Douglas Oliveira Camata

Modelagem de cloud computing para gerenciamento de recursos computacionais na UENF

Campos dos Goytacazes - RJ

2014

Douglas Oliveira Camata

Modelagem de cloud computing para gerenciamento de recursos computacionais na UENF

Monografia apresentada ao curso de graduação em Ciência da Computação da Universidade Estadual do Norte Fluminense Darcy Ribeiro como requisito para obtenção do título de Bacharel em Ciência da Computação.

Orientadora:

Prof[a]. Dr[a]. Annabell del Real Tamariz

Universidade Estadual do Norte Fluminense Darcy Ribeiro
Laboratório de Ciências Matemáticas

Campos dos Goytacazes - RJ

2014

Douglas Oliveira Camata

Modelagem de cloud computing para gerenciamento de recursos computacionais na UENF

Monografia apresentada junto ao Curso de Ciência da Computação, da Universidade Estadual do Norte Fluminense Darcy Ribeiro – Campos / RJ, como requisito para obtenção do título de Bacharel em Ciência da Computação. Orientadora: Profª. Drª. Annabell del Real Tamariz.

Aprovado em 25/09/2014.

COMISSÃO EXAMINADORA

Profª. Drª. Annabell del Real Tamariz
Orientadora - Universidade Estadual do Norte Fluminense Darcy Ribeiro

---

Prof. Dr. Luis Antônio Rivera Escriba
Universidade Estadual do Norte Fluminense Darcy Ribeiro

Prof. M.Sc. Rodrigo Manhães
Universidade Estadual do Norte Fluminense Darcy Ribeiro

AGRADECIMENTOS

À meu pai, meu melhor amigo e grande companheiro, por estar sempre presente em todos os momentos felizes e tristes da minha vida e por ser a pessoa mais importante na formação do meu caráter, por eu me tornar quem sou.

À minha mãe e meus irmãos, que mesmo distantes estão sempre por perto e também são parte da minha motivação.

Aos amigos Eduardo, Tarsis e Hugo, que me convidaram para o *glorioso* NSI, que foi fundamental para minha formação e meu futuro profissional.

Ao amigos e chefes Rogério, Fábio e Fernando, por estarem sempre no NSI dando oportunidade e orientando muito bem todos os bolsistas, fazendo acontecer e puxando a nossa orelha quando necessário.

À todos os companheiros do NSI, desde os mais antigos até os mais novos, por evoluirmos juntos e posteriormente podermos compartilhar nossos conhecimentos com os novos bolsistas e sermos reconhecidos e níveis nacional e internacional.

À professora Annabell, pela amizade, pelo ânimo e por confiar na gente na hora de começar coisas novas e pela paciência para aguentar a todos nós, alunos, durante todo esse tempo.

À todos meus companheiros de curso, principalmente Eduardo, Thawan e Pedro, por estarem na luta comigo desde o início de tudo, estudando, se divertindo e, as vezes, até se dando mal juntos.

À toda minha família, meus avós, meus primos e meus tios, por todo o suporte e felicidade que me proporcionam sempre.

Resumo

O objetivo deste trabalho é apresentar um estudo de caso de um ambiente de *cloud computing*, realizado na Universidade Estadual do Norte Fluminense (UENF), construído através do conjunto de ferramentas de código livre conhecido como *OpenStack*. Esta monografia apresenta um apanhado de problemas comuns em instituições públicas, de ensino e pesquisa com relação a gerencia de seus recuros computacionais, já identificados por diversos autores, alguns conceitos que servem de base para o *cloud computing* e como ele pode resolver estes problemas. Também é apresentado um resumo sobre a história do *OpenStack*, um relato sobre sua instalação, alguns problemas enfrentados e a interface final que é fornecida para os utilizadores desta *cloud*.

Abstract

This work intends to present a study of case of a cloud computing environment in Universidade Estadual do Norte Fluminense (UENF), build using the open-source tools known as OpenStack. It presents a bunch of common problems in public, learning and research institutions related to their computational resources management, already identified by many other authors, some concepts which serve as base for cloud computing and how it can solve these problems. This paper also presentes a little briefing of OpenStack's history, a report about its instalation, the problems encountered, and the final interface that is provided to the clients of this cloud.

Sumário

Lista de Figuras

Lista de Tabelas

1 Introdução

Segundo Truong et al. (2012), muitas instituições de ensino superior e pesquisa de países em desenvolvimento sofrem com diversos problemas relacionados com o gerenciamento dos seus recursos computacionais. Eles trabalham com grupos que apresentam 3 características:

- Não possuem acesso adequado aos recursos computacionais para pesquisa e ensino;
- Se envolvem em alta carga de atividades de ensino;
- São formados por poucos membros e possuem um orçamento reduzido.

Estas atividades (pesquisa e ensino), ainda apresentam outras importantes características. Por exemplo, elas fazem um grande uso de recursos computacionais em curtos períodos de tempo, devido à estrutura dos semestres letivos, enquanto outros são usados mais regularmente (TRUONG et al., 2012).

Apesar dos avanços em tecnologia, o compartilhamento e gerenciamentos destes recursos entre instituições de ensino tem recebido uma grande prioridade, principalmente nas regiões em desenvolvimento (BO; HONGYU, 2011). Ekwe (2012) diz ainda que devido às altas demandas de recursos por estas instituições, os grupos dificilmente conseguem adquirir ou até mesmo manter recursos computacionais de alta performance.

Muitos estudos mostram que *cloud computing* tem benefícios distintos, como redução de custos, utilização eficiente de recursos computacionais, flexibilidade e provisionamento elástico (ARMBRUST et al., 2010). Além disso, Truong et al. (2012) diz que todos estes benefícios se aplicam às duas principais atividades em instituições de ensino superior: pesquisa e ensino. Ele também diz que existem muitos outros artigos analisando o quão atrativo é o *cloud computing* para países em desenvolvimento. Ele cita o exemplo de Greengard (2010), que apresenta vários benefícios, como fácil acesso à infraestrutura com baixo custo, melhorando os esforços para colaboração e o acesso a hardware e softwares

mais recentes. Estes benefícios foram de fato confirmados pelo trabalho de Truong et al. (2012).

Até mesmo instituições governamentais, como o Governo Federal dos Estados Unidos da América, identificaram *cloud computing* como uma ótima solução para os problemas que estavam enfrentando que sua estrutura de tecnologia de informação (KUNDRA, 2011). Estima-se que cerca de 20, dos seus 80 bilhões de dólares de investimento em TI, serão utilizados para a migração de seus sistemas para este novo paradigma.

1.1 Problemática e justificativa

Segundo Ekwe (2012), a característica principal relacionada a pesquisa em instituições de ensino é que os grupos de pesquisa são pequenos e não são bem coordenados. Como resultado, os investimentos nas pesquisas acabam sendo *ad-hoc*, sob demanda e muito frequentemente não coordenados. Ainda dentro deste assunto, o autor diz que estes grupos tendem a resolver problemas pequenos e que não são recentes, porque não possuem recursos suficientes para resolver problemas maiores e mais complexos. Alguns grupos precisam de acesso a computação de alta *performance* para simulações e análises de seus modelos. Porém, o orçamento reduzido acaba limitando muito a performance dos recursos computacionais que a instituição é capaz de adquirir. Todas estas características culminam nos seguintes problemas:

- Recursos fragmentados e inadequados para resolução de grandes problemas. Eles são divididos em pequenos e desconectados grupos, capazes de resolver somente pequenos problemas.
- Os pesquisadores querem ter total controle sobre seus recursos, principalmente para customizá-los para seus próprios problemas. Esta atitude não facilita, mas sim atrapalha, o compartilhamento destes recursos.
- O baixo orçamento torna muito difícil a aquisição de recursos computacionais. Porém, a fragmentação e falta de esforços coordenados em dividir e compartilhar os recursos já obtidos resultam em uma subutilização destes.

Um estudo feito por Alabbadi (2011) mostra que existem muitos modelos de *cloud computing* que alcançaram o sucesso dentro de instituições de ensino, utilizando soluções comerciais e não-comerciais, terceirizadas e não-terceirizadas. Porém, existem várias "formas" de *cloud computing* e é essencial que as instituições de ensino que desejam utilizar

este modelo escolham a forma adequada para melhor satisfazer suas necessidades computacionais.

Pode-se identificar estes mesmos problemas anteriormente mecionados na Universidade Estadual do Norte Fluminense (UENF). Como exemplo, há o caso de alguns professores do Centro de Ciências Biológicas, onde eles possuem seu conjunto de recursos computacionais de alta performance. Estes recursos são relativamente pequenos para soluções de grandes problemas, com necessidades de software bem específicas, os professores desejam 100% destes recursos disponíveis quando eles desejam executar seus procedimentos e não admitem que existam outros procedimentos concorrentes além dos seus próprios. Além destes poblemas, ainda há a dificuldade burocrática e financeira envolvida na compra de recursos adicionais, pois a compra demora uma quantiadade considerável de tempo até ser efetivada e o orçamento da universidade dificilmente permite a compra de equipamentos de última geração e com configurações de topo de linha.

Há também o problema de gerenciamento das aplicações desenvolvidas por alunos e professores que são disponibilizadas para o público. Atualmente, por exemplo, os alunos do curso de Ciência da Computação dispõem de apenas um computador pessoal para ser utilizado como servidor. Diversos problemas são enfrentados pelos alunos, dentre eles, os mais graves são a correta configuração e instalação dos programas necessárias para que sua aplicação funcione e ainda sem interferir com outras aplicações instaladas no mesmo servidor, e questões relacionadas a segurança: quem deve ter a senha do servidor compartilhado, todos ou só um? Estes problemas acabam causando uma ilha de conhecimento, pois por diversos motivos, nem todos oa alunos podem ou precisam aprender isso e apenas um deles é escolhido para fazer todas as instalações, culminando numa dependência muito grande de uma única pessoa.

1.2 Objetivos

Este trabalho tem como objetivo apresentar um modelo de *cloud computing*, definido pela escolha de uma ferramenta e modelo adequados para sua implantação, capaz de minimizar estes problemas, e todos os outros já mencionados no início deste capítulo, dentro da Universidade Estadual do Norte Fluminense, especialmente para a disponibilização de projetos desenvolvidos pelos alunos e pesquisadores do Laboratório Ada Lovelace/LCMAT, mas não limitando-o a este propósito. Tal modelo é concebido através do *framework* proposto por Alabbadi (2011). Também é um objetivo deste trabalho fazer uma discussão

mais profunda sobre os detalhes técnicos da instalação do modelo, prover instruções para este processo, analisar as características do produto resultante e do processo, como um todo, verificar o nível do desenvolvimento do conjunto de ferramentas escolhido.

1.3 Metodologia

Vários trabalhos já aqui mencionados enumeram casos de sucesso, onde o *cloud computing* soluciona os problemas encontrados por instituições de ensino localizadas em países em desenvolvimento com relação à gerência e aproveitamento dos seus recursos computacionais. Apesar de todos eles mencionarem que a solução foi alcançada, nenhum deles apresenta mais a fundo quais foram as ferramentas e as configurações destas que foram utilizadas pelas instituições, muito menos a razão da escolha de determinada ferramenta em relação a outras e os procedimentos seguidos para a implantação do ambiente.

Este trabalho pretende abordar estes pontos que ainda não foram foco de muita literatura, escolhendo o *OpenStack*, que é um dos conjuntos de ferramentas para *cloud computing* mais utilizado e que evolui com mais velocidade, fazendo uma análise dos seus componentes, prosseguindo com sua instalação em um pequeno ambiente de testes e análise de suas funcionalidades, segurança e facilidades.

1.4 Organização

Este trabalhado será apresentado seguindo a estrutura de capítulos abaixo:

1. Introdução: onde são brevemente apresentados alguns casos de uso, os objetivos e justificativa, e a metodologia de desenvolvimento do mesmo.

