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

EDUARDO DE OLIVEIRA ALBERTINI

DESENVOLVIMENTO DE UM SISTEMA WEB PARA USUÁRIOS SOLICITAREM INSTÂNCIAS DE CLUSTER EM NUVEM

TRABALHO DE CONCLUSÃO DE CURSO

CAMPO MOURÃO

2014

EDUARDO DE OLIVEIRA ALBERTINI

DESENVOLVIMENTO DE UM SISTEMA WEB PARA USUÁRIOS SOLICITAREM INSTÂNCIAS DE CLUSTER EM NUVEM

Trabalho de Conclusão de Curso apresentado ao Curso de Graduação em Tecnologia em Sistemas para Internet da Universidade Tecnológica Federal do Paraná como requisito para obtenção do grau de Tecnólogo em Tecnologia em Sistemas para Internet.

Orientador: Prof. Me. Rodrigo Hübner

Co-orientador: Prof. Me. Alessandro Kraemer

CAMPO MOURÃO

2014

RESUMO

ALBERTINI, Eduardo de Oliveira. DESENVOLVIMENTO DE UM SISTEMA WEB PARA USUÁRIOS SOLICITAREM INSTÂNCIAS DE CLUSTER EM NUVEM. 32 f. Trabalho de Conclusão de Curso – Curso de Graduação em Tecnologia em Sistemas para Internet, Universidade Tecnológica Federal do Paraná. Campo Mourão, 2014.

Diversos estudos científicos atuais utilizam nuvens computacionais para executar aplicações de diferentes domínios. Tais aplicações podem ser executadas diversas vezes com diferentes entradas e configurações tanto de hardware como de software. Esse trabalho realizado pelo desenvolvedor pode gerar uma grande demanda de tempo, pois pode ser que em cada experimento seja necessário uma reconfiguração e gerência de diversas instâncias de *clusters* virtuais em uma plataforma de nuvem computacional. Com o objetivo de reduzir o tempo gasto e facilitar a configuração e gerenciamento das instâncias nos *clusters* virtuais esse trabalho apresenta o desenvolvimento de uma interface gráfica complementar ao `Sunstone` para facilitar e viabilizar o serviço de instanciar e configurar *clusters* virtuais para a execução de aplicações de diversos domínios.

Palavras-chave: OpenNebula, Sunstone, Interface Gráfica, Computação em Nuvem

ABSTRACT

ALBERTINI, Eduardo de Oliveira. DEVELOPMENT OF A WEB SYSTEM FOR USERS TO REQUEST INSTANCES OF CLUSTER IN CLOUD. 32 f. Trabalho de Conclusão de Curso – Curso de Graduação em Tecnologia em Sistemas para Internet, Universidade Tecnológica Federal do Paraná. Campo Mourão, 2014.

Several current scientific studies utilize cloud computing to run various applications domains. Such applications can be run several times with different inputs and configurations of both hardware and software. This work performed by the developer can generate a large demand of time, it may be that in each experiment a reconfiguration and management of multiple instances of virtual clusters on a cloud computing platform is needed. Aiming to reduce the time spent and facilitate the configuration and management of virtual clusters instances in this paper presents the development of a graphical interface to complement Sunstone to facilitate and enable the service to instantiate and configure virtual clusters for executing various applications domains.

Keywords: OpenNebula, Sunstone, Graphic Interface, Cloud Computing

LISTA DE FIGURAS

SUMÁRIO

1 INTRODUÇÃO

Devido à crescente evolução tecnológica, muitas instituições começaram a usar a computação em nuvem em seu cotidiano, seja para armazenamento ou para processamento de dados. O uso da computação em nuvem normalmente é feito por meio de aplicações web, onde usuários administradores solicitam instâncias de máquinas virtuais e softwares. Porém, para que a solicitação de certos recursos computacionais se torne uma tarefa objetiva, é necessário realizar otimizações nas interfaces dos usuários. Muitas vezes os usuários administradores têm dificuldades em utilizar as interfaces padrões em uma plataforma de nuvem. Essa dificuldade é dada pela ampla variedade de opções e situações que podem ser montadas.

O objetivo deste trabalho é desenvolver um sistema que seja especializado na instância de *clusters* virtuais, ou seja, na instância de várias máquinas virtuais que juntas formam um *cluster* dentro da plataforma de nuvem, satisfazendo assim uma das demandas de projetos científicos da Universidade Tecnológica Federal do Paraná - Campus Campo Mourão (UTFPR-CM). Para isso, utilizamos o modelo de máquina virtual implantado na nuvem da UTFPR-CM, apresentado por Kraemer et al. (2013). Os sistemas atuais que estão integrados na plataforma da UTFPR-CM não oferecem soluções específicas para certas necessidades dos usuários, como a própria montagem do *cluster* virtual. A interface atual é complexa para usuários iniciantes, desencorajando sua utilizção. A primeira etapa para realização do trabalho de desenvolvimento de uma interface amigável consiste em compreender os mecanismos da nuvem. Nesse sentido, são abordados inicialmente o que é uma nuvem e o que dá forma à ela. Compreender seus mecanismos é importante porque a interface será fortemente acoplada à nuvem.

A computação em nuvem foi desenvolvida com o objetivo de fornecer serviços, baseados na *utility computing*, na qual se paga somente pelo serviço utilizado. Uma nuvem pode ser classificada de acordo com sua finalidade, podem ser: nuvem privada, nuvem pública, nuvem híbrida ou nuvem comunitária. A nuvem privada é utilizada por apenas uma instituição, para suas diversas aplicações. A nuvem pública atende o público em geral, desde simples usuários domésticos até a demanda de grandes empresas. Na nuvem híbrida existem duas ou mais nuvens interligadas, na qual o produto final é visto como sendo um só. Na modalidade comunitária

a nuvem pode ser utilizada por diversas organizações que possuem um objetivo em comum. Podemos dividir a nuvem ainda conforme os tipos de serviço que são disponibilizados por elas, sendo eles Software como Serviço, Plataforma como Serviço e Infraestrutura com Serviço. No Software como Serviço existem diversos softwares que são implantados na nuvem e podem ser acessados via web. O usuário solicita o software e utiliza-o sem se preocupar com instalação. Por outro lado, na Plataforma como Serviço são implantadas várias Interfaces de Programação de Aplicações (APIs) com linguagens de programação e ferramentas para o desenvolvimento de aplicativos. O usuário pode alterar configurações com mais controle sobre o ambiente virtualizado. Por fim, a Infraestrutura como Serviço é a base para as outras camadas, citadas acima, sendo responsável por fornecer e gerenciar recursos de hardware e armazenamento.

Os computadores que formam o *cluster* físico da nuvem conseguem virtualizar recursos (hardware e rede de comunicação) por meio dos softwares chamados *Hypervisors*. Um *Hypervisor* faz comunicação entre a máquina física e a máquina virtual. Essa modalidade de software também é conhecida como Gerenciador de Máquinas Virtuais. Entretanto, um *cluster* físico e um *Hypervisor* implantado, não caracterizam uma plataforma de nuvem. Uma plataforma de nuvem adiciona sistemas de gerenciamento e monitoramento de hardware, assim como formas avançadas de controlar máquinas virtuais instanciadas ao longo do *cluster* físico. O software com esses papéis é conhecido como Gerenciador de Recursos da Nuvem.

Nas plataformas de nuvem o usuário não tem total controle de onde serão instanciadas as máquinas virtuais ao longo das máquinas físicas. Por outro lado, em pesquisas científicas é importante estabelecer a localização exata das instâncias, pois a distribuição em diferentes combinações de rede afeta no desempenho dos algoritmos (KRAEMER et al., 2013). Levando em consideração essa abordagem, surge o seguinte questionamento: Como oferecer o controle total da alocação de máquinas virtuais dentro da nuvem por meio de uma interface amigável?

Este trabalho de conclusão de curso está organizado como segue. No capítulo 2 é apresentada a metodologia necessária para realizar os objetivos deste trabalho. No capítulo 3 é apresentado o Referencial Teórico abordando conceitos e aplicações da computação em nuvem, assim como seus principais componentes arquiteturais. No capítulo 4 apresentamos o sistema desenvolvido para facilitar a instância de clusters virtuais na nuvem privada da UTFPR-CM. Por fim, no capítulo 5 são apresentadas as conclusões e trabalhos futuros.

2 METODOLOGIA E TECNOLOGIAS UTILIZADAS

A compreensão de Computação em Nuvem ocorreu por meio de pesquisas em livros e artigos, com preferência nos artigos completos e disponíveis na ACM[1], IEEE[2] e Scopus[3].

A descrição do *cluster* virtual e de seus principais componentes formadores tem como base o trabalho de Kraemer et al (2013), assim como leituras de publicações nas bases relacionadas. Informações complementares também foram obtidas por meio da equipe técnica que implantou a nuvem na UTFPR-CM e pelo site `cloud4university.org`.

A plataforma de nuvem implantada na UTFPR-CM, observada na Figura 1, é o ambiente onde o novo sistema facilitador de instâncias será utilizado.

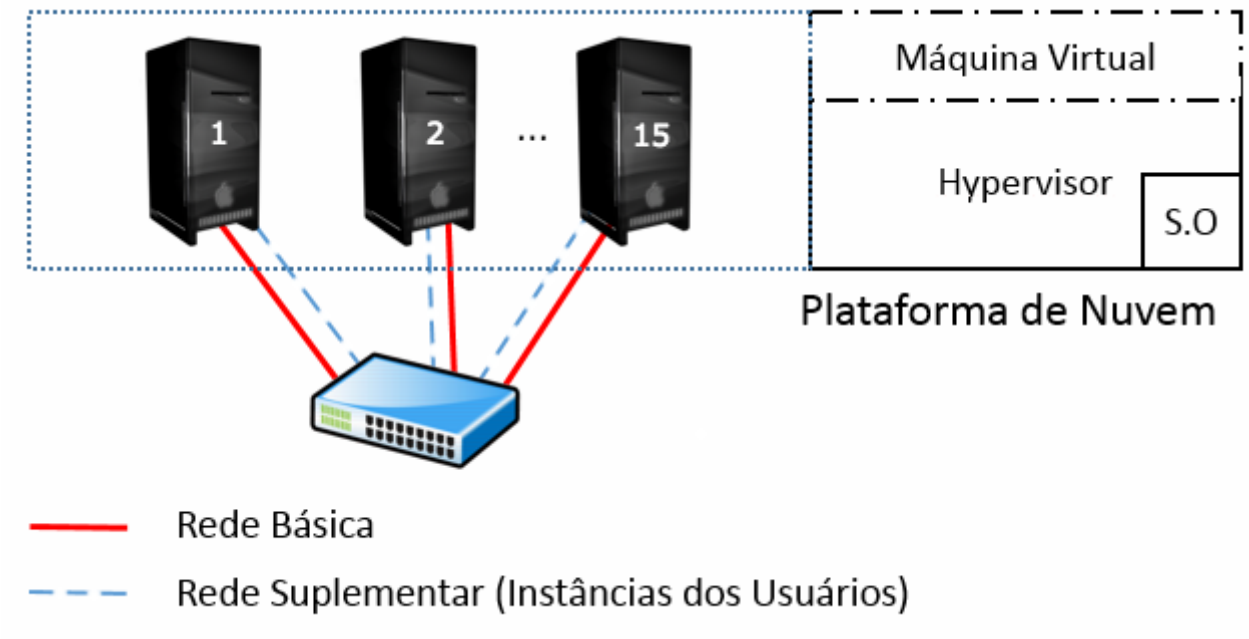

Figura 1: Infraestrutura da Nuvem da UTFPR-CM.

