CHARLES BOULHOSA RODAMILANS

UMA METODOLOGIA PARA CARACTERIZAÇÃO DE APLICAÇÕES E DE INSTÂNCIAS DE MÁQUINAS VIRTUAIS NO AMBIENTE DE COMPUTAÇÃO EM NUVEM

São Paulo

2014

CHARLES BOULHOSA RODAMILANS

UMA METODOLOGIA PARA CARACTERIZAÇÃO DE APLICAÇÕES E DE INSTÂNCIAS DE MÁQUINAS VIRTUAIS NO AMBIENTE DE COMPUTAÇÃO EM NUVEM

Tese apresentada à Escola Politécnica da Universidade de São Paulo para obtenção do Título de Doutor em Ciências

Área de Concentração:
Engenharia de Computação

Orientador:
Prof. Dr. Edson Toshimi Midorikawa

São Paulo
2014

AGRADECIMENTOS

Ao meu orientador, professor Edson Toshimi Midorikawa, por sua ajuda e sugestões que foram valiosíssimas para o direcionamento deste trabalho. Também por me ajudar nos momentos críticos e pela amizade construída no decorrer dos anos.

À professora Liria Matsumoto Sato por acreditar em mim e por me dar forças em todas as fases dessa jornada. Também agradeço a sua amizade e seu lado humano.

À minha esposa, Calila de Santana Rodamilans, que foi a pessoa que mais teve paciência comigo, sendo uma parceira maravilhosa. Obrigado por fazer parte da minha vida.

Aos colegas do laboratório que me proporcionaram momentos de descontração e de aprendizado. Em especial, agradeço a Artur Baruchi e Fernando Ryoji Kakugawa pelas contribuições e pelo companheirismo.

RESUMO

Os provedores de Computação em Nuvem estão fornecendo diversas instâncias e configurações de Máquinas Virtuais. Entretanto, existe dificuldade quanto à seleção da instância mais adequada para uma determinada aplicação, levando em consideração, por exemplo, o menor tempo de execução. O presente trabalho tem como objetivo determinar os procedimentos para a seleção de uma instância de Máquina Virtual para uma determinada aplicação e também propor uma arquitetura que permita interligar os diferentes provedores de Nuvem. Foram implementadas funcionalidades da arquitetura e também foram aplicados os procedimentos propostos em uma aplicação científica e em instâncias de diferentes provedores de Nuvem. Os principais resultados obtidos foram (a) desenvolvimento da metodologia Caracterização, Seleção e Execução (CSE); (b) demonstração da importância da caracterização e do recurso preponderante da aplicação; (c) avaliação de desempenho do disco em diversas Nuvens; (c) caracterização e avaliação de desempenho da aplicação OpenModeller; e (d) arquitetura de interligação de nuvens públicas e privadas, e implementação de suas principais funcionalidades. Estes resultados mostram que o principal problema de seleção das instâncias pode ser solucionado a partir da caracterização das instâncias e das aplicações.

Palavras-chave: Computação em Nuvem. Sistemas Distribuídos. Metodologia. Análise de Desempenho.

ABSTRACT

The Cloud Computing providers are providing several instances and configurations of Virtual Machines. However, there is a difficulty in selecting the instance more adequate for specific application. This study aims to determine the procedures for selecting a Virtual Machine instance for a particular application and also propose an architecture to link the different Cloud providers. Features of the architecture have been implemented and the proposed procedures were applied in scientific applications and in instances of different Cloud providers. The main results were (a) development of Characterization, Selection and Execution (CSE) of methodology; (b) demonstration of the importance application characterization and the preponderant resource; (c) performance characterization and evaluation of OpenModeller application; (d) interconnection architecture of public and private Clouds, and implementation of their main features. These results show that the main instance selection problem can be solved from the instances and applications characterization.

Keywords: Cloud Computing. Distributed Systems. Methodology. Performance Evaluation.