2. Fundamentação teórica, onde serão apresentados conceitos, métodos e definições relacionadas a *cloud computing*, suas vantagens e desvantagens, assim como o estado de arte do tema.

3. Apresentação do experimento, das ferramentas escolhidas para sua execução e das razões que levaram à escolha destas em relação a outras por meio de comparações de prós e contras. Enumeração das possíveis configurações para instalação, assim como das características ótimas para obter o maior desempenho através deste novo paradigma. Inclui também informações instrutivas sobre a configuração destas ferramentas em um ambiente de testes pré-determinado.

4. Descrição do funcionamento das ferramentas no ambiente de teste mencionado no item anterior. Verificação das vantagens reais das ferramentas através deste ambiente e comparação com o paradigma de computação tradicional.

5. Conclusões sobre o trabalho, considerações finais e comentários sobre possíveis trabalhos futuros.

2 Cloud Computing

Muitas são as definições encontradas nos trabalhos atuais para cloud computing (ALABBADI, 2011), porém será adotada para este trabalho a mesma definição utilizada por este autor, que foi criada pelo *NIST* (Instituto Nacional de Padrões e Tecnologia dos Estados Unidos da América). Em Mell e Grance (2011), o NIST define *cloud computing* como: "um modelo acesso via rede, ubíquo, conveniente e sob-demanda a uma pilha de recursos computacionais configuráveis (redes, servidores, armazenamento e serviços) que podem ser rapidamente provisionados e lançados com o mínimo de esforço em gerenciamento ou interação com o provedor de serviços". Também são definidos pelo NIST quatro modelos de implantação de *cloud computing*:

Private cloud (nuvem privada) é provisionada e utilizada por uma única organização. Pode pertencer e ser gerenciada pela própria organização, terceiros, ou uma combinação destes e estar dentro ou fora dos limites físicos da organização.

Community cloud (nuvem comunitária) é provisionada e utilizada por uma comunidade de organizações que tem interesses comuns. Pode pertencer e ser gerneciada por uma das organizações, terceiros ou uma combinação destes e estar dentro ou fora dos limites físicos da organização.

Public cloud (nuvem pública) é privisionada para o uso do público em geral. Pode pertencer e ser gerenciada por uma empresa, instituição de ensino ou governamental, ou uma combinação destes e estar dentro dos limites físicos do seu provedor.

Hybrid cloud (nuvem híbrida) uma composição de dois ou mais dos modelos já mencionados que mantêm-se entidades únicas, porém são ligadas por tecnologia padronizada ou proprietária que possibilita compartilhamento de recursos entre elas.

Algumas características são essenciais ao cloud computing, segundo Alabbadi (2011): *self-service* sob-demanda, um consumidor do serviço pode provisionar seus recursos sem

a necessidade de intervenção humana; acesso a rede de banda larga, que será o meio por onde o cliente se comunicará e acessará o serviço requisitado através de mecanismos padronizados, que permitam que isto seja feito a partir de smartphones, notebooks, etc; *pool* de recursos, onde os recursos físicos são frequentemente utilizados através de um *pooling* para servir a múltiplos clientes com diferentes recursos físicos e virtuais dinamicamente determinados pelos próprios clientes; rapida elasticidade, onde estes recursos virtuais podem ter suas capacidades rapidamente modificadas e provisionadas, até mesmo automaticamente, tanto para cima quanto para baixo; e serviço mensurado, onde o provedor pode controlar e otimizar o uso de recursos físicos através de um sistema de mensuração em algum nível do ambiente.

Este mesmo autor ainda diz que o cloud computing só se tornou realizável graças a outras 4 tecnologias já existentes:

Virtualização possibilita a criação de recursos computacionais virtualizados (redes, servidores e/ou armazenamento). Oferece consolidação dos recursos computacionais físicos, separação e proteção.

Grid computing possiblita a execução de tarefas em múltiplos computadores dispersos, fornecendo de maneira transparente a capacidade de um supercomputador.

Utility computing possibilita o empacotamento e provisionamento dos recursos computacionais virtualizados através de um serviço de mensuração e um sistema de preços para que estes sejam oferecidos sob-demanda.

Web Service são "pedaços" de *software* que são publicados e num registro para serem descobertos dinamicamente e usados por outros *softwares* através de um protocolo de transporte (HTTP, TCP/IP, etc) independente da plataforma ou linguagem de programação.

Dentro do conceito de *cloud computing*, o NIST identificou 3 tipos de serviços básicos que este modelo de computação é capaz de oferecer. São eles:

- Infraestrutura como serviço (IaaS): são oferecidos recursos computacionais de base, como redes, servidores e armazenamento através da rede. Possibilitando todos o poder de gerenciamento de um servidor físico, porém livrando seu usuário das preocupações relacionadas ao *hardware* do servidor.

- Plataforma como serviço (PaaS): oferece ambientes pré-configurados com determinados softwares, para que os desenvolvedores possam facilmente desenvolver suas soluções usando-os e publicá-las sem as complicações de ter que gerenciar manualmente um servidor com todos estes softwares.

- Software como serviço (SaaS): esta camada oferece um software usável diretamente ao cliente, com total transparência do servidor ou plataforma necessário para este. Tudo é devidamente abstraído e gerenciado pelas ferramentas usadas no cloud computing. Os softwares oferecidos podem ser os mais diversos, como aplicações de produtividade ou Gestão de Relacionamento com o Cliente (CRM).

Cada um destes modelos é usado como base para o modelo acima poder oferecer um serviço que terá mais valor para o cliente leigo, vide Figura 1. Assim, entende-se que quando um desenvolvedor faz uso do seu serviço de PaaS, podem ser instanciadas várias unidades de IaaS e quando um cliente faz uso do SaaS, várias instâncias de PaaS (e, consequentemente, IaaS) podem ser criadas. A quantidade de instâncias de nível inferior necessárias depende das configurações determinadas para cada item dentro do serviço requisitado.

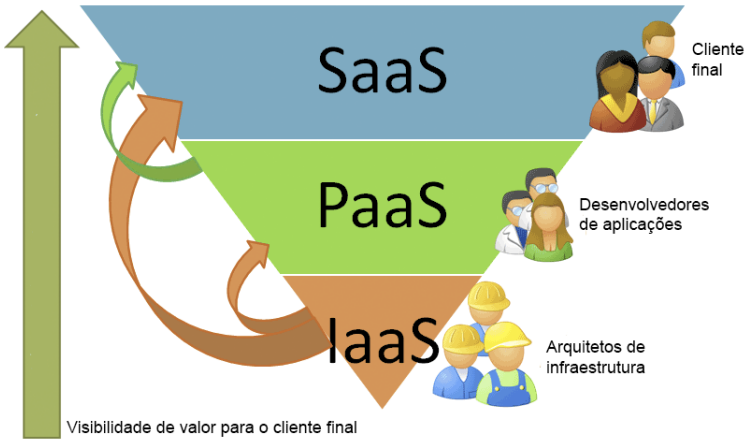

Figura 1: Pirâmide representando os 3 serviços oferecidos por uma cloud

2.1 Virtualização

Virtualização é a técnica que permite particionar um único sistema computacional em vários outros, denominados máquinas virtuais (CARISSIMI, 2008). Segundo este mesmo autor, estas máquinas virtuais oferecem um ambiente extremamente similar ao de uma máquina física. Possuem seu próprio sistema operacional, aplicativos e serviços de rede.

Ainda é possível conectar um conjunto dessas máquinas virtuais através redes, switches ou roteadores virtualizados.

São diversas as tecnologias que implementam a virtualização de recursos computacionais. Alguns dos mais conhecidos e evoluídos softwares desta área são: Hyper-V, Xen, KVM, VMWare ESXi. Laureano e Maziero (2008) definem 3 partes básicas que podem ser observadas em um ambiente de virtualização:

1. O sistema nativo (real), conhecido também como sistema hospedeiro, que contém todo o hardware físico;

2. O sistema virtual, conhecido também como sistema convidado, que executa o sistema virtualizado, simulando a maioria dos recursos de hardware que um sistema nativo poderia ter;

3. A camada de virtualização, conhecida como *hipervisor*, que faz a comunicação entre a interface virtual e nativa. Esta camada é representada pelos softwares de virtualização, como os já mencionados Hyper-V, Xen, KVM e VMWare ESXi.

Este mesmo autor ainda diz que *hipervisors* são mais complexos do que podem parecer, por exemplo, caso o conjunto de instruções da máquina virtual e da real sejam diferente (Linux virtualizado dentro de um Windows, ou vice-versa) haverá a necessidade da conversão de instruções pela CPU da máquina real. É necessário também mapear os recursos de hardware da máquina real (como mouses, armazenamento externo, etc) que poderão ser acessados pela máquina virtual. Contudo, todos os *hipervisors* conhecidos já são extremamente evoluídos e lidam muito bem com todas as complicações necessárias para possibilitar uma virtualização adequada.

Carissimi (2008) diz que é comum encontrarmos infra-estruturas com a filosofia de "um servidor por serviço", por motivos de heterogeneidade das necessidades dos clientes e, alguns casos, até segurança. Normalmente, este contexto acaba deixando o processador e alguns outros recursos subutilizados.

A virtualização resolve este problema de subutilização de recursos computacionais, através do tratamento de ambas suas duas mais comuns causas: a filosofia "um servidor por serviço" e a heterogeniedade das necessidades dos clientes. Obtém-se diversos sistemas operacionais contidos em apenas um hardware físico. Com uma máquina virtual disponibilizada para cada serviço, ela pode ter o sistema operacional, bibliotecas e exatamente a configuração de memória, processador e armazenamento necessários para

aquele serviço funcionar, onde todos estes recursos podem ser inclusive alterados e dimensionados de acordo com o crescimento ou decrescimento do consumo destes pelo serviço. Resumidamente, temos a flexibilidade, portabilidade, um melhor aproveitamento dos recursos do hardware físico através do uso da quantidade adequada de máquinas virtuais e elasticidade de executar diversos serviços diferentes, em sistemas operacionais diferentes, com necessidades de recursos diferentes e que mudam conforme o tempo dentro, tudo dentro de um mesmo hardware. Este ato é conhecido como consolidação de servidores e é imensamente importante para *data-centers*, por exemplo, devido a diversas demandas de sistemas operacionais por seus clientes, menor ocupação de espaço por máquinas físicas, melhor aproveitamento de energia elétrica, refrigeração, cabeamento, etc.

2.2 Grid Computing

Segundo Foster e Kesselman (1999), o principal objetivo do *grid computing* é multiplexar recursos computacionais, de diversos provedores (hardware físico), de maneira transparente para vários usuários. Desta maneira, o usuário percebe um único super-recurso, que na verdade é composto por vários outros recursos menores. Figueiredo, Dinda e Fortes (2003), enunciam que, sistemas tradicionais de computação, por sua vez, multiplexam os recursos de um único computador para vários usuários através da implementação de contas de usuários pelo sistema operacional. Este modelo assume que existe uma entidade administrativa 100% confiável que possui permissão para adicionar usuários e restringir seu acesso conforme necessário. Enquanto isso, sistemas de *grid computing* devem possuir vários domínios diferentes, cada qual sua própria entidade administrativa, e não podem depender de uma autoridade central.

Foster et al. (2008) nos mostram que a definição de *cloud computing* sobrepõe a de muitas tecnologias recentes, incluindo *grid computing*, *utility computing* e até *web services* (estes dois últimos serão devidamente mencionados nas duas próximas subseções). Segundo ele, a definição de *cloud computing* não só sobrepõe a de *grid computing*, ela é de fato uma forma evoluída do *grid computing* e usa-o como base para seu funcionamento. Tal evolução teve como motivação a mudança do foco, que antes era de infraestrutura para armazenamento e recursos de processamento, para recursos mais abstratos e serviços que apresentam um maior valor para seus usuários.