Fonte: Adaptação de Cloud4University

[1] Association for Computing Machinery. Disponível em <http://www.acm.org/>
[2] Instituto de Engenheiros Eletricistas e Eletrônicos. Disponível em <http://ieeexplore.ieee.org/>
[3] Elsevier Scopus. Disponível em <http://www.scopus.com/home.url>

A Figura 1 mostra uma infraestrutura com 15 servidores Intel QuadCore com 4 GB de RAM cada e 2 servidores QuadCore Xeon IBM com 8 GB de RAM cada. O Gerenciador de nuvem é o `OpenNebula`[4].

Utilizamos o Servidor TomCat Apache[5] para o desenvolvimento do sistema especializado em instanciar *cluster* virtual devido a fácil integração do mesmo com a linguagem de programação Java, que foi escolhida por ser de melhor domínio do desenvolvedor do sistema. O servidor está instalado no computador mestre da plataforma de nuvem.

Depois de identificar os parâmetros da aplicação por meio de entrevistas com pesquisadores da UTFPR-CM, identificamos os comandos necessários para interagir com o Gerenciador da Nuvem, de forma a instruí-lo sobre a distribuição das máquinas virtuais ao longo do *cluster* físico. Depois da descoberta dos requisitos, elaboramos um diagrama de fluxo contendo todos os processos que devem ser implementados. Resumidamente, o sistema web desenvolvido consiste, em alto nível, em capturar os dados inseridos pelo usuário e repassá-los ao Gerenciador da Nuvem por meio de comandos remotos no terminal.

[4]OpenNebula - Flexible Enterprise Cloud Made Simple. Disponível em <http://www.opennebula.org/>
[5]Apache Server - The Apache Software Foundation. Disponível em <http://www.apache.org/>

3 REFERENCIAL TEÓRICO

O presente capítulo apresenta os conceitos e aplicações da computação em nuvem. Na sequência serão abordados os tipos de nuvens e os serviços disponíveis. A seguir será apresentado as características de uma plataforma de computação em nuvem e por fim, os conceitos sobre *cluster* virtual.

3.1 CONCEITOS E APLICAÇÕES DE COMPUTAÇÃO EM NUVEM

A Enciclopédia Britannica (2013) diz que a expressão descende da prática de utilizar desenhos de nuvens para representar a infraestrutura dos sistemas de computação e telecomunicação. Sousa et al. (2010) definem a nuvem como uma abstração que oculta a complexidade da infraestrutura. Cada parte da infraestrutura possui um serviço a ser oferecido, e esses geralmente são colocados em *Data Centers*, utilizando o compartilhamento de hardware para processamento e armazenamento.

Uma infraestrutura de nuvem é composta de várias máquinas físicas conectadas por meio de uma rede. Cada máquina possui configurações de software compatíveis com as demais máquinas que formam o cenário da nuvem. Também podem haver diferentes especificações de hardware, por exemplo, máquinas que possuem diferentes quantidades e tipos de processadores, quantidade de memória RAM e até mesmo quantidade de armazenamento em discos diferentes. Apesar das diferenças de hardware, dentro da nuvem podem haver várias máquinas virtuais (MVs) em execução. A plataforma de nuvem, composta por vários subsistemas, é que permite a virtualização sob diferentes hardwares. A quantidade de MVs em execução depende da capacidade do hardware da máquina real (SOUSA et al., 2010). Complementarmente, cada plataforma de nuvem pode gerenciar a rede e os recursos de virtualização usando diferentes mecanísmos.

Sousa et al. (2009) dizem que para acessar os serviços disponíveis na nuvem os usuários necessitam ter apenas alguns softwares, que por sua vez interagem com a nuvem. Esses softwares podem ser Sistemas Operacionais, navegadores web ou APIs de integração. Todos os

recursos estão acessíveis na nuvem por meio da Internet e os usuários não necessitam de uma máquina com recursos sofisticados, o que leva à diminuição do custo de aquisição de hardware pelos usuários.

A computação em nuvem foi desenvolvida com a meta de fornecer serviços de baixo custo, fácil acesso e com garantias de disponibilidade e escalabilidade. A computação em nuvem é uma evolução dos serviços e tecnologias sob demanda, também conhecida como *Utility Computing*. O objetivo dela é fornecer recursos computacionais como armazenamento e processamento através de provedores e cobrar um valor por isso. Uma vantagem deste tipo de serviço é que o valor do serviço é cobrado apenas pela quantidade de uso (SOUSA et al., 2009).

Computação em nuvem é uma maneira bastante eficiente de maximizar a utilização dos recursos computacionais. Um exemplo disso é a capacidade que um processador físico tem de conseguir atender diferentes quantidades de processadores virtualizados por outro software, o Gerenciador de Recursos Virtuais. Nesse sentido ocorre a maximização, ou ampliação, mesmo que virtual dos recursos disponíveis. No ambiente de plataformas de computação em nuvem podem ser aplicadas técnicas de tolerância a falhas que permitem a continuidade do funcionamento de alguns recursos caso um ou mais desses apresente defeito (TAURION, 2009).

A partir de meados de 2008, a computação em nuvem se tornou conhecida pelo mundo todo e foi amplamente adotada para facilitar a utilização de alguns serviços, como os repositórios de dados e serviços web. Alguns exemplos desses serviços são: `iCloud`[1], `Google Drive`[2], `Dropbox`[3], `OneNote`[4], entre outros. Os serviços web implantados em nuvem são utilizados para facilitar e ajudar o desenvolvedor a criar e gerenciar suas aplicações. Um exemplo desse tipo de serviço é o `Amazon Web Service` (AWS).

Além da computação em nuvem ser utilizada como repositório de dados e como provedor de serviços web, ela pode ser utilizada também em meios acadêmicos. A Universidade da Califórnia (US) em Berkeley, com o apoio da Amazon Web Services, utilizou a computação em nuvem em um de seus cursos para focar no desenvolvimento de aplicações SaaS (Software como Serviço). Outro exemplo é o da Faculdade de Medicina de Wisconsin Biotecnologia e do Centro de Bioengenharia em Milwaukee que utilizou a computação em nuvem para fazer pesquisas sobre proteínas e torná-las mais acessíveis para cientistas do mundo todo. Há também o caso da Faculdade de Engenharia Elétrica e Ciência da Computação (EECS) da Universidade Estadual de Washington, que utiliza a nuvem por causa da economia na aquisição de hardware,

[1] Apple iCloud. Disponível em <https://www.icloud.com/>
[2] Google Drive. Disponível em <https://drive.google.com/>
[3] Dropbox. Disponível em <https://www.dropbox.com/>
[4] Microsoft OneNote. Disponível em <http://http://www.onenote.com/>

devido a cortes orçamentais (SULTAN, 2010).

3.1.1 TIPOS DE NUVEM

De acordo com a finalidade das plataformas de nuvem, Sousa et al. (2009) classificam em três tipos: nuvem privada, nuvem pública e nuvem híbrida. Alguns autores incluem o tipo de nuvem comunitária. Essas divisões foram feitas de acordo com os tipos de acesso, serviços e restrições de quem acessa a nuvem. A Figura 2 apresenta uma arquitetura geral sobre esses tipos.

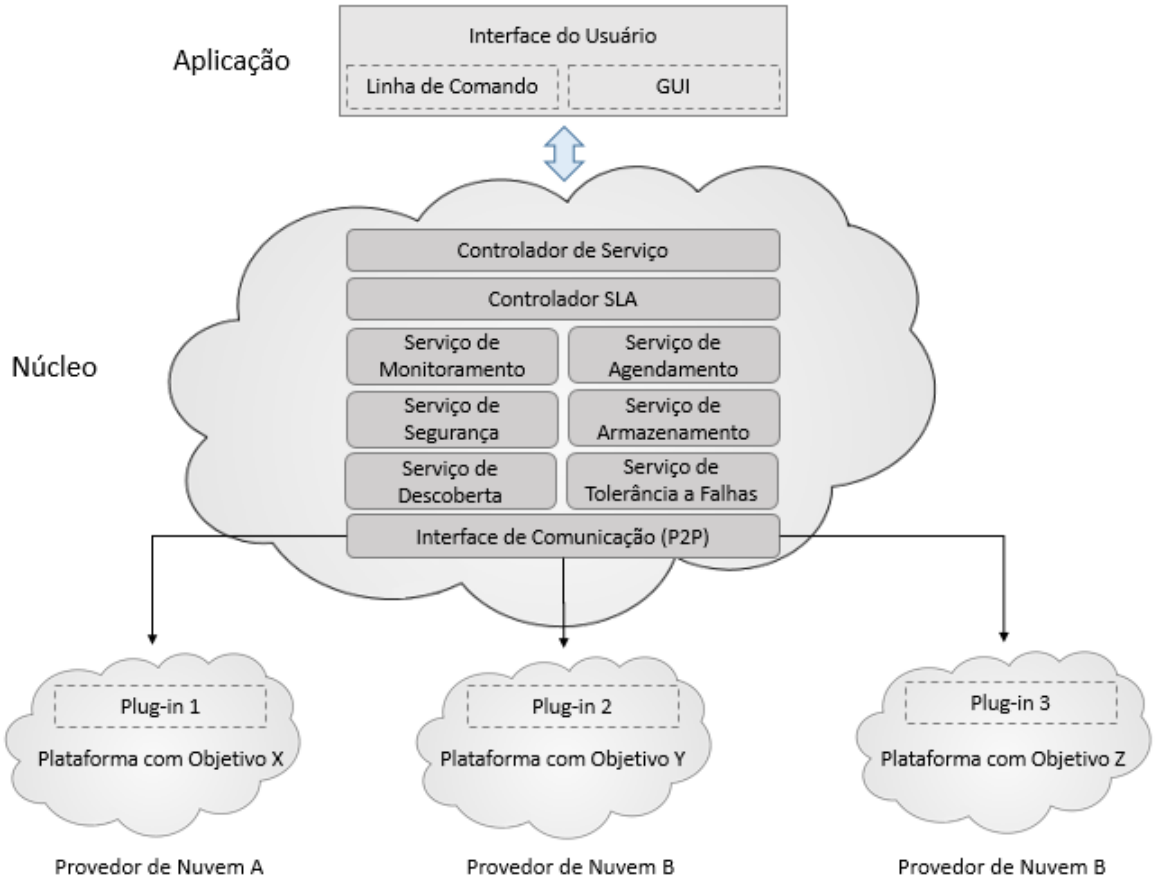

Figura 2: Arquitetura genérica com nuvens integradas.
Fonte: Adaptação de Saldanha et al. (2012, p. 5)

A Interface do Usuário é o ponto de partida para que a nuvem instancie os recursos desejados. Os usuários podem utilizar diversos tipos de interface. Por exemplo, por meio de linha de comando, formulários web, etc. O núcleo da nuvem, normalmente um computador dito como mestre, recebe as solicitações de tarefas dos usuários, verifica o contrato do cliente e aciona os demais serviços para fazer a instanciação dos recursos. Esses serviços dentro da nuvem

são apresentados na Seção 3.2. Na mesma Figura 2, podemos observar que vários provedores de nuvem podem estar integrados, formando assim uma grande rede de comunicação e serviços que inclui diferentes tipos de plataformas.

