidade. Com esse trabalho, a UFPB terá informações de grande valor para aumentar o nível de ensino, já que os alunos terão a possibilidade de usar um tipo de tecnologia completamente nova, e que está crescendo rapidamente no mercado. No futuro, os alunos e professores poderão usar esse projeto para implantar uma nuvem computacional, visando a melhor forma de usar os recursos disponíveis. Portanto, a construção da nuvem é viável, levando em consideração todas as vantagens descritas anteriormente, mas por outro lado, há muita complexidade em se implantar uma nuvem como essa em um ambiente que já existe uma rede completa em funcionamento.

6.2 SUGESTÕES DE TRABALHOS FUTUROS

Com base em todo o conhecimento adquirido nesse projeto, a universidade é capaz de montar uma equipe experiente em redes e áreas afins para criar uma nuvem, com o objetivo de melhorar suas atividades acadêmicas e facilitar os trabalhos realizados pelos alunos. A UFPB pode disponibilizar um mini-curso sobre computação em nuvem, explorando seus conceitos teóricos, e ensinando o funcionamento das duas ferramentas, isso seria de grande valor para os alunos, já que essa área está crescendo rápido. Outro projeto que a UFPB pode realizar, é a terceirização das máquinas depois que a nuvem estiver pronta, compartilhando o conhecimento para que outras comunidades acadêmicas tenham a possibilidade de entrar nesse tipo de mercado.

7 REFERÊNCIAS

ALANIS P. Abmar B. Francisco B. Marcos N. **“Uma nuvem privada oportunista para execução de aplicações Bag-of-Tasks”** em SBRC 2013.

DRIVE G. C. **“Eucalyptus open-source cloud infrastructure an overview”** Eucalyptus Systems, Inc., 2009.

SMITH, J.R, Nair, R. “**The architecture of virtual machines**”. IEEE Computer, v.38, n.5, pp. 32-38, 2005.

SILBERCHATZ, A.; Galvin, P; **Sistemas Operacionais.** (1ª edição). Campus, Rio de Janeiro, 2001.

CHIRIGATI, Fernando Seabra. "**Computação em Nuvem**". Rio de Janeiro, RJ. 2009. Disponível em: http://www.gta.ufrj.br/ensino/eel879/trabalhos_vf_2009_2/seabra/. Acessado em Julho de 2013.

TAURION, Cezar. **Cloud Computing: Computação em Nuvem: Transformando o mundo da tecnologia da informação.** 1.ed. Rio de Janeiro: Brasport, 2009.

VAQUERO, L.M. Rodero-Merino L. , Caceres, J. , Lindner, M. **, "A break in the clouds: towards a cloud definition"** em ACM SIGCOMM Computer Communication Review, 2008

WEI, G. V. Athanasios, Y. Zheng and N. Xiong, “**A game-theoretic method of fair resource allocation for cloud computing services**” J. Supercomputing, 2009, DOI 10.1007/s11227-009- 0318-1.

PEPPLE, K. "**Deploying OpenStack**". O'Reilly Media, Inc., 2011
Johnson, D. Murari K, Raju M, Suseendran R, Girikumar Y. **"Eucalyptus Beginner's Guide"**, 2010, UEC Edition

Openstack. OpenStack Compute. Disponível em: http://www.openstack.org/software/openstack-compute/. Acessado em Setembro de 2013.

Openstack. Welcome to Swift's documentation!. http://docs.openstack.org/developer/swift/. Acessado em Setembro de 2013.

Openstack. Welcome to Glance's documentation. Disponível em: http://docs.openstack.org/developer/glance/. Acessado em Setembro de 2013.

Eucalyptus. Introduction to Keypairs. Disponível em: https://engage.eucalyptus.com/customer/portal/articles/256624-introduction-to-keypairs. Acessado em Agosto de 2013.

CSS Corp Open Source Services. Disponível em: http://cssoss.wordpress.com/2010/05/10/eucalyptus-beginner%E2%80%99s-guide-%E2%80%93-uec-edition-chapter-7-%E2%80%93-network%C2%A0management/. Acessado em Agosto de 2013.

DevStack stack.sh. Disponível em: http://devstack.org/stack.sh.html. Acessado em Setembro de 2013.

Ubuntu Manuals. Disponível em: http://manpages.ubuntu.com/manpages/lucid/man8/iptables.8.html. Acessado em Setembro de 2013.

OurGrid. Disponível em: http://www.ourgrid.org/. Acessado em Setembro de 2013.

CY13-Q1 Community Analysis — OpenStack vs OpenNebula vs Eucalyptus vs CloudStack. Disponível em: http://www.eucalyptus.com/blog/2013/04/02/cy13-q1-community-analysis-%E2%80%94-openstack-vs-opennebula-vs-eucalyptus-vs-cloudstack. Acessado em Outubro de 2013.

Amazon; Disponível em: http://aws.amazon.com/pt/ebs/. Acessado em Outubro de 2013.