A sobreposição dos conceitos e a mudança de foco podem ser melhor entendidos visualizando a Figura 2. Através dela, é evidente perceber que *grid computing* é um

conceito mais focado em aplicações e *cloud computing* em serviços. Pode-se notar também que *clouds* utilizam *grids* e que *grids*, por sua vez, são baseadas *clusters* de computadores ou supercomputadores.

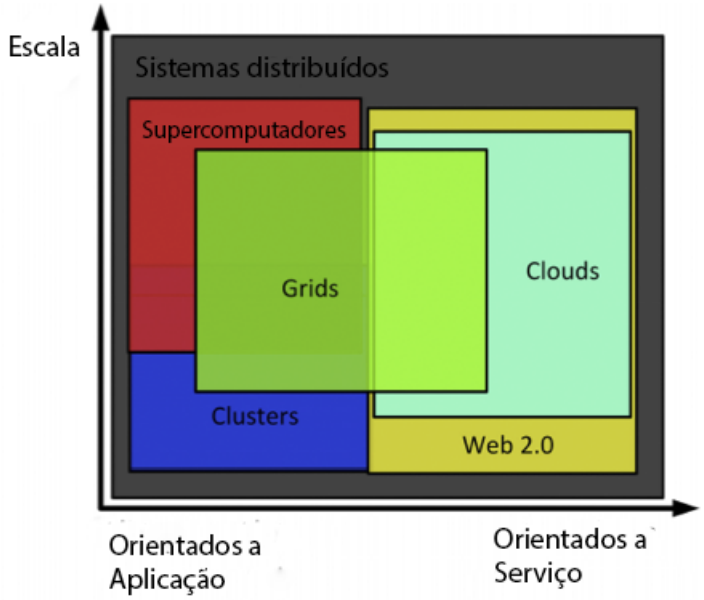

Figura 2: Comparação de *grid* e *cloud computing*, mostrando a sobreposição dos conceitos

Fonte: Foster et al. (2008)

De acordo com Foster (2002), a essência das definições do *grid computing* pode ser capturada de maneira simples em 3 itens:

1. É composta por recursos coordenados que não dependem de um controle centralizado. Ela integra e coordena recursos e usuário de diferentes domínios de controle, por exemplo, diferentes unidades administrativas da mesma compania, ou até mesmo diferentes companias, cuidando de tudo que diz respeito a segurança, políticas de acesso e qualquer outra configuração que venha a surgir de acordo com o contexto;

2. Usa protocolos e interfaces padronizados, abertos e de propósito geral. Todos estes itens devem tratar problemas fundamentais, como autenticação, autorização, descoberta e acesso a recursos. É muito importante que estes protocolos e interfaces sejam abertos, caso contrário o sistema seria específico para determinada aplicação;

3. Oferece uma qualidade de serviço não-trivial. Uma *grid* permite que seus recursos sejam coordenados de maneira a entregar qualidades de serviço complexas, relacionadas a tempos de resposta, *throughput*, avaliabilidade, segurança e co-alocação de múltiplos recursos para satisfazer as demandas de seus usuários. Assim, o sistema como um combinado tem uma utilidade maior que a soma de suas partes.

2.3 Utility Computing

Segundo Broberg, Venugopal e Buyya (2007), as técnicas tradicionais de gerenciamento de recursos (alocação, controle de admissão e agendamento) se mostraram inadequadas para *grids* compartilhadas, definidas na Seção 2.2. Estas técnicas de gerenciamento de recursos não fornecem facilidades para usuários requisitarem os recursos demandados de maneira apropriada e não refletem o valor, importância e prazos da necessidade do usuário. Este problema ainda afeta diretamente os provedores de *grids* compartilhadas, já que não oferece nenhuma forma de seus contratantes serem compensados pelos recursos compartilhados com outras entidades.

Não há como garantir que determinados usuários tenham acesso dedicado e com certos níveis de qualidade de serviço e uma determinada gama de recursos, como processamento, memória e armazenamento, não conseguindo extrapolar este limite. Pesquisadores e entusiastas encontraram técnicas de gerenciamento de recursos baseadas no mercado que resolvem essas limitações. Tais técnicas tem como objetivo fornecer um *framework* para incentivar os provedores de serviços priorizar a tarefas de maior importância, urgência e utilidade. Este *framework*, por sua vez, atua como *middleware* em sistemas de *grid computing* garantindo que todos os conceitos de *utility computing* sejam aplicados e respeitados dentro daquele ambiente.

> Existem muitas abordagens com relação ao *utility computing*. Amplamente, elas se dividem em duas categorias. A primeira é o modelo de *utilidade compartilhada*, onde os recursos são utilizados por diversos clientes ao mesmo tempo. A segunda, mais recente, é definida como modelo de *utilidade de servidor completo*, onde aplicações programaticamente adquirirem e liberam o servidor conforme for necessário. (ANDRZEJAK; ARLITT; ROLIA, 2002)

Para melhor entender as diferenças entre os modelos de utilidade compartilhada e de servidor completo, temos a Figura 3, demonstrando um modelo de utilidade compartilhada. Nele, cargas de trabalho são entregues a uma central de processamento, que irá distribuí-las entre vários servidores (unidades de processamento compartilhado). Já para exemplificar o modelo de servidor completo, temo a Figura 4: neste modelo, aplicações são delegadas a um ou mais servidores exclusivos (de acordo com a carga de processamento que esta aplicação necessita no momento), que processam nada além destas aplicações a eles atribuídas.

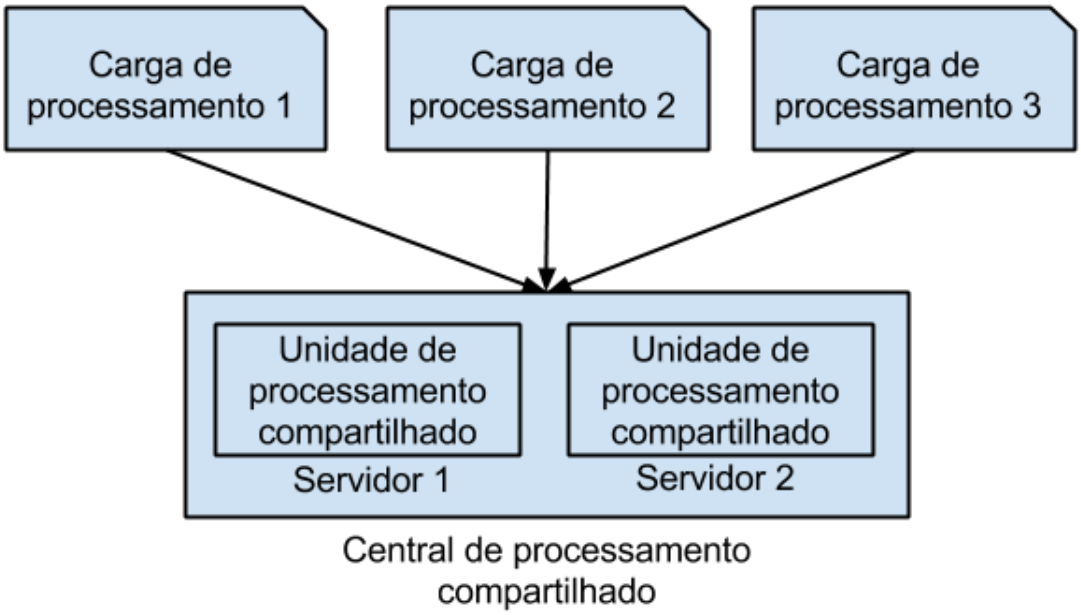

Figura 3: Modelo de utilidade compartilhada

2.4 Web Service

Chaudhary, Saleem e Bukhari (2002) definem web services como: uma entidade programática que fornece uma funcionalidade particular, como lógica de aplicação, e é disponibilizado para diversos potenciais usuários através da internet. Segundo o mesmo, um grande conceito por trás dos *web services* é o de arquiteturas orientadas a serviços (*service-oriented architecture*, em inglês, ou *SOA*). *Web services* adicionaram um novo nível de funcionalidade para a atual *Web*, tomando o primeiro passo em direção à integração de componentes de *software* distribuídos usando padrões da *web* (protocolo HTTP, XML ou JSON, URIs, SOAP, WSDL, etc), definidos pela *World Wide Web Consortium* (W3C) (ALONSO et al., 2004). Tal arquitetura sugere algumas atividades essenciais para o funcionamento dos determinados serviços, assim como sua utilização por clientes (internos ou externos à organização desenvolvedora do serviço). São elas:

1. Um serviço deve ser criado, com suas interfaces e métodos de chamada definidos;
2. Ele deve ser publicado, seja em intranet ou internet, para que possíveis usuários possam localizá-lo;
3. Deve ser localizável para que possa ser utilizado pelos potenciais usuários;
4. O serviço precisa ser utilizado para que ofereça algum tipo de benefício;

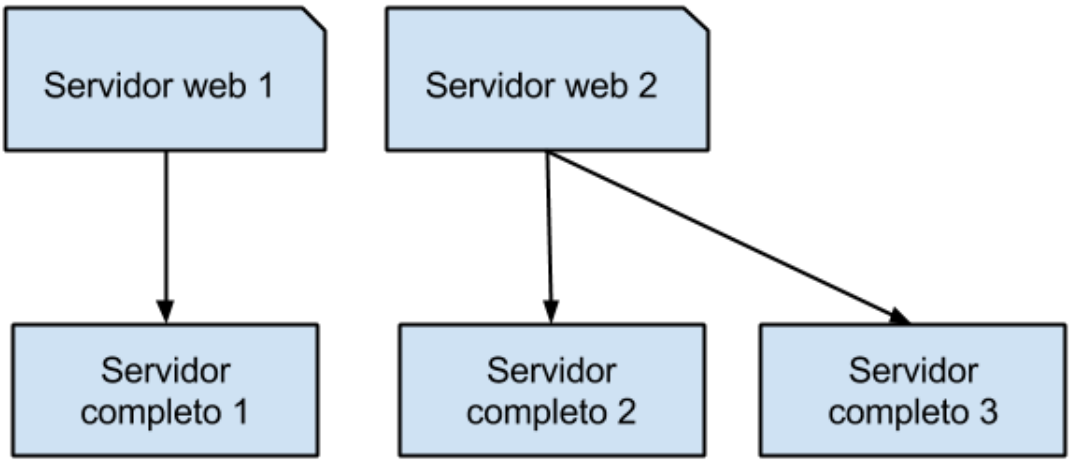

Figura 4: Modelo de utilidade de servidor completo

5. E o serviço preciso ser "despublicado", quando não é mais necessário ou não está mais disponível.

Pode-se ver na Figura 5, os passos necessários para que um cliente possa finalmente utilizar um web service. Primeiramente, após construído, este serviço (*service provider*) deve ser publicado em um índice (chamado de *service broker*). Então, um cliente (*service requester*), pode acessar o índice e procurar por um serviço disponível para ele. Após este cliente acessar os dados do índice, ele se conecta diretamente com o serviço e realiza a comunicação necessária para prosseguir com o uso deste.