No tipo de nuvem privada, a infraestrutura da nuvem é utilizada somente por uma organização, sendo que esta pode ser uma nuvem local ou remota. A administração é feita pela própria empresa ou por empresas terceirizadas (MELL; GRACE, 2013). A Figura 3 apresenta um pacote de serviços chamado de cCloud[5], que é formado por várias nuvens privadas, caracterizando-se assim como uma grande Nuvem. Nesse exemplo há uma sequência de interações entre usuários, hardware e serviços disponíveis.

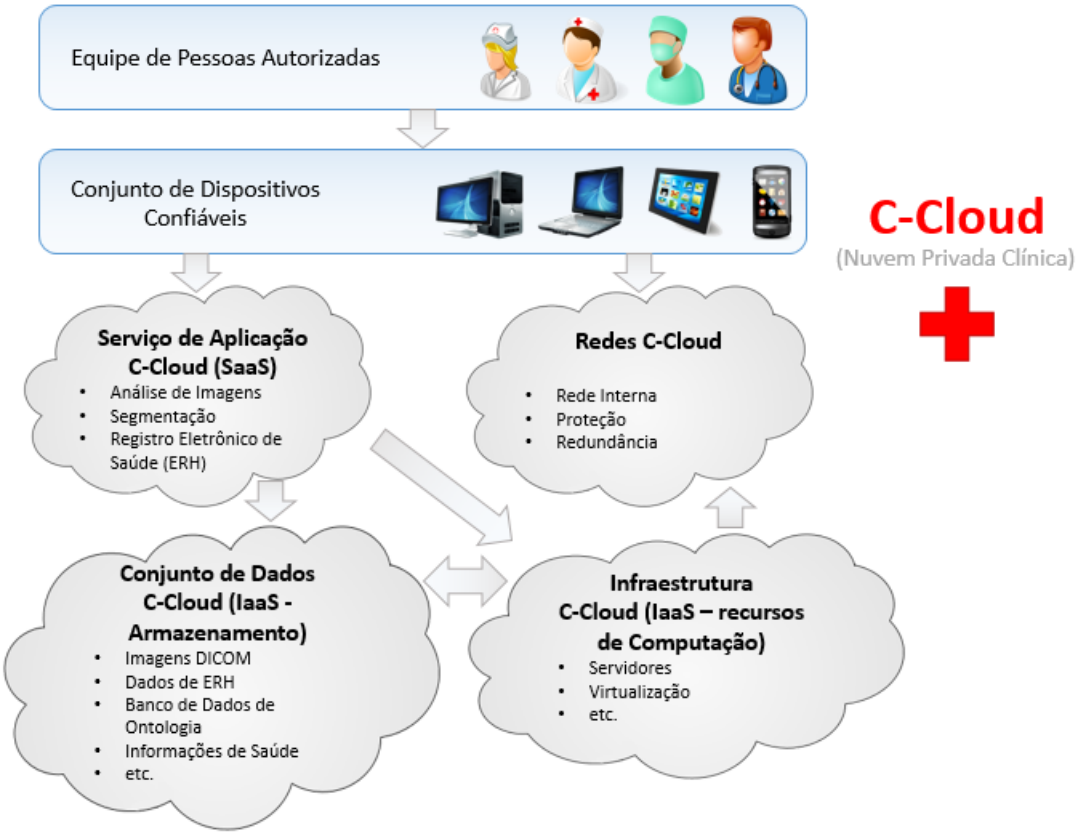

Figura 3: Arquitetura de uma nuvem privada.

Fonte: Adaptação de Schoenhagen et al. (2013, p. 10)

Ainda na Figura 3 é possível observar a divisão da nuvem para que cada segmento possa prover um serviço diferente para os usuários, por exemplo, uma parte responsável por disponibilizar serviços de rede, outra parte disponibilizando serviços de processamento, outra disponibilizando o armazenamento de resultado de exames. Tudo isso é solicitado através de alguns dispositivos que fazem interface com o usuário. Esse tipo de nuvem pode ser utilizado

[5]cClould. Disponível em <https://www.ccloud.com>

por uma rede de hospitais, na qual eles podem compartilhar os dados de exames, independentes da localização real de cada hospital.

Na nuvem pública a infraestrutura é disponibilizada para o público em geral. Nesse tipo de nuvem o usuário pode ser cobrado pelas solicitações (exemplo, Amazon) ou não (exemplo, Google Docs). Independente do porte das empresas, todas podem usar a nuvem para resolverem seus problemas, como servidor de aplicações, repositório de dados, etc. Estudantes e usuários domésticos também podem utilizar a nuvem pública. Contudo, esse tipo de nuvem é de uso geral e para uma ampla diversidade de público.

Na nuvem comunitária há um compartilhamento por diversas instituições, sendo que essas instituições possuem um interesse em comum, como a missão, requisitos de segurança, entre outros (MELL; GRACE, 2013).

Por fim, na nuvem híbrida há duas ou mais nuvens que podem ser públicas, privadas ou comunitárias ligadas de alguma forma tecnológica. Essas nuvens integradas são vistas como apenas uma grande nuvem. Cada nuvem é responsável por um serviço diferente das outras e assim se completam para prover todos os serviços necessários. Um exemplo de infraestrutura de nuvem híbrida foi apresentado na Figura 2, onde há várias nuvens que são ligadas por um *plugin* de comunicação (SOUSA et al., 2009).

3.2 TIPOS DE SERVIÇO

Atualmente, os serviços de computação em nuvem podem ser divididos em vários tipos, entre eles Infraestrutura como Serviço (IaaS), Software como Serviço (SaaS), Plataforma como Serviço (PaaS), Desenvolvimento de Software como Serviço (DevaaS), Comunicação como Serviço (CaaS), Email como Serviço (EaaS), Rede como Serviço (NaaS), de acordo com a oferta de serviços. Mas diversos autores abordam apenas três como sendo os principais modelos (SaaS, PaaS e IaaS). A Figura 4 apresenta alguns setores que tem relacionamento com esses modelos.

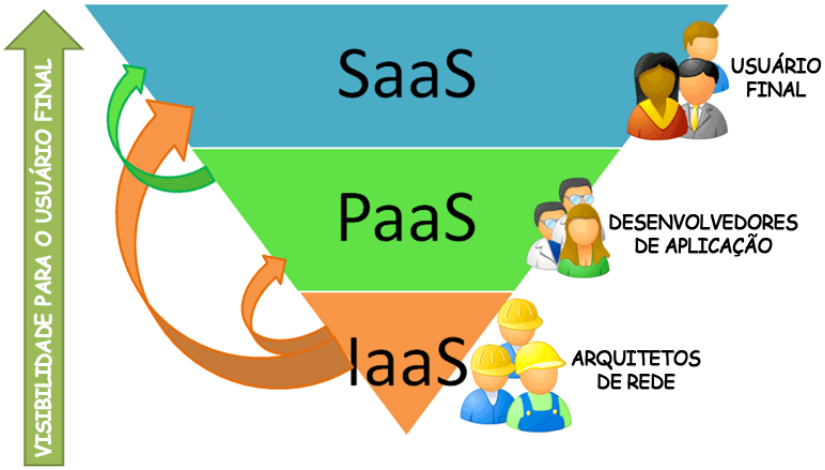

Figura 4: Relacionanento entre diversas áreas de atuação e os tipos de serviço da nuvem.
Fonte: Meriat (2014)

3.2.1 SOFTWARE COMO SERVIÇO (SAAS)

No modelo SaaS as aplicações são entregues para os usuários por meio da Internet, proporcionando sistemas de software com propósitos específicos. Não é necessário instalação e manutenção de software, basta acessá-lo por meio de um navegador web. Com isso, o usuário livra-se de um software muito complexo e do gerenciamento de hardware (SULTAN, 2010). Ou seja, o usuário não administra e nem controla a infraestrutura, apenas solicita configurações de hardwares e softwares que devem ser implantados. Algumas vezes, a própria plataforma oferece soluções prontas. Assim, os desenvolvedores podem se concentrar apenas na inovação e não na infraestrutura, o que proporciona o desenvolvimento rápido de sistemas de software.

Por ser um serviço web, ele pode ser acessado a qualquer momento e de qualquer lugar, necessitando apenas da Internet, oferecendo uma maior integração entre os serviços de software. Alguns exemplos de SaaS são os serviços de *Customer Relationship Management* (CRM) da Salesforce[6] e o Google Docs[7] (SOUSA et al., 2010).

3.2.2 PLATAFORMA COMO SERVIÇO (PAAS)

O modelo PaaS oferece uma infraestrutura de alto nível de integração para implementar e testar aplicações na nuvem (SOUSA et al., 2010). Para entender esse modelo é necessário lembrar do modelo de computação tradicional, onde cada aplicativo gerenciado localmente requer hardware, um sistema operacional, banco de dados, servidores web e outros serviços (SUL-

[6]Salesforce. Disponível em <http://www.salesforce.com/br/>
[7]Google Docs. Disponívem em <https://docs.google.com/>

TAN, 2010). No PaaS o usuário não administra ou controla a infraestrutura, mas tem controle sobre as aplicações implantadas e, possivelmente, as configurações das aplicações hospedadas nessa infraestrutura (SOUSA et al., 2010). Também podemos tratar o PaaS como uma plataforma de desenvolvimento e implantação de um conjunto de APIs, bibliotecas, linguagens de programação e ferramentas associadas utilizadas para a criação de aplicativos (VORAS et al., 2011).