LISTA DE ILUSTRAÇÕES

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

AMI	*Amazon Machine Image* (Imagem de Máquina da Amazon)
API	*Application Programming Interface* (Interface de Programação de Aplicações)
CPU	*Central Processing Unit* (Unidade Central de Processamento)
E/S	Entrada e Saída
EBS	*Amazon Elastic Block Store* (Armazenamento de Bloco Elástico da Amazon)
EC2	*Amazon Elastic Compute Cloud* (Nuvem de Computação Elástica da Amazon)
ECU	*Elastic Compute Unit* (Unidade de Computação Elástica)
FCoE	*Fibre Channel over Ethernet*
FD	Ferramenta de Desenvolvimento
MMV	Monitor de Máquinas Virtuais
HDD	*Hard Disk Drive* (Disco Rígido)
HPC	*High Performance Computing* (Computação de Alto Desempenho)
HTC	*High Throughput Computing* (Computação de Alta Vazão)
HTTP	*Hypertext Transfer Protocol* (Protocolo de Transferência de Hipertexto)
IaaS	*Infrastructure as a Service* (Infraestrutura como Serviço)

IOPS	*Input/Output Operations Per Second* (Operações de Entrada e Saída por Segundo)
iSCSI	*Internet Small Computer System Interface*
KVM	*Kernel-based Virtual Machine* (Máquina Virtual baseada no Núcleo)
LAHPC	*Laboratory of Architecture and High Performance Computing* (Laboratório de Arquitetura e Computação de Alto Desempenho)
MV	Máquina Virtual
OCCI	*Open Cloud Computing Interface* (Interface de Computação de Nuvem Aberta)
OVF	*Open Virtual Format* (Formato Virtual Aberto)
PaaS	*Platform as a Service* (Plataforma como Serviço)
POLI	Escola Politécnica
QoS	*Quality of Service* (Qualidade de Serviço)
REST	*Representational State Transfer* (Transferência de Estado Representativo)
S3	*Amazon Simple Storage Service* (Serviço de Armazenamento Simples da Amazon)
SaaS	Software as a Service (Software como Serviço)
SATA	*Serial Advanced Technology Attachment*
SO	Sistema Operacional
SOAP	*Simple Object Access Protocol* (Protocolo Simples de Acesso a Objetos)

SSD	*Solid-State Drive* (Disco de Estado Sólido)
USP	Universidade de São Paulo
vCPU	*virtual Central Processing Unit* (Unidade Central de Processamento virtual)

SUMÁRIO

1. INTRODUÇÃO

Em decorrência dos avanços ocorridos na computação, os usuários passaram a demandar por serviços relacionados a esta área, nos mesmos moldes como são fornecidos serviços essenciais, a exemplo de água encanada, energia elétrica e telefonia. Visando ao suprimento de tais exigências, foi criado o serviço de Computação em Nuvem, detalhado neste trabalho.

Atualmente, Computação em Nuvem (ARMBRUST *et al.*, 2009) tem sido adotada em muitas empresas, governos e universidades. As razões técnicas para essa adoção são o isolamento de desempenho, elasticidade, serviços sob demanda (*on-demand*), escalabilidade e disponibilidade (ARMBRUST *et al.*, 2009). As nuvens têm sido usadas para o armazenamento e processamento de uma grande quantidade de dados.

1.1. MOTIVAÇÃO

Não existe um consenso para definir precisamente Computação em Nuvem (*Cloud Computing*). Há diversas definições. Uma delas é :

> "Computação em Nuvem refere-se tanto às aplicações entregues como serviços utilizando a Internet quanto a hardware e software dos Centros de Dados que provêem esses serviços." (Tradução de (ARMBRUST *et al.*, 2009)).

Existe uma idéia errônea de que Computação em Nuvem é a Internet. Isso acontece porque (a) em representações gráficas de topologia de Rede de Computadores, utiliza-se uma nuvem como uma simbologia para os dados que serão trafegados pela Internet; e também porque (b) o serviço de Computação em Nuvem necessita estar conectado à Internet para que se possa ter acesso ao serviço de qualquer lugar e de qualquer computador.