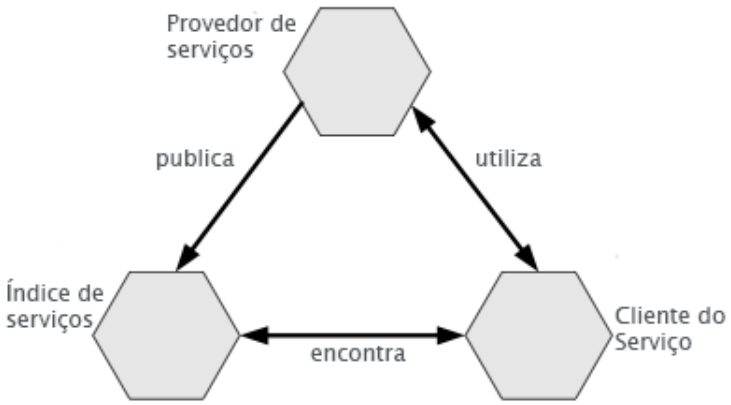

Figura 5: Fluxograma mostrando os passos para publicação, localização e uso de um web service

Adaptado para português de Chaudhary, Saleem e Bukhari (2002)

A comunicação baseada em tecnologias *web* garante a interoperabilidade dos *web services*, independentemente plataforma em que é desenvolvido ou consultado. Isto acaba

ajudando equipes a desenvolver um software altamente escalável, programando as partes mais custosas como serviços externos, usando as ferramentas mais adequadas para que haja o esperado ganho em escalabilidade, ou em legibilidade do código. Em um ambiente de *cloud computing*, escalabilidade e interoperabilidade são de suma importância para um aproveitamento eficiente dos recursos computacionais disponíveis para esta finalidade e garantir o crescimento da *cloud* sem problemas. É evidente que manter uma parte de um *software* isolada torna a manutenção e evolução do seu código muito mais simples do que fazer o mesmo no *software* completo, por inteiro. Cada parte pode ser testada e finamente polida, ainda aproveitando das melhores ferramentas que determinadas linguagens podem oferecer para a resolução de problemas específicos.

3 Trabalhos relacionados

São vários os trabalhos que já apresentam casos de sucesso de uso de diversos modelos e tecnologias de cloud computing. Alguns dão enfoque a instituições de ensino situadas em países em desenvolvimento, como o trabalho de Truong et al. (2012). Ele descreve a implantação de cloud computing na Universidade de Tecnologia de Ho Chi Minh, no Vietnã, para as tarefas de pesquisa e ensino.

No trabalho de Truong et al. (2012), recursos fixos e pré-configurados deixaram de ser fornecidos, em prol de recursos elásticos (que podem ser facilmente redimensionados) e móveis (podem estar inclusive em lugares físicos diferentes de acordo com a vontade do seu utilizador). Os pesquisadores forneciam os dados relacionados aos recursos que precisavam, seja de hardware ou software, e em segundos obtinham o devido acesso que necessitavam (IaaS). Eles também podiam ter acesso simples e direto a plataformas para desenvolverem suas pesquisas da maneira mais prática possível (PaaS). Além disso, foi criada uma rede social para que os pesquisadores compartilharem informações científicas mais facilmente entre eles.

Em resumo, Truong et al. (2012) nos dizem que o cloud computing solucionou um grande problema da universidade, que eram as diferentes necessidades de sistemas operacionais e softwares, muitas vezes conflitantes, e a massiva manutenção manual demandada para a instalação e configurações dos softwares adequados a cada um dos cursos. A cada mudança nas necessidades de software de um determinado curso, era necessário que este fosse instalado e devidamente configurado em cada máquina do laboratório, tomando muito tempo e esforço. Com *cloud computing*, as máquinas dos laboratórios passaram a ser servidas através de *IaaS*, onde cada curso possui uma imagem relacionada a ele, que cumpre todos os requisitos de software, hardware virtual e sistema operacional necessários. Quando há alguma modificação nesses requisitos, ela é feita diretamente na imagem e esta é novamente disponibilizada para todos os terminais de acesso. Facilita-se assim o gerenciamento do laboratório, eliminando uma boa parte do excesso de trabalho manual e ainda agilizando todo o processo de manutenção dos recursos computacionais

físicos e de software.

Anteriormente, em outro trabalho, Sultan (2010) apresentou um estudo de caso da Universidade de Westminster, no Reino Unido. Com mais de 22 mil alunos, ela é uma de várias universidades desse país que aderiu ao cloud computing. Tudo começou quando o serviço de email desta instituição começou a ser abandonado por parecer desatualizado. Através de pesquisas, foi descoberto que 96% dos estudantes desta universidade configuravam suas contas de email acadêmico para automaticamente redirecionar os emails para uma conta externa (SULTAN, 2010). Isto estava causando alguns problemas, dentre eles: os emails começaram a ser tratados como *spam* pelos emails externos dos estudantes e, consequentemente, eles estavam deixando de receber comunicados importantes; e os alunos acabavam frequentemente usando pendrives, que são muito suscetíveis a perdas ou mal-uso, porque o serviço não fornecia uma capacidade de armazenamento adequada.

Segundo este mesmo autor, para solucionar o problema, em 2007 a universidade começou a estudar o *Google Apps: Education Edition*. Através dele, seriam oferecidos email com 7.3gb de armazenamento (o suficiente para os alunos abandonarem o uso de pendrives), um sistema de troca de mensagens, calendário compartilhado e até uma suíte de escritório com aplicações para processamento de textos, planilhas e apresentações. Durante o ano 2009/9, após um determinado período de testes e *feedback*, o sistema foi implantado e os problemas apontados foram mitigados com sucesso. Os estudantes passaram a ter uma melhor experiência e a universidade estima uma economia de 1 milhão de libras esterlinas, caso fosse fornecer um serviço semelhante de forma independente.

Ainda há outros dois casos de uso em instituições de ensino mencionados por Sultan (2010). O primeiro, da Universidade da Califórnia, nos Estados Unidos da América, que está usando cloud computing para seu curso de desenvolvimento e implantação de aplicações SaaS (*software as a service*). Ajudada por uma doação da *Amazon Web Services* (AWS), todo o curso foi movido para a nuvem da empresa e eles podem fornecer a grande quantidade de servidores que o curso demanda em apenas alguns minutos (FOX, 2009). O segundo é o caso do Colégio de Medicina do Centro de Biotecnologia e Bioengenharia de Wisconsin, que está usando os servidores de cloud computing do Google para tornar pesquisas relacionadas a proteínas, que costumam ser caríssimas, disponíveis para cientistas em todo o mundo.

Contudo, as aplicações de *cloud computing* já estudadas não se limitam ao setor empresarial, de forma geral, e de ensino. Kundra (2011) nos mostra estudos sobre a sua utilização no Governo Federal dos Estados Unidos. Foi planejado investir 25% do

orçamento do governo (cerca de 20 bilhões de dólares) para migração de sistemas de todas as suas agências para *cloud computing*. Para auxiliar tais agências neste processo foi elaborada a Estratégia Federal de Cloud Computing, com o objetivo de:

- expor benefícios, considerações e dilemas do *cloud computing*;
- fornecer um framework de decisões e alguns casos de uso de exemplo para ajudar as agências a migrarem seus sistemas;
- destacar recursos para sua implementação;
- identificar atividades, paéis e responsabilidades do governo federal para catalisar a sua adoção.

Então elaborou-se um paralelo entre as principais características do ammbiente atual e do ambiente de *cloud computing* proposto, segundo 3 critérios: eficiência, agilidade e inovação, que pode ser visto na Tabela 1. Todas as agências, antes de migrar seus sistemas, deveriam fazer este comparativo aplicado a cada sistema específico a avaliar se esta migração oferecerá vantagens consideráveis. É incluso neste trabalho o exemplo do Centro de Experiências do Exército (AEC), onde foi identificada a necessidade de atualizar seu sistema de gerencimanto de relação com o cliente e após considerarem diversas opções, como atualizar seu sistema proprietário de 10 anos de idade, escolherem uma solução de *software* como serviço disponível comercialmente. Esta solução foi adaptada para todos os requisitos de segurança desta agência, proporcionava todas as funcionalides que eram necessárias e foi entregue a uma fração do tempo e dinheiro necessários para atualizar o antigo sistema legado. Durante o processo de escolha chegou-se às seguinte conclusões sobre os critérios propostos:

- Eficiência: o custo inicial de atualizar o sistema legado era de 500 mil a 1 milhão de dólares, enquanto o investimento inicial da solução de *software* como serviço teve o custo de 54 mil dólares;
- Agilidade: foi considerado também o tempo para atualização do antigo *software*. Apesar dos upgrades ao longo dos anos, era inviável customizá-lo com as necessidades do centro. Enquanto isso, a nova solução oferece provisiosamento em uma fração do tempo do seu antecessor, além de ser mais escalável e fácil de atualizar conforme o tempo;

Tabela 1: Benefícios do Cloud Computing (Fonte: (KUNDRA, 2011))

Eficiência	
Cloud Computing	Ambiente atual
• Alta utilização de recursos (maior que 60-70%) • Demanda agregada e sistema acelerado de consolidação • Produtividade no desenvolvimento e gerenciamento de aplicações, redes e usuários finais	• Baixa utilização de recursos (menor que 30%) • Demanda fragmentada e sistemas duplicados • Difícil gerenciamento
Agilidade	
Cloud Computing	Ambiente atual
• Comprado como serviço de fornecedores confiáveis • Incrementos e decrementos de capacidade quase instantâneos • Mais responsivo a necessidades urgentes	• Requer anos para construção de data-centers para novos serviços • Requer meses para aumentar a capacidade de serviços existentes
Inovação	
Cloud Computing	Ambiente atual
• Muda o foco da posse dos recursos para gerenciamento de serviços • Incentiva inovação no setor privado • Encoraja uma cultura empreendedora • Mais ligado a tecnologias emergentes	• Sobrecarregado com gerenciamento de recursos • Totalmente desacoplado de inovações no setor privado • Sem riscos, uso de tecnologia legada

- Inovação: a solução de *software* como serviço integra-se diretamente com email e *Facebook* e acesso a informação de qualquer lugar, através da *internet* e permite a agência aproveitar do sistema de inovações do seu provedor de serviços sem ter que gerenciar pesados recursos computacionais.

De acordo com o trabalho de Sylvia, Peterson e Uniack (2013), a IBM, grande empresa do ramo de computação, conhecida desde os seus primórdios, tem um caso de uso de sucesso absoluto de *cloud computing*. Na empresa, esta tecnologia é utilizada dentro das 6 principais cargas de trabalho de tecnologia da informação: desenvolvimento e testes, análise, armazenamento, collaboração, computação pessoal e aplicações em produção. Graças a este uso, segundo estes mesmos autores, a IBM melhorou sua eficiência, enquanto ao mesmo obteve impressivas economias em capital e operações. O trabaho apresenta as seguintes considerações:

- As equipes de desenvolvimento e testes presenciaram o tempo de provisionamento de um servidor cair de 5 dias ou mais para apenas uma hora. Isto impactou num

acelerado desenvolvimento de novas funcionalidades e na velocidade com que as aplicações desenvolvidas chegavam no mercado;

- A nuvem de análise de dados melhorou incrivelmente a área de *business intelligence*, e reduziu os custos de projetos desta área, que eram da magnitude de centenas de milhares de dólares. As organizações que utilizam esta nuvem tem acesso a informação agregada de centenas de *warehouses* (depositos). Estas informações tem altíssimo valor, chegando a 300 milhões de dólares, somente para as 20 maiores consumidoras, de um total de 300 organizações.

- Sua central de armazenamento baseada em *cloud computing* cortou o custo de armazenamento por *byte* em 50%, permitindo a IBM acomodar o grande crescimento de demanda por armazenamento (estimado em 25% ao ano) sem aumentar o orçamento total destinado a essa área.

- A rede social *IBM Connections*, hospedada em sua própria *cloud*, aumentao dramasticamente a colaboração entre seus funcionários e, consequentemente, a produtividade e inovação destes. Esta rede social suporta cerca de 50 milhões de minutos em conferência por mês.