No PaaS os desenvolvedores dispõem de ambientes escaláveis, porém, há algumas limitações que o ambiente impõe desde a concepção das aplicações até a utilização de sistemas de gerenciamento de banco de dados (SGBDs). O PaaS permite que usuários se inscrevam para solicitações de serviços de TI ou para resoluções de problemas pela web. *PaaS Google App Engine* e Aneka são exemplos dessa plataforma (SOUSA et al., 2010).

3.2.3 INFRAESTRUTURA COMO SERVIÇO (IAAS)

O IaaS é útil como camada base para outros modelos de serviço, como o PaaS e o SaaS (DAWOUD et al., 2010). O IaaS é responsável por fornecer toda a infraestrutura necessária para o PaaS e o SaaS, tendo como principal objetivo tornar mais fácil e acessível o fornecimento de recursos, tais como servidores, rede, armazenamento e também recursos para construir ambientes sob demanda, que podem incluir sistemas operacionais e aplicativos, dos quais o PaaS e o SaaS não se preocupam em gerenciar. Algumas características relevantes do IaaS são a interface única para administração da infraestrutura, uma API para interação com os *hosts*, *switches*, balanceadores, roteadores e o suporte para a adição de novos equipamentos de forma simples e transparente. No IaaS os usuários possuem controle sobre os sistemas operacionais, armazenamento e aplicativos implantados (SOUSA et al., 2010).

No IaaS a infraestrutura pode se modificar dinamicamente, aumentando ou diminuindo os recursos de acordo com as necessidades das aplicações. Isso é possível porque a infraestrutura é baseada em técnicas de virtualização de recursos de computação. Podemos mencionar como exemplos de IaaS o *Amazon Elastic Cloud Computing* (EC2) e o *Elastic Utility Computing Architecture Linking Your Programs To Useful Systems* (Eucalyptus) (SOUSA et al., 2010).

3.3 CARACTERIZAÇÃO DE UMA PLATAFORMA PARA COMPUTAÇÃO EM NUVEM

A computação em nuvem possui algumas plataformas de código livre, entre elas podemos citar o `Eucalyptus`[8], o `OpenNebula`[9], o `Nimbus`[10] e o `OpenStack`[11]. Neste trabalho utilizaremos especificamente o `OpenNebula`, pois o mesmo já está instalado como gerenciador da nuvem da UTFPR-CM. Existem também plataformas privadas, como o *Elastic Compute Cloud* (EC2) e o *Simple Storage Service* (S3), ambas da Amazon[12]. Como não temos acesso às plataformas privadas proprietárias, podemos apenas destacar os componentes que as plataformas de código livre tem em comum, entre eles, o *cluster* de máquinas físicas, o virtualizador dos recursos computacionais e o gerenciador de recursos da nuvem, explanados a seguir.

Por muito tempo os supercomputadores foram líderes no campo da computação, porém, com os problemas enfrentados na área da ciência, engenharia e negócios, devido as suas limitações de desempenho e escalabilidade, eles foram substituídos por *clusters*. Um *cluster* é um conjunto de computadores paralelos ou distribuídos que são interligados entre si através de uma rede de alta velocidade. Eles executam tarefas computacionais em conjunto que não seriam possíveis de se executar em um computador comum, ou, em alguns casos, em super computadores. Como os computadores são interligados entre si no *cluster*, eles dividem a carga de trabalho computacional. Do ponto de vista do usuário, esses computadores interligados podem ser vistos como um único computador virtual. As solicitações do usuário são recebidas e distribuídas entre todos os computadores que formam o *cluster* (SADASHIV; KUMAR, 2011). Assim, o *cluster* pode processar grande quantidade de dados sem sobrecarregar um único computador. Esses dados são distribuídos por um *host* "mestre"para outros *hosts* "escravos". Um exemplo de uso de *cluster* consiste em executar algoritmos paralelos que realizam muitos cálculos simultaneamente com um grande volume de dados. O objetivo, nesse caso, é o ganho de maior desempenho na obtenção dos resultados.

Dentro da plataforma de nuvem, cada *host* do *cluster* pode virtualizar recursos computacionais, como hardware, disco rígido e rede. Entretanto, para que isso ocorra é necessário que um gerenciador de recursos de virtualização esteja instalado. Esse gerenciador é comumente conhecido como *Hypervisor*. A Figura 5 apresenta os principais recursos gerenciados pelo *Hypervisor*.

[8] Eucalyptus. Disponível em <https://www.eucalyptus.com>
[9] OpenNebula - Flexible Enterprise Cloud Made Simple. Disponível em <http://www.opennebula.org/>
[10] Ninbus. Disponível em <http://www.nimbusproject.org>
[11] OpenStack - Cloud Software. Disponível em <https://www.openstack.org>
[12] Amazon - Amazon Web Services. Disponível em <http://aws.amazon.com/pt/>

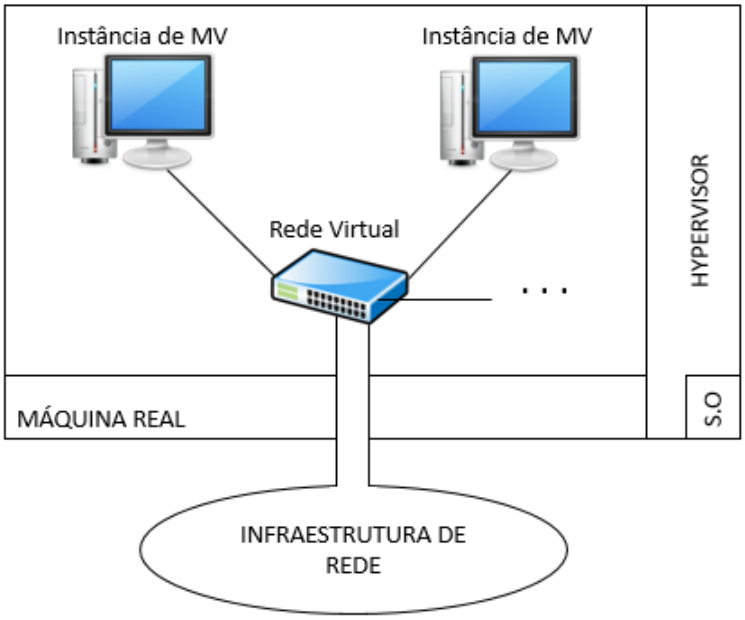

Figura 5: Principais recursos de gerenciamento do Hypervisor
Fonte: Adaptação de KRAEMER et al. (2013, p. 4)

O *Hypervisor* possui um monitor de recursos que tem o papel de fazer a comunicação entre a máquina virtual e a máquina real, ou seja, ela recebe as instruções enviadas pela máquina virtual, trata-as e as repassa para a máquina real, e vice-versa (SMITH; NAIR, 2005). Existem diversos *Hypervisors* disponíveis no mercado. Alguns exemplos são: `Xen`, `KVM` (*Kernel-based Virtual Machine*), `VMWare`, `Qemu`, `Microsoft Hyper-V`, `Virtual Box`, entre outros. Entre esses, destacamos o `KVM`[13], o `Xen`[14] e o `VMWare`[15]. O `KVM` é o *Hypervisor* padrão da plataforma `OpenNebula`, que é a plataforma utilizada nos experimentos. O `Xen` é utilizado principalmente pela Amazon, que é a maior provedora de nuvem que existe atualmente. O `VMWare` é uma plataforma proprietária que pode ser adquirida para formar todos os tipos de nuvem. Dessa forma, abordamos os principais *Hypervisors* de software livre (`KVM` e `Xen`) e software proprietário (`VMWare`). A Tabela 1 apresenta as principais características desses virtualizadores.

[13]Linux KVM. Disponível em <http://www.linux-kvm.org/page/Main_Page/>
[14]Xen Project. Disponível em <http://www.xenproject.org/>
[15]VMWare. Disponível em <http://www.vmware.com/br/>

Hypervisor	**VMWare vSphere 5.1**	**Red Hat RHEV 3.2**[16]	**Citrix XenServer 6.2**
Última Revisão	Setembro 2012	Junho 2013	Junho 2013
Gerenciamento Baseado em Navegador	Sim	Sim	Não
Gerenciamento de Operações Avançadas	Sim	Não	Não
Segurança	ESXi Firewall, vShield Zones e vShield Endpoint	SELinux, iptables,VLANs e Port Mirroring	iptables
Migração de MVs Ativas	vMotion, Metro vMotion	Live Migration	XenMotion
Migração Automatizada Ativa	CPU, Mem, Armazenamento, vCD	CPU	Não
Migração de Armazenamento	Live Storage vMotion	Fully suported	Storage XenMotion
Tamanho do Cluster	32 hots/ 4000 VMs	200 host/cluster	16 hots
Taxa Máxima de Consolidação	512 VMs, 2048 vCPU	Ilimitado	500 VMs(Windows) | 650 VMs(Linux) por Host
Qtde Máxima de CPU por Host	160	160	160
Qtde Máxima de Cores por CPU	Ilimitado	Ilimitado	Ilimitado
Qtde Máxima de Memória RAM por Host	2 TB	2 TB	1 TB
Qtde Máxima de vCPU por MV	64	160	16(Windows) | 32(Linux)
Qtde Máxima de Memória RAM por MV	1 TB	2 TB	128 GB
Overcommit Dinâmico	Memory Ballooning	Virtio	DMC
Compartilhamento de Paginação	Transparent Page Sharing	KSM	Não

Hypervisor	**VMWare vSphere 5.1**	**Red Hat RHEV 3.2**[16]	**Citrix XenServer 6.2**
Suporte a SO Convidado	Very	Limited	Good
API de Scripting	CIM/SMASH API, SDK, Perl, Power CLI	REST API,Pyton CLI, Hooks, SDK	SDK, API, PowerShell
API de Nuvem	vCloud API	REST API, Delta-cloud API	CloudStack API, AWS API
Armazenamento Suportado	DAS, NFS,FS, iSCSI, SDD for Swap, FCoE, FC HBA	DAS, iSCSI, NFS,FC, FCoE, SAS, POSIX	DAS, SAS, iSCSI, NAS, FC, FCoE
Formato do Disco Virtual	vmdk, raw disk (RDM)	RAW, Qcow2	vhd, raw disk(LUN)
Tamanho Máximo do Disco	2 TB (vmdk) / 64 TB (RDM)	8 TB	2 TB
QoS de Armazenamento	SIOC, NFS	Não	Básico
Switch de Rede Avançado	vDS	Não	Open vSwitch
VLAN	Sim	Sim	Limitado
QoS de Rede	netIOC	Não	Sim

Tabela 1: **Comparação de diversas características entre os principais Hypervisors. Fonte: Virtualization Matrix, 2013**

O *Hypervisor* é utilizado pelo Gerenciador de Recursos da Nuvem para que as instâncias sejam criadas. Ou seja, o usuário solicita instâncias, que por sua vez são processadas pelo Gerenciador de Recursos no sentido de distribuí-las ao longo dos demais *hosts* da nuvem (máquinas do *cluster* físico). Essa distribuição de recursos pode ser balanceada e monitorada. O balanceamento é uma forma de não sobrecarregar um *host* da nuvem e o monitoramento é uma forma de acompanhar o uso dos recursos computacionais, como Unidade Central de Processamento (CPU), Memória RAM e rede. Assim, a cada novo pedido de instância haverá uma alocação de forma balanceada. Por outro lado, o balanceamento é resultado de uma otimização da plataforma. Na plataforma `OpenNebula` esse recurso não faz parte da instalação padrão.