Nas representações gráficas, para demonstrar a topologia das Redes de Computadores, a nuvem representa a abstração do caminho de dados até estes chegarem ao seu destino. Ou seja, para representar como os computadores comunicam-se utilizando a Internet, os computadores são conectados a uma nuvem, e essa nuvem representa a Rede Internet. Essa nuvem abstrai os caminhos que os dados podem realizar para chegar ao seu destino. Este tipo de representação gráfica também contribuiu para a falsa percepção de que Computação em Nuvem (*Cloud Computing*) seria a Internet.

De qualquer maneira, Computação em Nuvem assemelha-se aos Serviços de Utilidade Pública (*Utility Services*), tais como fornecimento de energia elétrica, água e telefone. No modelo convencional de hospedagem e licença de software, paga-se pelo aluguel da hospedagem e/ou licença do software. O cliente aluga uma máquina e um serviço de armazenamento de dados e compra a licença vitalícia ou temporária de uma versão do software. O valor é pago independentemente da real utilização dos recursos e dos softwares.

Já no modelo econômico de Nuvem, paga-se somente pelo serviço que foi utilizado, seja no nível de hardware (*Infrastructure as a Service - IaaS*), plataforma (*Platform as a Service - PaaS*) ou de software (*Software as a Service - SaaS*). Esses tipos de serviços serão detalhados na seção 2.1.3. O modelo de Computação Utilitária (*Utility Computing*) não é um conceito novo (ARMBRUST *et al.*, 2009)(PARKHILL, 1966). Entretanto, sempre houve limitação tecnológica para sua implantação. Este conceito está sendo concretizado pela Computação em Nuvem.

Como exemplo de serviço de hardware (IaaS), a Amazon EC2 (AMAZON, 2014a)(JAMES MURTY, 2008) fornece um serviço de (ambiente de) Computação Virtual. O usuário deste serviço paga pela quantidade de tráfego de rede, de uso da CPU, da memória e pelo tipo de sistema operacional que utilizou por hora. No nível de *software* (SaaS), a Google Apps (GOOGLE, 2014b)(VECCHIOLA; PANDEY; BUYYA, 2009) oferece vários aplicativos (Gmail, Google Calendar, Google Docs) que são acessados através do navegador Web.

Atualmente, os serviços de plataformas são oferecidos para a utilização de uma Nuvem específica e há uma limitação da linguagem de programação. A

utilização dessas plataformas causa o “aprisionamento” a uma Nuvem específica, tornando o cliente vulnerável à política da Nuvem. Este aprisionamento ocorre devido à dificuldade de transição das aplicações de maneira transparente e à falta de interoperabilidade entre as Nuvens. Como exemplo, o App Engine (Google) (GOOGLE, 2014a)(VECCHIOLA; PANDEY; BUYYA, 2009)(ARMBRUST *et al.*, 2009) e o Azure (Microsoft) (AZURE, 2014)(VECCHIOLA; PANDEY; BUYYA, 2009)(ARMBRUST *et al.*, 2009) oferecem uma plataforma de desenvolvimento de aplicações proprietária, específica para suas respectivas infraestruturas.

Uma forma de contornar este problema é a criação de uma plataforma aberta, que interligue as diferentes Nuvens. Entretanto, devido aos serviços de plataforma disponíveis atualmente serem fechados, essa interligação fica restrita no nível de infraestrutura (IaaS). Disso ocorre a falta de uma plataforma que interligue os diferentes recursos virtualizados do IaaS com a transparência desejada pelo usuário. Este, por outro lado, precisa ter a opção de escolha de qual infraestrutura quer utilizar, para não ficar restrito às políticas de uma única Nuvem.