Antes da implantação de tecnologias de *cloud computing*, vários gárgalos e problemas eram encontrados na IBM. Dentre estes problemas, por exemplo, o setor de desenvolvimento e testes enfrentava semanas de espera para obter um servidor, utilizava somente 10% dele e não liberava seu uso devido ao tempo de espera para obtê-lo novamente. Neste trabalho é apontado que 95% do provisionamento dos servidores é agora feito através de *cloud*, confirmando sua velocidade e facilidade de uso.

Ridley et al. (2014) faz uma análise sobre o assunto e chega a conclusão que os diversos serviços baseados em *cloud computing* já disponíveis fizeram com que as pessoas mudassem a maneira como pensam sobre conteúdo digital e como utilizá-lo. No setor empresarial, a implantação de *cloud computing* está aumentando constantemente, independente do nicho. O crescimento de serviços que podem ser oferecidos neste modelo de *utility computing* continuará a mudar o mercado, apresentar novos modelos de negócios e revolucionar a maneira como compartilha-se informações.

A Doutora Tua Huomo, em Ridley et al. (2014), diz que *cloud computing* não implica em apenas uma mudança de tecnologia, mas também forçará todos a adotarem processos operacionais e de desenvolvimento que são mais focados no valor para o cliente. Neste mesmo trabalho, o Professor Ian Biterllin prediz que qualidades únicas desta nova

tecnologia tornam-a muito mais amigável ao meio ambiente do que todos os outros paradigmas conhecidos, mitigando as grandes críticas recebidas por *data centers* quando ao seu consumo energético. Finalmente, o Dr Jonathan Liebanau, da Escola de Economia de Londres, em sua análise chega à conclusão que *cloud computing* trará diversas mudanças positivas na economia, porém cada país sentirá essas mudanças com uma intensidade diferente: elas dependerão de como os provedores de serviço, o governo e os diretores de empresas se adaptarão.

4 O Modelo Implementado

Para implementar o ambiente de teste de *cloud computing*, foi escolhida a pilha de ferramentas conhecida como *OpenStack*, implementada usando um nó para controle e *networking* da nuvem e outro nó para processamento. O modelo de rede escolhido para o ambiente de testes foi o que é chamado pelo *OpenStack* de *simple flat networking*, onde apenas uma interface de rede é utilizada para todas as comunicações entre os serviços. A instalação foi concebida com auxílio de uma ferramenta de automação fornecida pela *Rackpsace*, uma das empresas criadores do próprio *OpenStack* e uma das maiores do ramo de hospedagem baseada em *cloud computing*. Inicialmente, a intenção é de uma nuvem privada, somente para uso interno da universidade. É esperada uma evolução do projeto para uma nuvem híbrida, englobando também o conceito de uma nuvem pública, para oferecer para empresas e outras insituições da região os benefícios do *cloud computing*.

4.1 OpenStack

O *OpenStack* é um grupo de ferramentas, comumente chamado de "ecossistema" por sua comunidade desenvolvedora e utilizadora. Esse grupo é composto por diversos serviços, são eles:

- Keystone: serviço de autenticação entre os nós e serviços;
- Nova: serviço de processamento, que é responsável por distribuir, configurar e executar máquinas virtuais propriamente ditas;
- Glance: serviço responsável pela descoberta, registro e fornecimento de imagens de máquinas virtuais;
- Swift: serviço de armazenamento de objetos (arquivos), semelhante ao *Amazon Simple Storage Service* (S3);

- Cinder: serviço de armazenamento de blocos, responsável por armazenar os discos virtuais utilizados pelas máquinas virtuais;
- Neutron: serviço de gerenciamento de redes virtuais, responsável por simular switches e roteadores e atribuir IPs às máquinas virtuais;
- Horizon: serviço de gerenciamento da nuvem. Fornece uma interface web para controle de tudo o que os outros serviços fornecem.

Pode-se observa como cada parte se comporta e com quem se comunica dentro do sistema através da Figura 6.

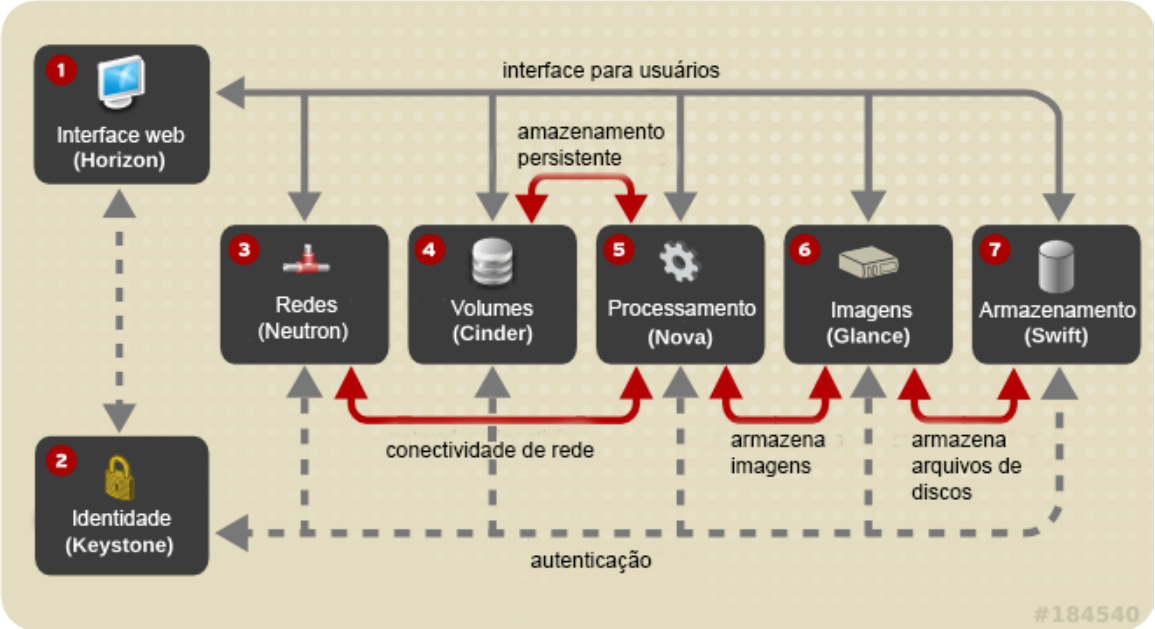

Figura 6: Arquitetura do OpenStack

Fonte: `https://access.redhat.com/documentation/en-US/Red_Hat_Enterprise_Linux_OpenStack_Platform/2/html/Getting_Started_Guide/ch01.html`

Segundo Dignan (2014), este ecossistema foi fundado pela NASA e pela *Rackspace Hosting* e a partir de então teve uma grande adesão de parceiros, tanto no meio científico quanto no meio corporativo. Um destes grandes parceiros na área científica é o *CERN* (Organização Europeia para Pesquisa Nuclear), que analisa dados coletados pelo famoso *LHC* (Grande Colisor de Hádrons). Purcell (2014) diz que o *OpenStack* é de grande importância dentro desta organização, fornecendo recursos computacionais em larga escala para físicos espalhados pelo globo com menos gargalos burocráticos.

Estes fatores, aliados às contribuições de diversos membros da comunidade de software livre para uma rápida evolução do ecossistema foram essenciais para a escolha destas

ferramentas. Hoje, o projeto conta com novas versões a cada 6 meses, sempre incoporando diversas funcionalidades e implementando sugestões de diversos usuários. A fundação do *OpenStack* é composta por um comitê técnico de 13 membros eleitos por contribuidores tecnológicos, um quadro de 24 diretores, sendo 8 apontados por membros, 8 eleitos por membros de classe e 8 eleitos por membros comuns, e por um comitê de usuários composto por mais de 75 grupos espalhados por todo o mundo.

4.1.1 História

Em 2008, os esforços da NASA para padronizar seus *websites* inspiraram uma explosão em tecnologias de *cloud computing* (RADEMAKERS; COLEMAN; REINY, 2012). O projeto, em seu início, era chamado de *NASA.net* e tinha como objetivo unificar todas abordagens usadas para desenvolver tais *websites*. Todos eles pareciam diferentes e eram gerenciados de maneiras diversas. A idéia era que os desenvolvedores pudessem criar seus *websites*, enviá-los para o *website* do *NASA.net* e ele tomaria conta de todos os passos necessários.

Apesar que o projeto estava apenas começando naquela época, seus desenvolvedores já identificaram a necessidade de ter grandes ferramentas na sua fundação. O próximo passo era construir um "serviço de infraestrutura". Foi criada então uma plataforma capaz de fornecer poder computacional para rodar serviços da *NASA.net* e que também poderia ser usado para outros tipos de aplicações. O que a equipe de desevolvimento percebeu foi que eles precisavam desenvolver um serviço de *cloud computing*. A idéia era se autenticar no serviço e dizer: preciso de 10 computadores e, dentro de um minuto, ter acesso a estes 10 computadores para qualquer que seja o propósito.

Conforme o escopo do projeto cresceu, ele foi renomeado para *Nebula*. O *Nebula* estava fazendo muitos mais do que criar padrões para os desenvolvedores *web* da agência: era fornecida uma grande gama de serviços para desenvolvedores, pesquisadores, e cientistas, para que eles pudessem acessar e gerenciar toda enorme quantidade de dados que a NASA acumulava dia após dia.

Um dos pontos decisivos sobre o sucesso do *Nebula*, antes de se tornar oficialmente *OpenStack* foi a escolha do modelo de *software* livre. *Software* proprietário é tão útil e conveniente algumas vezes que a equipe chegou a pensar se seria possível continuar o desenvolvimento do produto sem se apoiar neste modelo. Por outro lado, o modelo de desenvolvimetno de código aberto facilitaria um ambiente colaborativo: qualquer um com o conhecimento poderia melhorar o código. Esta foi então a escolha, segundo O'Brien (um dos criadores e idealizadores da ferramenta), desde o início era desejado que o projeto

envolvesse uma grande comunidade (empresas privadas, instituições acadêmicas e laboratórios de pesquisa, por exemplo) que pudesse contribuir e levar o *Nebula* a um nível mais alto. O *Nebula* acabou por evoluir e se tornar um projeto que estava atendendo demandas e necessidades gerais, não somente da NASA, mas do mundo todo. Entretanto, houve um problema com umas das partes do sistema, feita com software proprietário, que precisava ser migrada para uma linguagem aberta. A linguagem escolhida foi Python, a preferida da equipe de desenvolvimento.

A partir da publicação do *Nebula* como *software* livre, foi chamada a atenção de diversos membros da indústria. Um deles, uma empresa de hospedagem conhecida como *Rackspace*, que também desenvolvia suas próprias ferramentas para automatizar o processo de gerenciamento das aplicações que seus clientes queriam hospedar, estava pronta para liberar suas próprias ferramentas dentro do mesmo modelo de código aberto. Com a notícia, a Rackspace, que tinha paixão por *software* livre, entrou em contato com a equipe do *Nebula* para colaborar com o desenvolvimento do projeto. Um complementou o outro, e da fusão dos dois projetos foi anunciado o *OpenStack*, atraindo milhares de colaboradores por todo o mundo e centenas de grandes empresas. Antes, o que somente poderia ser feito pagamento para um terceiro, poderia ser então feito de maneira totalmente gratuita, com uma ferramenta gratuita e aberta.

4.1.2 Modelo de rede

Uma das características mais complexas do ecossistema escolhido é a arquitetura de rede. O ambiente de produção ideal deve fornecer 4 interfaces de rede para total separação dos tráficos de gerenciamento, dos sistemas de armazenamento, da rede privada das máquinas virtuais e da rede pública (que fornece IPs flutuantes) e assim atingir o máximo de performance.