[16]Apesar do Red Hat RHEV 3.2 ser um software proprietário, em sua implemetação ele utiliza o *hypervisor* KVM, por isso a sua presença nessa comparação, pois os dados apresentados são referentes ao KVM.

O Gerenciador de Recursos da Nuvem, ou simplesmente, Gerenciador da Nuvem, é responsável por designar as tarefas que serão executadas dentro da nuvem, desde a adição de máquinas físicas ao *cluster*, até a instância de novas MVs. Para as MVs realizarem atividades de redimensionamento de recursos conforme a demanda são utilizados dois módulos, chamados de *Policy Decision Module* (PDM) e *Policy Enforcer Module* (PEM). O PDM é responsável por receber os pedidos das MVs e analisá-los. Quando o pedido é aceito, ele repassa as ações a serem executadas para o PEM. Caso o pedido seja negado, uma notificação de rejeição é enviada ao usuário. O PEM se comunica com a nuvem e é o responsável pela criação, implantação e migração de MVs. Além disso, ele é responsável por executar os pedidos repassados pelo PDM, como por exemplo, listar as MVs em execução, listar os *hosts* disponíveis, entre outros (APOSTOL et al., 2011).

A Tabela 2 mostra um comparativo entre diversos gerenciadores de nuvem, e nessa comparação estão os *Hypervisor*s que podem ser utilizados em cada um deles.

Gerenciador de Nuvem	**Open Source**	**Interface do Usuário**	**Hypervisor**
Amazon Web Service	Não	AWS Management Console, AWS APIs, Comand Line Interface (CLI)	XEN
Apache CloudStack	Sim	CloudStack API, AWS APIs, CLI, Web Interface	XEN, KVM, QEMU, VMWare e Hyper-V
Eucalyptus	Sim	AWS APIs, CLI, Web Interface	XEN, KVM, QEMU, VMWare e Hyper-V
OpenNebula	Sim	AWS APIs, Open Cloud Computing Interface (OCCI), CLI, Sunstone Web Interface	XEN, KVM, QEMU, VMWare e Hyper-V
OpenStack	Sim	AWS APIs, CLI, OpenStack Dashboard Web Interface	XEN, KVM, QEMU, PowerVM, Hyper-V, LXC e VMWare
VMWare vCloud	Não	vCloud API,vCloud Director e vCloud Connector	VMWare

Tabela 2: Comparação entre diversos Gerenciadores de Nuvem.
Fonte: Adaptação de KRAEMER et al. (2013, p. 2)

No nível mais baixo da arquitetura, uma plataforma de nuvem é formada pelo *cluster* físico, que contém o *Hypervisor* implantado em cada *host*. A plataforma usa o Gerenciador de Nuvem para gerenciar todos os recursos computacionais disponibilizados por essa infraestrutura. Entre diversas plataformas disponíveis utilizamos o `OpenNebula` porque é software

livre e pelo fato de já estar implantado na UTFPR-CM, além de se integrar com os principais *Hypervisors* e dispor de várias formas de interface com o usuário, conforme a Tabela 2.

O `OpenNebula` é um conjunto de ferramentas de software livre de computação em nuvem utilizado para gerenciar a infraestrutura de *Data Centers* complexos e heterogêneos. Ele pode oferecer formas flexíveis que ajustam dinamicamente os recursos necessários e oportuniza melhor interoperabilidade para construir nuvens privadas, públicas ou híbridas, do tipo Infraestrutura como Serviço (IaaS) (WEN et al., 2012). Em 2008 o `OpenNebula` foi publicado por Ignácio M. Llorente e Rubén S. Montera, e desde então opera como um projeto de software livre, sendo o carro-chefe da União Européia no domínio dos projetos de investigação em relação à virtualização e computação em nuvem (WIND, 2011). Algumas comunidades que utilizam ativamente o `OpenNebula` são: o Centro de Astronomia Espacial Europeu (*European Space Astronomy Centre*) e a Organização Européia para Pesquisa Nuclear (*European Organization for Nuclear Research* (CERN)) (CORDEIRO et al., 2010). Ele foi projetado para ser modular de modo a permitir a sua integração com o maior número de *Hypervisors*s e ambientes quanto possível (CORDEIRO et al., 2010). `KVM`, `XEN`, `Hyper-V` e `VMWare` (conforme Tabela 2) são alguns exemplos de *Hypervisors* compatíveis com o `OpenNebula`. A infraestrutura física do `OpenNebula` adota uma arquitetura de *cluster-like* clássica com um *front-end*, e um conjunto de nós onde MVs serão executadas. Há pelo menos uma rede física juntando todos os nós do *cluster* com o *front-end*(CORDEIRO et al., 2010).

Arquiteturalmente a plataforma `OpenNebula` é composta por três camadas, conforme Figura 6.

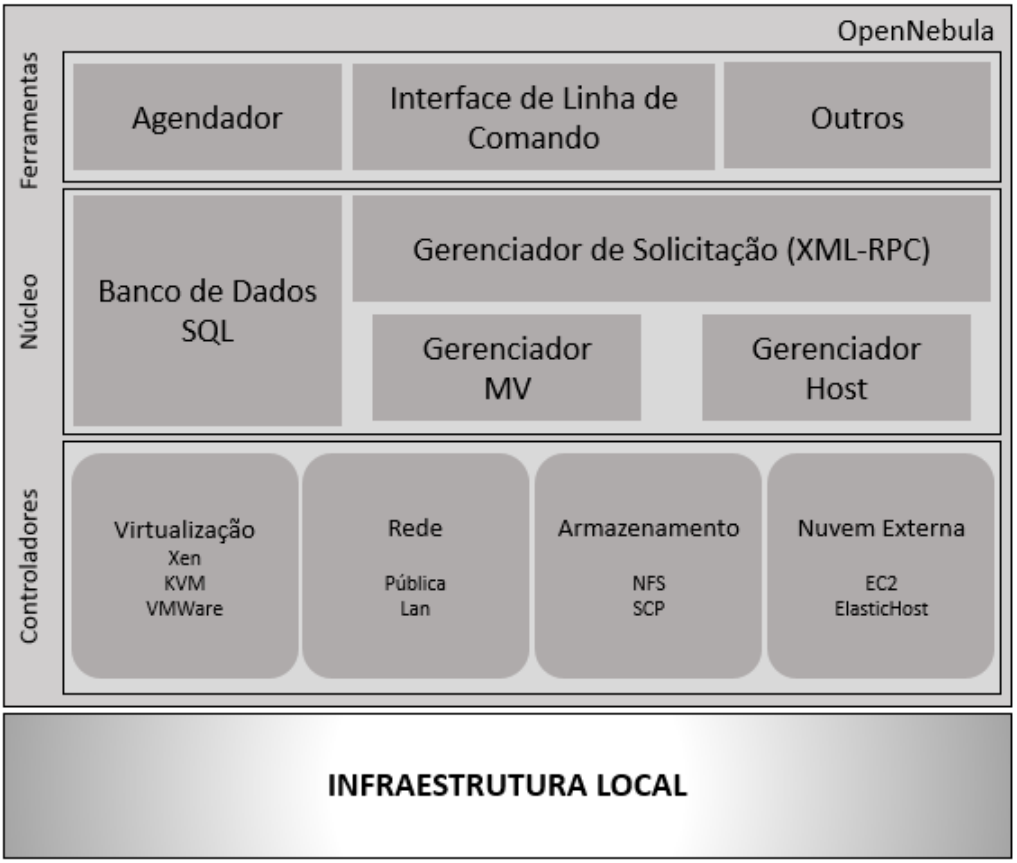

Figura 6: Arquitetura do Gerenciador de Nuvem OpenNebula
Fonte: Adaptação de Cordeiro et al. (2010), p.4.

A primeira camada (Ferramentas) contém funções para os administradores e os usuários da nuvem (WIND, 2011). Um dos componentes dessa camada é a Interface de Linha de Comando (*Command Line Interface* ou CLI), que pode ser utilizado pelos administradores para manipular a infraestrutura. O módulo Agendador, responsável pela alocação de MVs, é implementado nessa camada (CORDEIRO et al., 2010). A segunda camada (Núcleo) possui componentes usados para processar solicitações dos usuários e controle de recursos (WIND, 2011), sendo seu principal componente o Gerenciador de Soliticações (CORDEIRO et al., 2010). Por fim, a terceira camada (Controladores) dá suporte a diferentes tipos de plataformas subjacentes. Nesse nível existem controladores que são executados em processos separados que se comunicam com o módulo central através de um protocolo de mensagem simples. Esse nível contém controladores que regulam a transferência de dados e controlam as máquinas virtuais que estão em cada *host*, independente dos *Hypervisors* (WIND, 2011). Ainda nessa camada há controladores que lidam com a transferência de arquivos, que são implementados por protocolos de rede como `NFS` e `SSH`. Além disso, existem os controladores para gerenciar máquinas virtuais que são dependentes do *Hypervisor* instalado no *host* da nuvem. Por fim, existem os controladores que

solicitam serviços de nuvens externas, como o Amazon EC2 (CORDEIRO et al., 2010).

3.3.1 DEFINIÇÃO DE CLUSTER VIRTUAL

Existem diferenças e semelhanças entre um *cluster* físico e um *cluster* virtual. Um *cluster* físico é um acervo de máquinas físicas ligadas em rede. *Clusters* virtuais são construídos a partir de máquinas virtuais (MVs) instanciadas em um computador, ou distribuídas em diferentes computadores de um *cluster* físico. Se tratando de plataformas de nuvem, cada computador que forma a nuvem possui um *Hypervisor* implantado. O *Hypervisor* simula um *switch*. Por um lado, esse *switch* virtual conecta várias máquinas virtuais instanciadas localmente. Por outro lado, se comunica com uma placa de rede real fazendo uma ponte entre instâncias locais e instâncias remotas. Assim, todas as máquinas virtuais do *cluster* virtual se comunicam por meio dos *switches* virtuais. Poderão existir diferentes latências conforme a distribuição das máquinas virtuais ao longo do *cluster* físico da nuvem. Problemas de latência na comunicação e otimizações são trabalhos que vem sendo comumente apresentados na comunidade científica (HWANG et al., 2011). O desenvolvimento de um sistema web que permita alocar com especificidade instâncias de MVs contribuirá com trabalhos que visam abordar comunicação em nuvem. Um sistema com esse é nossa proposta desenvolvida como trabalho de conclusão de curso e será apresentado capítulo 4.