De acordo com a documentação do *OpenStack*, é possível usar no mínimo duas interfaces de rede com um *switch* gerenciável, ou seja, capaz de roteamento de pacotes nas camadas 2 e 3 do modelo OSI, através de uma técnica conhecida como *VLAN tagging* ((SRIDHAR, 1998)). Também pode-se chegar ao caso mais básico e simples: uma única interface de rede, com o uso de um um nó de *networking* (usando a ferramenta *Neutron*) para desempenhar este papel, porém usando uma técnica conhecida como *GRE* (Generic Routing Encapsulation), que tem como desvantagem uma certa lentidão para o encapsulamento dos pacotes e gera um gargalo na inteface de rede deste nó (todo tráfego precisa passar por ele).

Para este ambiente de testes, logicamente, foi usado o modelo de rede mais simples, conhecido como *simple flat networking*, dada a facilidade de implementação deste utilizando a ferramenta Neutron e a técnica *GRE*.

4.1.3 Configuração e instalação

Foram realizadas diversas tentativas de instalação e configuração do projeto utilizando os manuais do próprio *OpenStack*, disponíveis na documentação oficial, porém sem sucesso. São diversas configurações espalhadas por arquivos diferentes e diversos comandos extremamente sensíveis. Se um erro é cometido, ele só será notado quando todas as ferramentas do ecossistemas forem instaladas e implicará em uma análise de *logs* para futura correção. Como solução para este problema, o próprio projeto recomenda o uso de ferramentas de automação, sendo as mais conhecidas: *Salt*, *Puppet* e *Chef*.

Para o desenvolvimento deste ambientes de testes, foi escolhida a ferramenta *Chef*, usada com scripts fornecidos pela própria Rackspace, por causa do excepcional suporte oferecido pela empresa, tanto para o seu uso quanto para toda a configuração necessária, através do programa *Rackspace Private Cloud*. Durante todo o processo (que será descrito abaixo), diversos problemas na ferramenta de automação fornecida pela Rackspace foram encontrados e reportados à equipe de desenvolvimento, que rapidamente corrigiu-os para que fosse possível continuar o procedimento. O programa *Rackspace Private Cloud* compreende um pacote de *software livre* para a instalação do *OpenStack*, treinamento e certificação, prestação de serviços e opções de suporte para auxiliar a construção e gerenciamento da nuvem privada. Como eles possuem a maior infraestrutura baseada no *OpenStack* do mundo, há possibilidade de acesso aos profissionais que participaram deste grande feito e enfrentaram os mais diversos problemas para tal infraestrutura.

O *Chef* é uma ferramenta de automação e gerenciamento de infraestrutura baseada no modelo cliente-servidor. Tarefas para controle de infraestrutura, chamadas de de *recipes* (receitas), definem instruções para instalar e configurar bancos de dados, servidores web e balanceadores de carga, por exemplo. Estas receitas, por sua vez, são compostas de *resources* (recursos), como um arquivo, um *template* ou um pacote do sistema a ser instalado. Há também a possibilidade de agrupar receitas em *roles*, ou papéis, para uma maior abstração da infraesturura, permitindo que diversas receitas sejam executadas em múltiplos nós. Todas as receitas e papéis são armazenados no servidor (*chef-server*) e os clientes se registram e sincronizam-se através do *chef-client*. A partir deste ponto, ao comando do servidor, os próprios clientes executam as receitas das dos papéis atribuitdos

a eles em si mesmos. Para configuração tanto do chef-server quando dos nós que serão provisionados através do *chef-client*, existe a ferramenta *knife*. Ela que envia o *cookbook* (conjunto de receitas) para o chef-server e faz o registro dos *chef-clients* neste servidor. Tal ferramenta pode ser instala no próprio *chef-server* ou em outro computador, conforme necessidade e conveniência. Veja a Figura 7 para uma melhor compreensão da organização e comunicação deste conjunto.

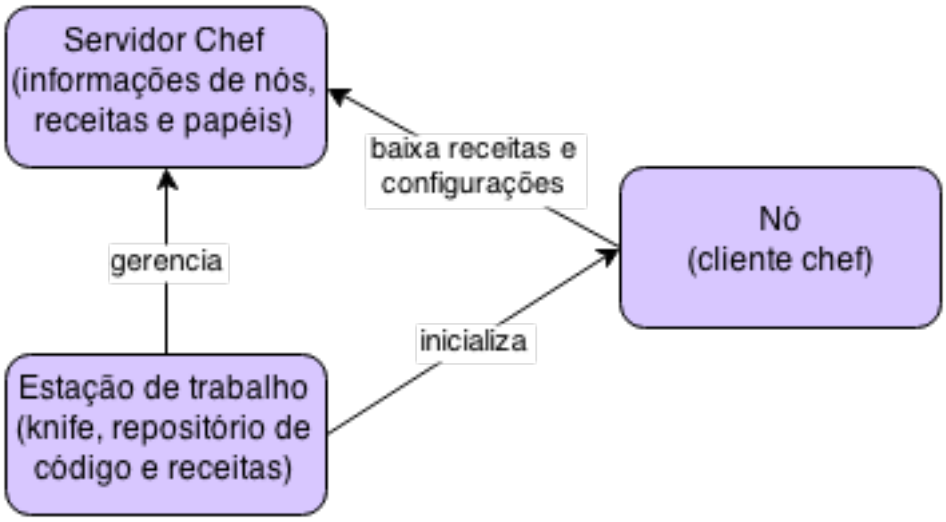

Figura 7: Fluxograma de funcionamento do Chef

Finalizando, para o ambiente de testes foram utilizados 4 computadores e um pequeno roteador, cedidos pelo Laboratório Ada Lovelace, do curso de Ciência da Computação da UENF. Cada um dos computadores possui a seguinte configuração:

- Processador: Core i5 de segunda geração, com 4 núcleos rodando a 3.4 Ghz;
- Memória: 4GB DDR3 1066 Mhz;
- Disco rígido: HDD 160GB 7200rpm.
- Sistema Operacional: Ubuntu Server 12.04 64 bits

Um deles foi configurado para ser o *chef-server*, um para ser o nó controlador da nuvem e um nó para processamento das máquinas virtuais. Basicamente, o processo de instalação consiste nos seguintes passos:

1. Preparação do nós (instalação de sistema operacional)
2. Instalação do *chef-server*, dos livros de receitas da Rackspace

3. Criação de um ambiente *chef* e definição dos seus atributos (configurações)
4. Instalação do *chef-client* nos nós
5. Atribuição e aplicação dos papéis aos nós controlador, de rede e de processamento

Como a preparação dos nós é aberta para ser feita de acordo com o costume do usuário, bastando que seja instalado o sistema operacional indicado, este passo não será descrito.

4.1.3.1 Instalação do chef-server e livros de receitas da Rackspace

Após preparar os nós, escolhe-se um para ser o *chef-server* e então, logado como o usuário *root*, instala-se o software com os comandos mostrados no Código 1

```
$ curl -s -O https://raw.githubusercontent.com/rcbops/support-tools/
master/chef-install/install-chef-server.sh
$ bash install-chef-server.sh
```

Código 1: Instalação do *chef-server*

Após este processo terminar, recarrega-se o terminal para atualizar as variáveis necessárias e testa-se o comando *knife* através dos comando no Código 2.

```
$ source /root/.bash_profile
$ knife client list
```

Código 2: Teste de instalação do *chef*

A instalação dos livros da receitas da Rackspace é feita através dos seus repositórios de código, seguindo os comandos do Código 3.

```
$ git clone https://github.com/rcbops/chef-cookbooks.git
$ cd chef-cookbooks
$ git checkout 4.1.2
$ git submodule sync
$ git submodule update
$ knife cookbook upload -a -o cookbooks
$ knife role from file roles/*rb
```

Código 3: Instalação dos livros de receitas da Rackspace

4.1.3.2 Criação do ambiente para configuração do OpenStack

A partir dai é iniciada a parte de criação do ambiente no *chef* e configuração das variáveis do *OpenStack* em si. Para criar o ambiente e abrir o arquivo de configuração, executa-se os comandos apresentados no Código 4.

```
$ knife environment create private-cloud -d "UENF Private Cloud"
$ knife environment edit private-cloud
```

Código 4: Criação do ambiente do *chef*, visualização e edição das variáveis deste ambiente

Ao executar o último comando, será aberto o arquivo em branco, onde deve-se sobrescrever as variáveis padrões definidas pela Rackspace de maneira adequada à infraestrutura onde o sistema está sendo configurado. Como mencionado anteriormente, o modelo de rede é complicado, tomando um tempo considerável para sua compreensão e o fato de haver somente uma interface de rede para o ambiente de testes adicionou uma complexidade extra ao processo. A equipe de suporte da Rackspace teve que ser acionada para auxiliar o processo de configuração. A partir dai, o arquivo de configuração ideal para a implantação do ambiente de teste foi estabalecido como mostrado no Código 5. Nesta imagem pode-se observar no atributo *osop_networks* que as 3 redes utilizadas pelo *OpenStack* estão configuradoas sob a mesma faixa de IPs, ou seja, todas elas na mesma interface de rede.

4.1.3.3 Instalação do chef-client nos nós da cloud

Para a configuração do *chef-client* nos nós onde os componentes do *OpenStack* serão instalados, é necessária a criação de uma chave SSH para conexão sem senhas e depois proceder a instalação usando a ferramenta *knife bootstrap*, como visto no Código 6, substituindo <*usuario*> e <*ip*> pelo usuário e IP de cada nó que se deseja adicionar ao ambiente.

Após o processo terminar, é necessário que cada nó da rede possa acessar o *chef-server* pelo *hostname*. Para isto, adiciona-se uma linha para cada nó no arquivo */etc/hosts* seguindo o seguinte modelo do Código 7. A partir daí, o registro do nó cliente com o *chef-server* é feito executando-se comando *chef-client*.

4.1.3.4 Atribuição e aplicação dos papéis aos nós

Prossegue-se então com a atribuição dos papéis aos nós que farão parte da *cloud*. Como dito anteriormente, para o ambiente de teste foram usados um nó controlador, um de rede e um de processamento. São executados os comandos mostrados no Código 8 em cada nó, substituindo o <*papel*> por *role[ha-controller]*, *role[single-compute]* e *role[single-network]* e <*hostname*> pelo *hostname* deste mesmo nó.

Ao fim do procedimento, o *OpenStack* já deve estar acesível através do painel Horizon através do IP do nó controlador. A autentição pode ser feita com o usuário *admin* e a senha *secrete*. Estas credenciais podem ser modificadas a qualquer momento e outras mais também podem ser criadas.

Pode-se adicionar mais nós de rede e processamento de máquinas virtuais conforme necessário, repetindo os processos de preparação do nó a ser adicionado, instalação do *chef-client* (Código 6), configuração do arquivo de *hosts* (Código 7), atribuição do papel (Código 8) e execução do comando *chef-client* no próprio nó.

```
{
  "name": "rpcv412",
  "description": "Rackspace Private Cloud v4.2.2",
  "cookbook_versions": {
  },
  "json_class": "Chef::Environment",
  "chef_type": "environment",
  "default_attributes": {
  },
  "override_attributes": {
    "nova": {
      "libvirt": {
        "vncserver_listen": "0.0.0.0"
      },
      "network": {
        "provider": "neutron"
      }
    },
    "neutron": {
      "ovs": {
        "provider_networks": [
          {
            "label": "ph-eth0",
            "bridge": "br-eth0",
            "vlans": "1:1"
          }
        ],
        "network_type": "gre"
      }
    },
    "mysql": {
      "allow_remote_root": true,
      "root_network_acl": "%"
    },
    "osops_networks": {
      "nova": "192.168.0.0/24",
      "public": "192.168.0.0/24",
      "management": "192.168.0.0/24"
    }
  }
}
```

Código 5: Arquivo de configuração das variáveis utilizadas para o ambiente de testes montado

```
$ ssh-keygen
$ knife bootstrap -E private-cloud -i .ssh/id_rsa_private --sudo
-x <usuario> <ip>
```

Código 6: Código para realizar a configuração do *chef-client* nos nós do ambiente

```
<ip-chef-server> <hostname-chef-server>
```

Código 7: Modelo de arquivo de configuração de *hosts* nos nós clientes

```
$ knife node run_list add <hostname> '<papel>'
```

Código 8: Comando para atribuit um papel a um nó

5 Resultados e discussões

Após todos os passos serem seguidos e os servidores devidamente configurados, iniciaram-se os testes do *OpenStack* para avaliar sua facilidade de uso, performance e funcionalidades, colocando em julgamento também integração com ferramentas para maiores abstrações (PaaS e SaaS) e o ciclo de desenvolvimento do projeto.