4 SISTEMA WEB PARA INSTÂNCIAS DE CLUSTERS VIRTUAIS

O `OpenNebula` é um gerenciador de nuvem utilizado em diversas instituições, atendendo a demanda gerada por pesquisadores de experientes e de jovens estudantes iniciantes na pesquisa. O `OpenNebula` abrange vários componentes necessários para criar e gerenciar uma nuvem, com destaque para criação de *templates* de máquinas virtuais, administração de *cluster* físico, criação de máquinas virtuais em máquinas físicas, distribuição de IPs para as máquinas virtuais, entre muitas outras funções. O gerenciamento de todas essas funções exige um conhecimento que é complexo para iniciantes. Muitas vezes os utilizadores da nuvem não querem se preocupar com configurações de máquinas virtuais e suas instâncias. Por exemplo, por questões de produtividade e até mesmo por serem de áreas distintas da computação, pesquisadores querem objetividade para instanciar cenários e avaliar algoritmos, sem se preocupar com detalhes específicos sobre as instâncias. Contudo, executar experimentos científicos em computação em nuvem pode ser um trabalho massante de reconfiguração da nuvem para cada modo de utilização.

O `BrainStone` pode ser utilizado em qualquer tipo de nuvem, desde que possua o `OpenNebula` como eu gerenciador. Ele é do tipo Infraestrutura como serviço (IaaS), pois é utilizado para instanciar as MVs que irão compor o cluster virtual.

Em experimentos científicos muitas vezes as mesmas tarefas precisam ser executadas por várias vezes e em cenários diferentes para obtençao de dados para comparação. Desta forma pode-se obter resultados mais aceitos na comunidade cientifica. Um exemplo disso é o que ocorre quando se deseja avaliar a latência da própria nuvem, onde vários testes com algoritmos podem ser feitos com diferentes distribuições de máquinas virtuais ao longo das máquinas físicas.

As instâncias de várias máquinas virtuais que se comunicam para formar um *cluster* também pode ser chamado de *cluster* virtual. A criação de *clusters* virtuais dentro da nuvem, possuindo diferentes configurações entre si (quantidades de máquinas virtuais, quantidades de memória RAM, processadores, armazenamento, tipos de processadores diferentes e máquina

física onde ficará a instância) é uma tarefa que pode ser automatizada, facilitando dessa maneira o uso da computação em nuvem. A configuração não automatizada é trabalhosa, o que demanda tempo para ser realizada. Uma mesma configuração pode ser necessária em várias telas, o que torna o processo repetitivo e entendiante para o usuário. O que pode favorecer a ocorrência de erros que irão influenciar nos experimentos e até mesmo gerar falhas no ambiente.

Partindo desses pontos, foi desenvolvido uma interface web, chamada de `BrainStone`, capaz de facilitar as instâncias dessas máquinas virtuais, tornando-as parte da configuração menos massante para os usuários, consequentemente deixando os resultados dos experimentos mais confiáveis.

O `BrainStone` consiste em uma tela que se divide em duas partes: monitoramento e instanciação. Conforme podemos observar na Figura 7.

Figura 7: Interface Gráfica do BrainStone

Na parte de monitoramento, os usuários podem ver sucintamente a nuvem, contendo apenas os detalhes mais importantes para instanciar as novas máquinas virtuais. Dentre esses detalhes se encontram: nome do *host* físico, quantidade de MVs sendo executadas no momento (divididas em MVs mestres e MVs escravas), quantidade total de memória RAM, quantidade de memória RAM que está sendo utilizada e velocidade da CPU.

Na parte de instanciação, os usuários escolhem, com base na parte de monitoramento, informações para instanciar as novas máquinas virtuais, como o *host* físico que possuirá a MV mestre, o *template* da MV mestre e quais hosts possuirão as MVs escravas, o *template* das MV escravas, que podem possuir diferentes configurações entre si, além da quantidade de máquinas virtuais que serão criadas em cada *host* físico. Com isso, instanciar *clusters* virtuais se torna uma tarefa mais rápida e menos massante para os usuários, além de possuir menos navegação entre telas, tornando-se também menos confuso.

O objetivo do `BrainStone` não é ser um substituto para o `OpenNebula`, mas sim, ser um complemento para facilitar a instanciação e distribuição de *clusters* de máquinas virtuais na nuvem. Todas as demais partes importantes para a criação de um cluster virtual serão criadas e administradas pelo próprio `OpenNebula`, além do monitoramento em tempo real das máquinas virtuais.

Para utilizar o `BrainStone` para instanciar os *clusters* virtuais é necessário seguir apenas uma regra: o nome dos *templates* mestre e escravo criados no `OpenNebula` devem seguir um padrão relativamente simples de nomenclatura, eles devem conter no nome o tipo de modelo de programação e o papel no cluster virtual (mestre ou escravo). Por exemplo, a nomenclatura para instância de um cluster virtual `BSPCGM` deverá ser `bspcgm_<especificacoes_template>_master` e `bspcgm_<especificacoes_template>_slave`, sendo que o conteúdo entre `bspcgm` e `master` são apenas especificações do template, podendo variar de *template* para *template*.

Os *templates* utilizados pelo `BrainStone` não são cadastrados no sistema, são cadastrados no `OpenNebula`, assim como todos os demais componentes utilizados pela nuvem, exceto as máquinas virtuais, que podem ser cadastradas pelo `BrainStone` ou podem ser cadastradas também pelo `OpenNebula`, ou seja, para fazer toda a validação dos nomes dos *templates*, o `BrainStone` apenas utiliza os *templates* já cadastrados pelo `OpenNebula`.

A Figura 8 mostra o sistema de escolha dos *templates* mestres e escravos, onde os *templates* mestres são posicionados no *combobox* superior e os *templates* escravos são posicionados no *combobox* inferior.

Figura 8: Visualização dos templates do BrainStone

Fonte: BrainStone

4.1 DIAGRAMA DE ATIVIDADES

Os diagramas de atividades são normalmente utilizados para modelar os processos de negócio, a lógica de um único caso de uso, ou para modelar uma regra de negócio. Um diagrama de atividade é basicamente um fluxograma, mostrando o fluxo de controle de uma atividade para outra e são utilizados para fazer a dinâmica do sistema. Diagrama de atividades é um diagrama definido pela Linguagem de Modelagem Unificada (UML)(AMBLER, 2004).

A Figura 9 mostra o diagrama de atividades do sistema, onde o ponto de entrada acontece quando o usuário visualiza os dados dos *hosts* formadores da nuvem. O usuário possui duas opções: visualizar os *hosts* físicos da nuvem e os dados referentes à nuvem e/ou instanciar um novo *cluster* virtual.

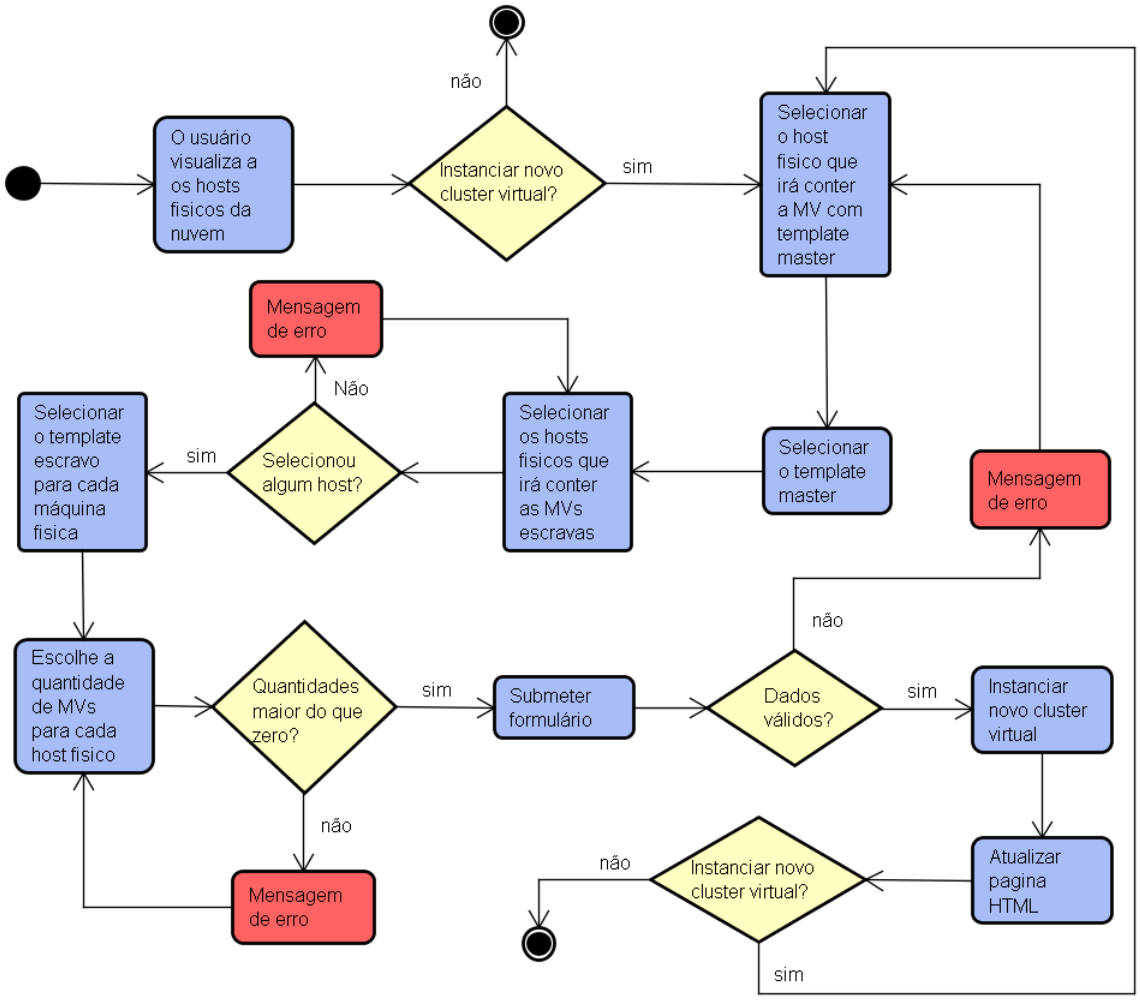

Figura 9: Diagrama de Interação do BrainStone

Caso o usuário opte apenas por visualizar a nuvem, a interação do mesmo com o sistema acaba nesse ponto, porém, caso o usuário queira instanciar as máquinas virtuais para um novo *cluster*, será necessário escolher qual a máquina física irá possuir a instância da máquina virtual *master*, ou seja, aquela que irá distribuir as tarefas para as outras máquinas executarem.

Após escolher qual máquina física irá comportar a MV *master*, será necessário escolher qual o *template* dessa MV. O *template* já possui sua especificação de memória RAM e processador. A seguir, pode-se escolher a quantidade de máquinas virtuais que serão criadas em cada máquina física da nuvem, ou seja, em uma máquina física pode haver uma ou mais máquinas virtuais com o *template* escravo escolhido anteriormente.

Ao clicar no botão para instanciar o novo *cluster* virtual, os dados serão validados para evitar possíveis erros e se tudo estiver correto, será instanciado um novo *cluster* e atualizada a página para que o usuário possa visualizar as ações anteriores. Caso algum dado não seja

válido, o sistema avisará o usuário para que seja realizada a correção e possa ser criado o novo *cluster*. Após ser realizado a criação do *cluster* virtual, o usuário tem a opção de instanciar um novo *cluster* ou apenas visualizar o *cluster* criado anteriormente.

4.2 CLASSES E MÉTODOS

O projeto `BrainStone` é um projeto relativamente pequeno, porém robusto. Ele não tem o objetivo de substituir o `OpenNebula` e o `Sunstone`, mas sim servir de auxílio na funcionalidade de criar novas instancias de máquinas virtuais para a formação de um *cluster* virtual. Devido a isso, não há uma grande quantidade de classes e métodos para a execução do seu objetivo.

O sistema consiste basicamente em executar os comandos do `OpenNebula` via linha de comando e processar os valores de saída para apresentá-los ao usuário.

A seguir é apresentada uma descrição sobre as classes do projeto.

- **Initializer:** é o *servlet* responsável por inicializar o `BrainStone`, por executar os comandos de consulta para que o usuário possa visualizar os dados da nuvem. Dentre as funções desse *servlet* estão: mostrar a quantidade de máquinas físicas que a nuvem possui, mostrar dados relevantes ao usuário para a instanciação de um novo *cluster*, como a quantidade de máquinas virtuais existentes em cada máquina física, diferenciando-as entre MVs mestres e MVs escravas, a quantidade de memória RAM, capacidade armazenamento e processamento que cada máquina física possui.

- **ComboBoxMaster:** é o *servlet* responsável por executar o comando para listar todos os *templates* cadastrados no `OpenNebula` e filtrar o resultado desse comando para que, com base no nome do *template*, possa apresentar para o usuário apenas os que terminem com a nomenclatura `master`, ou seja, todos os *templates master* cadastrados na nuvem.

- **ComboBoxSlave:** tem a funcionalidade similar ao **ComboBox Master**, que mostra ao usuário os *templates* escravos do *template* mestre selecionado no *combobox* anterior. Esse *servlet* é chamado quando o usuário seleciona o *template master* da máquina virtual mestre do *cluster* virtual da nuvem. Assim como o *servlet* **ComboBox Master**, ele executa o comando e filtra os resultados por começar com o nome do *template* mestre selecionado anteriormente e por terminar com a palavra `slave`. Por exemplo, se o *template* mestre começar com a palavra `template01` ele irá filtrar todos os *templates* escravos que iniciam com a palavra `template01`.

- **InstanciarCluster:** é o *servlet* mais importante do `BrainStone`. É o responsável por pegar os dados do usuário e criar o novo *cluster* virtual, ou seja, criar todas as máquinas virtuais com as especificações do cliente. Isso será realizado obtendo todas as informações do usuário montando os comandos do `OpenNebula` e executando-os na linha de comando.

4.3 REQUISITOS DO SISTEMA

O sistema foi desenvolvido em Java Web e utiliza recursos como Javascript e CSS na parte de visualização (JSP) e Java Servlets na parte de controle e processamento. Para que o sistema funcione são necessários alguns requisitos:

- Java EE 7
- Apache TomCat 7
- OpenNebula 4.6
- Sistema Operacional baseado em Linux

Para executar os comandos do `OpenNebula`, o `BrainStone` utiliza duas classe implementadas em linguagem Java: `ProcessBuilder` e `Process`.

O `ProcessBuilder` é utilizado para criar processos do Sistema Operacional. Ele cria uma coleção de processos que podem ser executados pela chamanda ao método `ProcessBuilder.start()`. Para utilizá-lo é necessário passar uma lista de `Strings` que representam os elementos que compõem o comando. Por exemplo, para executar o comando `ls -la` é necessário passar para o ProcessBuilder os parâmetros `"ls"` e `"-la"` separados (ORACLE, 2014b).

Quando o método `ProcessBuilder.start()` é chamando, ele retorna uma instância de um `Process` no qual podemos obtê-lo para controlar e/ou obter informações da execução desse `Process` no terminal do sistema operacional (ORACLE, 2014a).

A saída do comando executado é recuperada usando-se o método `Process.getInputStream()` que retorna a saída padrão do comando, ou o `Process.getErrorStream()` para pegar a saída de erro do processo caso aconteça alguma falha. O `BrainStone` utiliza esses princípios para poder funcionar. Nos *servlets* **Initializer**, **ComboBox Master** e **ComboBox Slave** utilizamos esse mencanísmo para obter

as saídas dos comandos para mostrar resultados na parte gráfica do sistema, e no *servlet* **InstanciarCluster** é utilizado para obter as saídas de erro, causo aconteçam.

A Figura 10 mostra a saída do comando `onehost show 0` executado no terminal do *host* com o usuário do `OpenNebula`, onde o numero `0` é o identificador do *host* físico na nuvem.

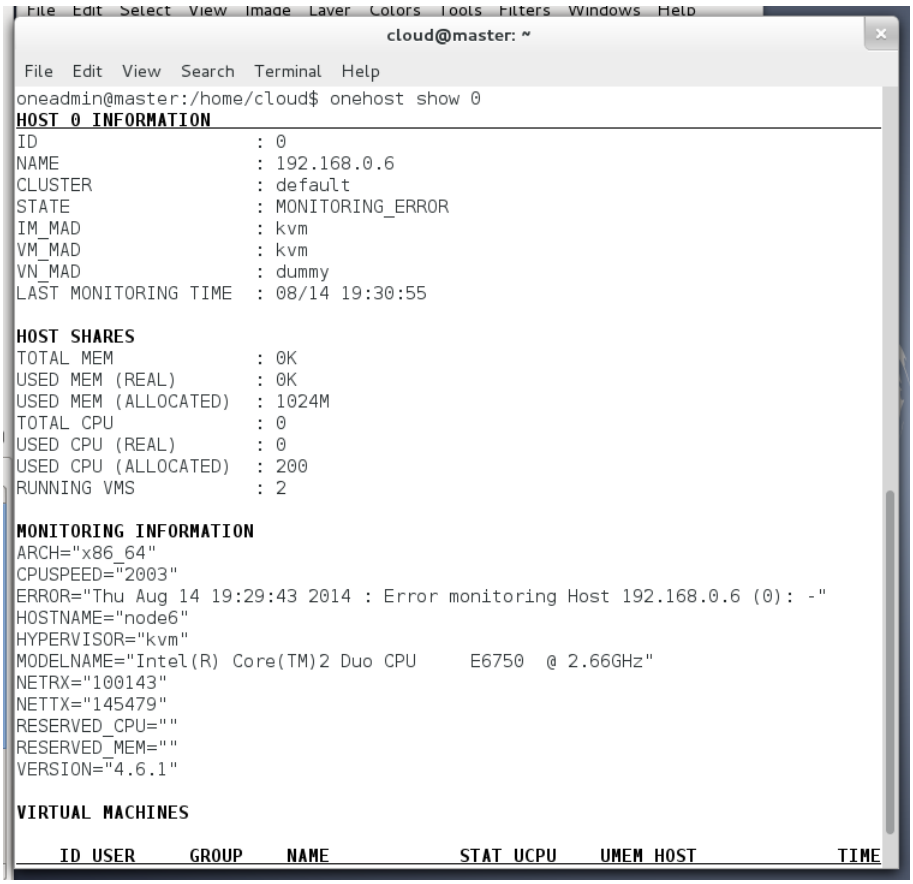

Figura 10: Saída do comando no console Linux

A Figura 11 mostra os detalhes do *host* de identificador `0` na interface gráfica do `Sunstone`.

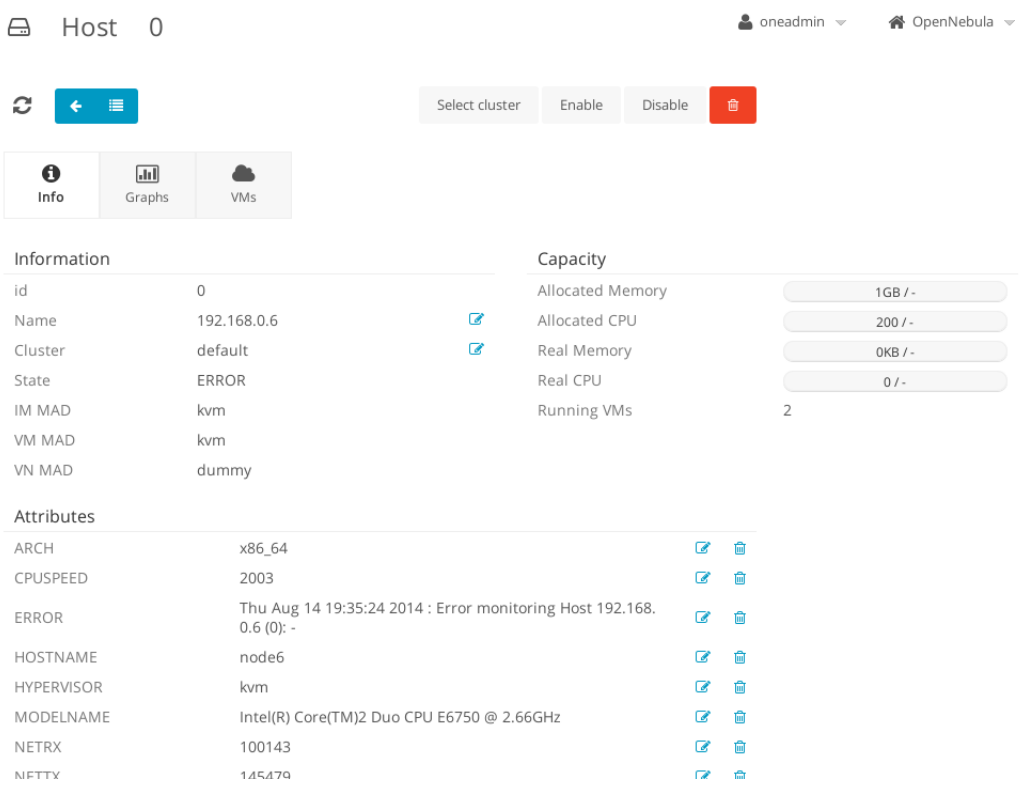

Figura 11: Visualização dos detalhes do Host 0 pelo Sunstone

O sistema encontra-se localizado no *host* físico mestre da nuvem, podendo ser acessado por qualquer navegador presente na máquina. Os comandos são executados no terminal do *host* mestre da nuvem, pois o mesmo é onde se encontra instalado o `OpenNebula`.