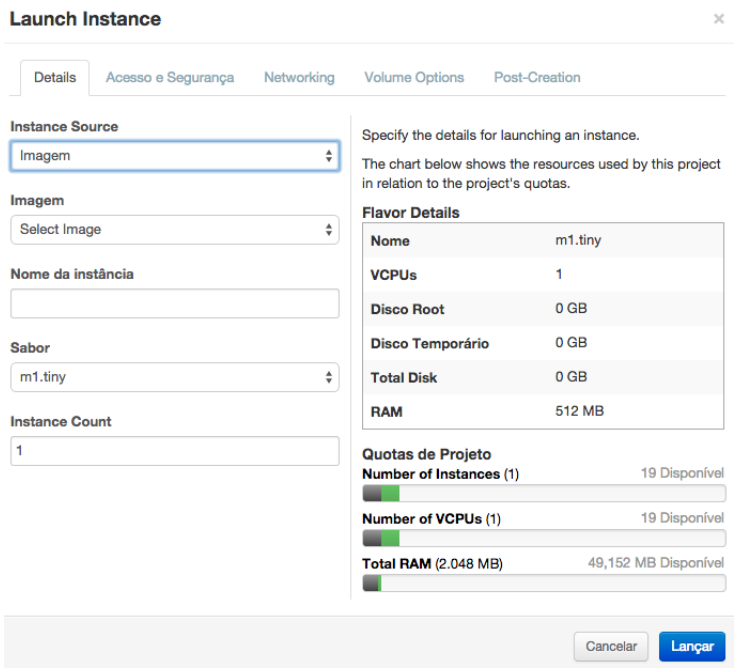

Figura 8: Detalhes da nova instância a ser criada

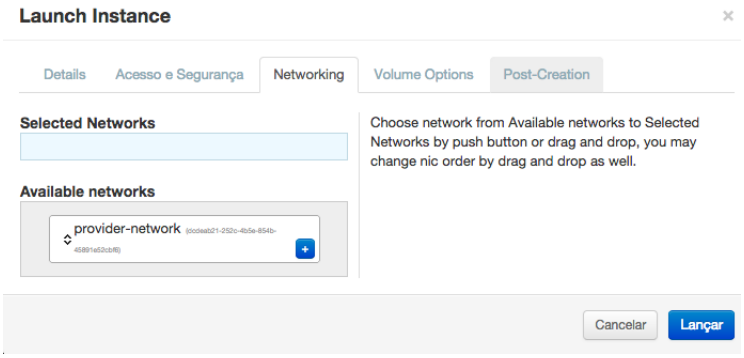

Figura 9: Configurações de rede de uma nova instância

A interface *web* para gerenciamento das máquinas virtuais é extremamente simples, desde a criação até a alteração dos recursos computacionais destas, atingindo as expectativas com relação a este quesito. É possível definir vários "sabores" de máquinas virtuais com configurações fixas de processador, armazenamento, memória e rede (vide Figuras 8 e 9). Estas máquinas são criadas baseadas em imagens enviadas para o sistema e podem ser ligadas e volumes persistentes de armazenamento para compartilhamento e consolidação (agrupamento) de dados, como mostra a Figura 10. Há também a possibilidade de fazer um *snapshot* (imagem) de um determinado servidor para posterior restauração ou duplicação, criando outro servidores a partir deste mesmo *snapshot* (ver Figura 11).

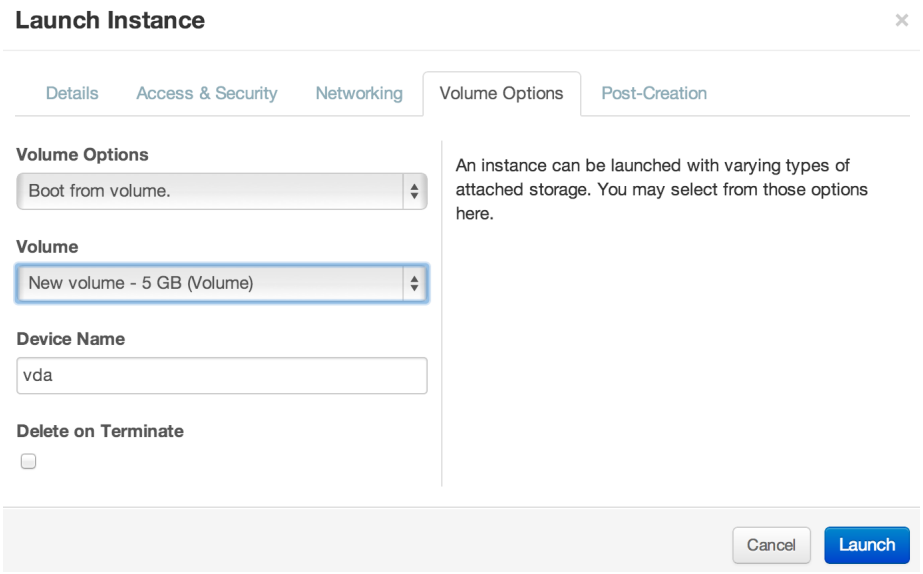

Figura 10: Configurações de volume da nova instância

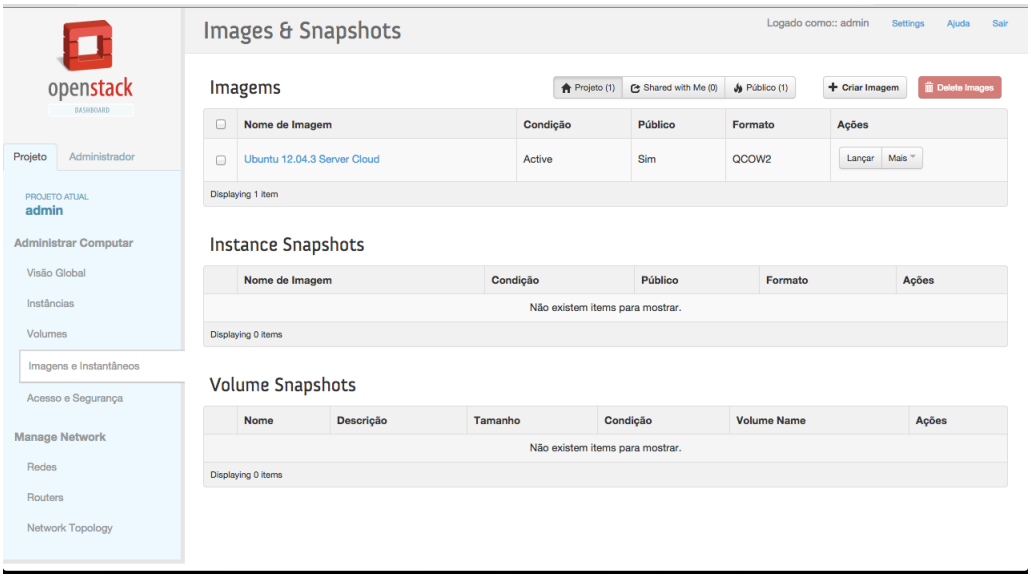

Figura 11: Interface para gerenciamento de volumes persistentes e snapshots

Existe também o controle de acesso e segurança das instância, onde são criados usuários para o sistema. Cada usuário pode ter uma chave de criptografia, que será usada para autenticá-lo com os servidores que tiver autorização de acesso e fornecer um canal seguro de comunicação entre eles, como poder ser visto na Figura 12. Na área mostrada na Figura 13 são configuradas quais portas de cada servidor que aceitarão conexões de entrada. Nesta seção, ilustrada pela Figura 14 é possível também visualizar todos os pontos de acesso dos serviços do *cluster*, assim como as credenciais para acesso a eles. Existe um *super admin* capaz de utilizar esta interface para criar *tenants*[1]. no *OpenStack*, permitindo que diversas organizações utilizem o sistema sem conhecimento e interferência entre si e com suas prórias regras individuais de segurança.

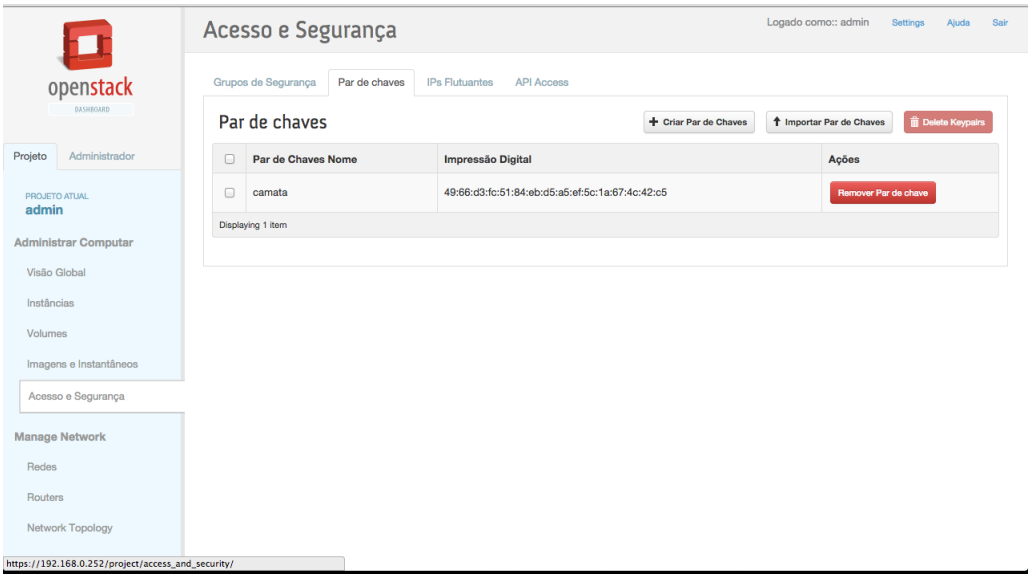

Figura 12: Interface para criação e visualização das chaves de acesso às máquinas virtuais

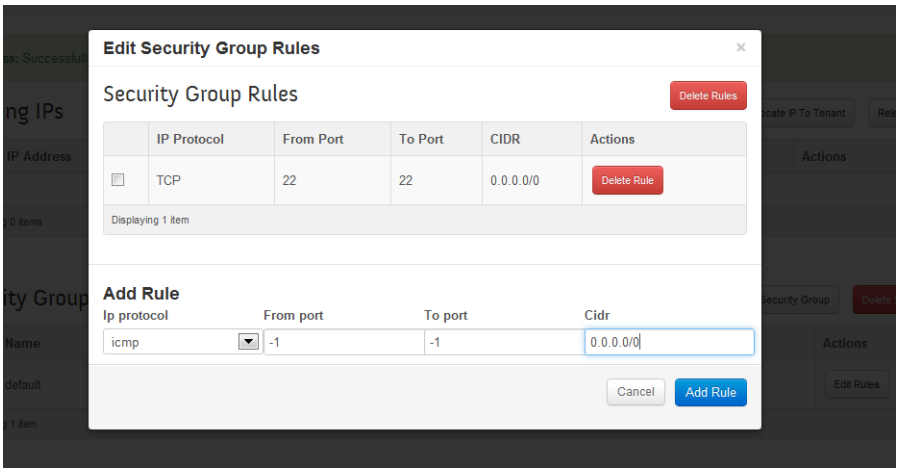

Figura 13: Configurações de permissões de rede de grupos

[1]conceito utilizado para isolar dados de diferentes clientes em um só sistema de uma maneira segura e eficiente