A Figura 12 mostra o interação do `BrainStone` com os demais componentes formadores da nuvem.

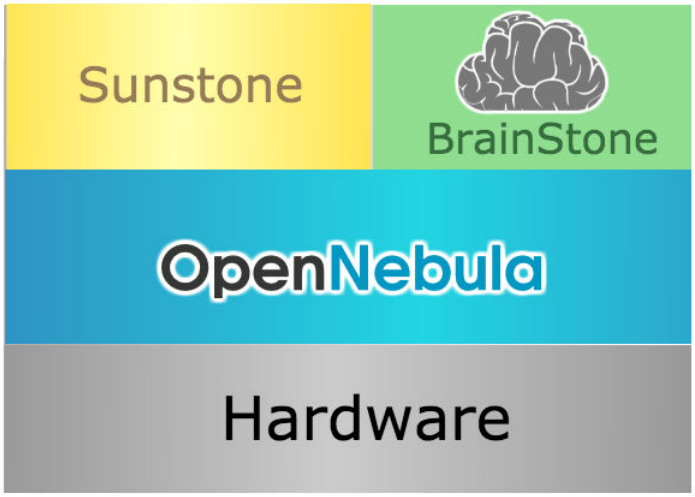

Figura 12: Interação do BrainStone com os demais componentes da nuvem

Como dito anteriormente, o BrainStone não tem a função de substituir o Sunstone, mas trabalhar em conjunto com ele para melhorar a interação do usuário com a nuvem. A Figura 12 mostra o posicionamento do BrainStone em relação à nuvem, onde na parte inferior encontra-se a infraestrutura da nuvem, com toda a parte de hardware (*hosts* físicos, *switches*, roteadores, etc). Na camada acima do Hardware, encontra-se o OpenNebula e todos os seus componentes instalados no hardware da nuvem. Na camada acima encontra-se as interfaces gráficas na qual o usuário do sistema irá interagir com o OpenNebula.

Nessa camada o Sunstone e o BrainStone estão lada a lado, pois o Sunstone completa o funcionamento do BrainStone e esse, por sua vez, facilita a interação do usuário na parte de instanciar novos clusters virtuais.

O próximo e último capítulo irá apresentar as conclusões obtidas por meio desta pesquisa e a sugestão de alguns trabalhos futuros.

5 CONCLUSÕES E TRABALHOS FUTUROS

Este trabalho de conclusão de curso apresentou o desenvolvimento de um sistema web para facilitar a instanciação de *clusters* virtuais em nuvem. As aplicações científicas executadas em nuvem exigem um trabalho dispendioso por parte do desenvolvedor e atualmente os mecanismos utilizados por esse trabalho, como o caso do `Sunstone`, pode ainda expor o usuário a erros durante o processo de instanciação.

Durante o desenvolvimento da ferramenta `BrainStone`, foi necessário um estudo sobre o `OpenNebula` para descobrir como é realizado a instanciação de máquinas virtuais e os demais recursos que compõem um *cluster*. Também foi necessário um estudo sobre o `Sunstone` para o entendimento de como obter os resultados do `OpenNebula` e os transformar em uma interface gráfica mais amigável para o usuário.

O `BrainStone` não substitui o `Sunstone`, mas sim, trabalha em conjunto para gerenciar a nuvem. O `BrainStone` é utilizado para instanciar as MVs e fazer um acompanhamento *offline* e o `Sunstone` é utilizado em todas as demais partes gerenciais da nuvem, assim como o acompanhamento em tempo real.

O `BrainStone` oferece um benefício à comunidade de pesquisadores que utilizam computação em nuvem, pois com a utilização do mesmo é possível instanciar MVs mais rapidamente, reduzindo o tempo dos experimentos, ou até aproveitando esse tempo com experimentos em outras configurações.

Como trabalho futuro, propomos um trabalho para executar os cadastros de *hosts* físicos, *templates*, entre outros de forma mais simplificada. Outro trabalho é estender o `BrainStone` para realizar o monitoramento *online* dos *clusters*, assim como no `Sunstone`.

REFERÊNCIAS

AMBLER, S. **The Object Primer - Agile Model Driven Development with UML 2**. Estados Unidos da América - EUA: Cambridge University Press, 2004.

APOSTOL, E.; BALUTA, I.; GORGOI, A.; CRISTEA, V. Efficient manager for virtualized resource provisioning in cloud systems. In: **Intelligent Computer Communication and Processing (ICCP), 2011 IEEE International Conference on**. Cluj-Napoca, Romania: IEEE, 2011. p. 511–517.

BRITANNICA, E. **Encyclopedia Britannica - Cloud Computing**. 2013. Acesso em 21 ago. 2013. Disponível em: <http://www.britannica.com/EBchecked/topic/1483678/cloud-computing>.

CORDEIRO, T.; DAMALIO, D.; PEREIRA, N.; ENDO, P.; PALHARES, A.; GONçALVES, G.; SADOK, D.; KELNER, J.; MELANDER, B.; SOUZA, V.; MåNGS, J. E. Open source cloud computing platforms. In: **Grid and Cooperative Computing (GCC), 2010 9th International Conference on**. Nanjing, Jiangsu, China: IEEE, 2010. p. 366–371.

DAWOUD, W.; TAKOUNA, I.; MEINEL, C. Infrastructure as a service security: Challenges and solutions. In: **Informatics and Systems (INFOS), 2010 The 7th International Conference on**. Cairo, Egito: IEEE, 2010. p. 1–8.

HWANG, K.; J., D.; FOX, G. **Cloud Computing: Virtual Clusters: Virtual clusters convey certain benefits over physical clusters in terms of speed, storage and flexibility**. 2011. Acesso em 21 set. 2013. Disponível em: <http://technet.microsoft.com/en-us/magazine/jj574501.aspx>.

KRAEMER, A.; OLIVEIRA, J. C.; SANTOS, F. G.; MACIEL, A. C.; GOLDMAN, A.; CORDEIRO, D. A. Dynamic creation of bsp/cgm clusters on cloud computing platforms. **4TH International Conference on Emerging Intelligent Data and Web Technologies - Workshop: Third International Workshop on the Service for Large Scale Distributed Systems (SeDiS-2013)**, 2013.

MELL, P.; GRACE, T. **The NIST Definition of Cloud Computing**. 2013. Acesso em 29 ago. 2013. Disponível em: <http://www.cloudbook.net/resources/stories/the-nist-definition-of-cloud-computing>.

MERIAT, V. **Modelos de Serviço na Nuvem: IaaS, PaaS e Saas**. 2014. Acesso em 14 ago. 2014. Disponível em: <http://vitormeriat.com.br/2011/07/08/modelos-de-servio-na-nuvem-iaas-paas-e-saas/>.

ORACLE. **Java - Process**. 2014. Acesso em 10 ago. 2014. Disponível em: <http://docs.oracle.com/javase/7/docs/api/java/lang/Process.html/>.

ORACLE. **Java - ProcessBuilder**. 2014. Acesso em 10 ago. 2014. Disponível em: <http://docs.oracle.com/javase/7/docs/api/java/lang/ProcessBuilder.html/>.

SADASHIV, N.; KUMAR, S. Cluster, grid and cloud computing: A detailed comparison. In: **Computer Science Education (ICCSE), 2011 6th International Conference on**. SuperStar Virgo Singapore, Singapura: IEEE, 2011. p. 477–482.

SALDANHA, H.; RIBEIRO, E.; BORGES, C.; ARAUJO, A.; GALLON, R.; HOLANDA, M.; WALTER, M.; TOGAWA, R.; SETUBAL, J. Towards a hybrid federated cloud platform to efficiently execute bioinformatics workflows, bioinformatics, dr. horacio pérez-sánchez. **InTech**, 2012.

SCHOENHAGEN, P.; ZIMMERMANN, M.; FALKNER, J. Advanced 3-d analysis, client-server systems, and cloud computing — integration of cardiovascular imaging data into clinical workflows of transcatheter aortic valve replacement. **Cardiovascular Diagnosis and Therapy**, v. 3, n. 2, p. 80–92, 2013. ISSN 2223-3660. Disponível em: <http://www.thecdt.org/article/view/1583>.

SMITH, J.; NAIR, R. The architecture of virtual machines. IEEE, Kyoko, Japão, v. 38, n. 5, p. 32–38, 2005.

SOUSA, F. R. C.; MOREIRA, L. O.; MACEDO, J. A. F. de; MACHADO, J. C. Gerenciamento de dados em nuvem: Conceitos, sistemas e desafios. **Tópicos em sistemas colaborativos, interativos, multimidia, web e bancos de dados, Sociedade Brasileira de Computacao**, p. 101–130, 2010.

SOUSA, F. R. C.; MOREIRA, L. O.; MACHADO, J. C. Computação em nuvem: Conceitos, tecnologias, aplicações e desafios. **III Escola Regional de Computação Ceará-Maranhão-Piauí, ERCEMAPI**, v. 1, 2009.

SULTAN, N. Cloud computing for education: A new dawn? **International Journal of Information Management**, Elsevier, v. 30, n. 2, p. 109–116, 2010.

TAURION, C. **Computação em Nuvem - Transformando o Mundo da Tecnologia da Informação**. Rio de Janeiro, Brasil: Brasport, 2009.

VORAS, I.; MIHALJEVIC, B.; ORLIC, M.; PLETIKOSA, M.; ZAGAR, M.; PAVIC, T.; ZIMMER, K.; CAVRAK, I.; PAUNOVIC, V.; BOSNIC, I.; TOMIC, S. Evaluating open-source cloud computing solutions. In: **MIPRO, 2011 Proceedings of the 34th International Convention**. Croácia: IEEE, 2011. p. 209–214.

WEN, X.; GU, G.; LI, Q.; GAO, Y.; ZHANG, X. Comparison of open-source cloud management platforms: Openstack and opennebula. In: **Fuzzy Systems and Knowledge Discovery (FSKD), 2012 9th International Conference on**. Chongqing, Sichuan, China: IEEE, 2012. p. 2457–2461.

WIND, S. Open source cloud computing management platforms: Introduction, comparison, and recommendations for implementation. In: **2011 IEEE Conference on Open Systems (ICOS)**. Langkawi, Malásia: IEEE, 2011. p. 175–179.