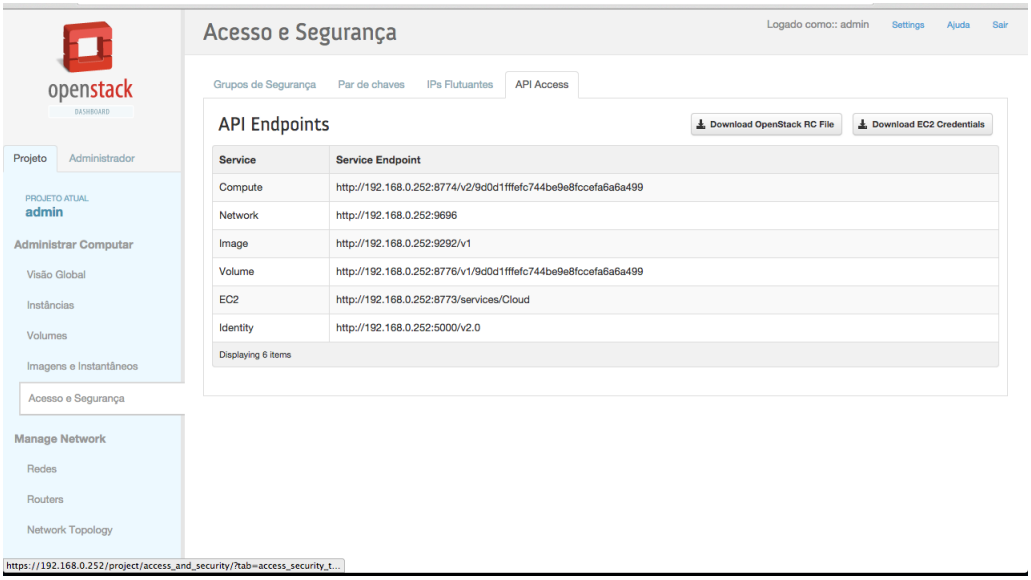

Figura 14: Visualização de *URLs* de serviços e credenciais de acesso

Durante o desenvolvimento deste experimento, não havia uma ferramenta dentro do *OpenStack* para alcançar o nível de *PaaS*: apenas ferramentas desenvolvidas por terceiros (como o Juju, da Canonical, e o Tsuru, da Globo.com) forneciam essa funcionalidade através das ricas APIs expostas pelo *OpenStack*. Contudo, antes mesmo do fim do experimento, foi desenvolvida e integrada uma ferramenta para esta funcionalidade no projeto, chamada de *Heat*, que não foi considerada nesta análise devido à falta de tempo hábil.

Através de todas estas características, uma conclusão é eminente: o processo de aquisição de infraestrutura computacional no paradigma de *cloud computing* é extremamente mais rápido e prático do que no modelo tradicional. Não bastante esta vantagen, cada usuário pode se servir e adquirir a quantidade de recursos necessários ao seu projeto e expandir ou diminuir conforme necessário (a interface mostrada na figura 8, é também utilizada para a edição das máquinas já criadas dentro do sistema. Além disso, a interface é extremamente simples, possibilitando que qualquer pesquisador ou aluno tenha acesso à infraestrutura que precisar para atingir seu objetivo, melhorando ensino, pesquisa e também a própria universidade. Chegamos então ao principal ponto econômico: graças a essa elasticidade, de uso e liberação dos recursos, o *cloud computing* oferece uma grande economia, evitando o desperdício através de recursos escassos e fragmentados, favorecendo ainda mais o melhoramento dos recursos computacionais da própria instituição.

6 Conclusões

Pode-se concluir através das análises da bibliografia, das tecnologias e das ferramentas apresentadas que *cloud computing* é extremamente eficaz em um ambiente acadêmico. Professores podem ter um rápido acesso a infraestrutura para poder executar e/ou disponibilizar seus projetos, e alunos para auxílio na instalação e configuração de ambientes para o aprendizado das mais diversas tecnologias disponíveis, sem toda a burocracia já conhecida das instituições públicas de ensino superior (um servidor pode ser entregue em questão de minutos, em vez de dias, ou até mais), Isto soluciona o problema alunos do curso de Ciência da Computação, que poderão instalar seus aplicativos em máquinas virtuais isoladas, sem interferir uns nos outros e ter que lidar com sistema operacional "congestionado" por ter diversos *softwares* diferentes instalados. Além disso, toda a arquitetura obtida desta maneira possui um grande potencial elástico e pode ser manipulada por uma interface web simples, intuitiva, poderosa e eficaz para que os recursos sejam redimensionados conforme a necessidade do usuário. É oferecida uma economia para a instituição de ensino, que através da distribuição dos servidores virtuais nos físicos e seu uso sob-demanda, tem um melhor proveito dos mesmos. Todas estas conclusões confirmam que os problemas encontradas pela UENF com a gerencia de seus recursos computacionais podem ser solucionados inteligentemente e oferecer vantagens para diversos alunos e professores da universidade.

É visto que a instalação e configuração do próprio *OpenStack* e das ferramentas que é compõe tem sua complexidade concentrada no modelo de rede: existem diversas ferramentas de automação que realizam todo processo de instalação e configuração de pacotes Linux nos servidores de maneira automática e bem transparente. A Rackspace oferece um ótimo suporte gratuito para auxiliar em algumas tarefas e tem um completo time experiente em grandes ambientes de *cloud computing* para consultorias e outros serviços pagos relacionados a escalabilidade e manutenção dos recursos computacionais necessários, o que é de suma importância.

Foi constado que a ferramenta está evoluindo em velocidade crescente, com cada vez

mais recursos sendo integrados e desenvolvidos pelos contribuidores do projeto. Somente durante o desenvolvimento deste trabalho, duas novas versões foram lançadas, ambas adicionando novas funcionalidades para suprir as necessidades dos seus usuários. Isto tudo é permitido graças ao desenvolvimento no modelo de software livre, onde todos podem contribuir com o projeto.

6.1 Trabalhos futuros

Este trabalho deixa inúmeros assuntos interessantes para trabalhos futuros, como uma análise de ferramentas capazes de levar a *cloud* para o nível de plataforma como serviço (usando ferramentas como as já mencionadas Juju e Tsuru). Tal serviço é de extrema utilidade para desenvolvedores de aplicações que não tem conhecimento de gerenciamento de servidores e torna esta tarefa extremamente simples, rápida e prazerosa. Também seria de grande contribuição uma análise de performance de diferentes *hypervisors*, elaboração de ambientes de alta disponibilidade de nós controladores, de processamento e de volumes.

Um importante trabalho futuro sequencial a este projeto está sendo elaborado no momento. Servidores e equipamentos de rede de alta performance foram adquiridos pela universidade através de um edital de apoio da FAPERJ para que a *cloud* seja disponibilizada para ser utilizada pelos próprios alunos e professores da UENF. O uso inicial deste equipamento abrange oferecer hospedagem e ambiente de testes para aplicações desenvolvidas pelos próprios alunos do curso de Ciência da Computação, através de bolsas de iniciação científica e tecnológica.

Referências

ALABBADI, M. M. Cloud computing for education and learning: Education and learning as a service (elaas). *Interactive Collaborative Learning (ICL), 2011 14th International Conference on*, p. 589–594, 2011.

ALONSO, G. et al. *Web services*. [S.l.]: Springer, 2004.

ANDRZEJAK, A.; ARLITT, M.; ROLIA, J. *Bounding the Resource Savings of Utility Computing Models*. [S.l.], 2002.

ARMBRUST, M. et al. A view of cloud computing. *Commun. ACM*, ACM, New York, NY, USA, v. 53, n. 4, p. 50–58, abr. 2010. ISSN 0001-0782. Disponível em: <http://doi.acm.org/10.1145/1721654.1721672>.

BO, W.; HONGYU, X. The application of cloud computing in education informatization. *Computer Science and Service System (CSSS), International Conference*, p. 2673–2676, 2011.

BROBERG, J.; VENUGOPAL, S.; BUYYA, R. Market-oriented grids and utility computing: The state-of-the-art and future directions. *Journal of Grid Computing*, v. 6, n. 3, p. 255–276, 2007.

CARISSIMI, A. Virtualização: da teoria a soluções. *26º Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos*, 2008.

CHAUDHARY, A. S.; SALEEM, M. A.; BUKHARI, H. Z. Web services in distributed applications advantages and problems. *IEE conference at Ghulam Ishaq Khan Institute of Engineering Sciences and Technology*, 2002.

DIGNAN, L. *Rackspace, NASA launch OpenStack: Can it prevent cloud lock-in?* julho 2014. Disponível em: <http://www.zdnet.com/blog/btl/rackspace-nasa-launch-openstack-can-it-prevent-cloud-lock-in/36850>.

EKWE, S. Cloud computing: Educational institutions in developing regions. *11th Research Seminar Series Workshop*, p. 38–42, 2012.

FIGUEIREDO, R. J.; DINDA, P. A.; FORTES, J. A. B. A case for grid computing on virtual machines. *23rd International Conference on Distributed Computing Systems*, 2003.

FOSTER, I. What is the grid? a three point checklist. *GRIDtoday*, v. 6, n. 6, 2002.

FOSTER, I.; KESSELMAN, C. *The grid: blueprint for a new computing infrastructure*. Morgan Kaufmann Publishers, 1999. (Advanced computing. Computer systems design). ISBN 9781558604759. Disponível em: <http://books.google.com.br-/books?id=ONRQAAAAMAAJ>.

FOSTER, I. et al. Cloud computing and grid computing 360-degree compared. *Grid Computing Environments Workshop*, 2008.

FOX, A. *Cloud computing in education*. março 2009. Disponível em: <https://inews-.berkeley.edu/articles/Spring2009/cloud-computing>.

GREENGARD, S. Cloud computing and developing nations. *Commun. ACM*, ACM, New York, NY, USA, v. 53, n. 5, p. 18–20, maio 2010. ISSN 0001-0782. Disponível em: <http://doi.acm.org/10.1145/1735223.1735232>.

KUNDRA, V. *Federal Cloud Computing Strategy*. [S.l.], 2011.

LAUREANO, M. A. P.; MAZIERO, C. A. Virtualização: Conceitos e aplicações em segurança. *Oitavo Simpósio Brasileiro de Segurança da Informação e Sistemas Computacionais*, 2008.

MELL, P.; GRANCE, T. *The NIST Defnition of Cloud Computing*. [S.l.], 2011.

PURCELL, A. *Tim Bell on the importance of OpenStack for CERN*. julho 2014. Disponível em: <http://www.isgtw.org/feature/tim-bell-importance-openstack-cern>.

RADEMAKERS, L.; COLEMAN, D.; REINY, S. *Web Solutions Inspire Cloud Computing Software*. [S.l.], 2012.

RIDLEY, M. et al. *The impact of cloud*. [S.l.], 2014.

SRIDHAR, T. *Layer 2 and Layer 3 Switch Evolution*. 1998. Disponível em: <http://www-.cisco.com/web/about/ac123/ac147/archived_issues/ipj_1-2/switch_evolution.html>.

SULTAN, N. Cloud computing for education: A new dawn? *International Journal of Information Management*, v. 30, p. 109–116, 2010.

SYLVIA, M.; PETERSON, B.; UNIACK, S. *Success in the cloud: why workload matters*. [S.l.], julho 2013.

TRUONG, H.-L. et al. Cloud computing for education and research in developing countries. In: _____. [S.l.]: IGI Global, 2012. cap. 5, p. 64–80.