A construção desta plataforma pode ser realizada de duas maneiras: por imagem da Máquina Virtual (MV) ou por contêiner. A imagem da Máquina Virtual já possui instalado o sistema operacional e a aplicação, bem como os demais programas necessários para executar a aplicação. Entretanto, cada software de virtualização (tais como Xen (BARHAM *et al.*, 2003)(HAGEN, 2008), VMware (VMWARE, 2014)(SMITH; NAIR, 2005)(HAGEN, 2008), VirtualBox (VIRTUALBOX, 2014)(HAGEN, 2008), KVM (KVM, 2014)(HAGEN, 2008)) possui seu próprio formato de imagem. Ou seja, a imagem utilizada por um IaaS baseada em Xen não poderá ser migrada para um IaaS baseado em Vmware. O Formato de Virtualização Aberto (*Open Virtualization Format* - OVF) (DMTF, 2010) é uma tentativa de padronização da imagem de Máquinas Virtuais, porém, atualmente, os provedores de IaaS não fornecem essa alternativa. A falta de utilização de um padrão de imagem dificulta a interoperabilidade entre IaaS.

Outra solução é a utilização de contêineres. Contêiner é projetado para ajudar a organizar e armazenar itens que são colocados dentro dele (ECKEL, BRUCE, 2006). Uma classe contêiner é uma classe projetada para organizar múltiplas

instâncias de outras classes. Um contêiner pode implementar um ambiente para a execução dos serviços. Sendo assim, sua principal responsabilidade é inicializar os serviços quando for necessário. O contêiner fica hospedado nas MVs da Nuvem para executar e monitorar a MV e a aplicação. Esses serviços podem ser compilação, monitoramento, execução e escalonamento. O contêiner pode ser instalado em cada Máquina Virtual (MV), fornecendo subsídios para que a aplicação seja executada. Dessa forma, é possível utilizar contêiner em diferentes IaaS, permitindo que a aplicação seja migrada entre diferentes IaaS de maneira transparente para os usuários.

Outro grande obstáculo para que as empresas ingressem na Nuvem está relacionado à Disponibilidade de Serviços oferecidos pelos IaaS (ARMBRUST *et al.*, 2009). A Amazon oferece 99,9% de disponibilidade. Entretanto, ficou 10 horas, em dois dias, com o serviço Simple Storage Service (S3) fora do ar no ano de 2008. Em (DAHARSH, J., 2014), um cliente do GoGrid reclama que seu site ficou 30 horas fora do ar, e o suporte técnico não forneceu a devida assistência.

Para empresas de grande porte, como bancos e lojas online, ficar tanto tempo indisponível é uma situação inviável, podendo haver sanções do Banco Central, no caso dos Bancos brasileiros, ou dos clientes que fazem propagandas negativas, no caso de lojas online. Uma saída para incrementar a disponibilidade aumentando a tolerância a falhas é a criação e utilização de múltiplos provedores de Nuvem.

As empresas não têm a intenção de que todos os seus dados fiquem em Nuvens Públicas. Primeiro, porque não desejam que seus dados sigilosos, que compõem o cerne da empresa, estejam armazenados em locais onde não poderão ter controle total, ou seja, não querem confiar este tipo de dados ao gerenciamento de terceiros. Outro motivo está relacionado às leis do país onde os dados estarão armazenados. Por exemplo, nos EUA existe a lei USA PATRIOT Act (ARMBRUST *et al.*, 2009), segundo a qual todos os dados podem ser inspecionados pelo governo caso haja suspeita de terrorismo.

Algumas soluções são possíveis para que as empresas comecem a ingressar na Nuvem (TAURION, C., 2009). A primeira está relacionada às aplicações que não são o foco da empresa, como por exemplo o serviço de *call-center*. A aplicação de

gerenciamento do *call-center* pode ser disponibilizada utilizando os serviços da Nuvem.

Outra solução é permitir que aplicações existentes possam utilizar a Nuvem de maneira transparente para o desenvolvedor. Ou seja, a plataforma deve prover os serviços necessários para que uma aplicação do cliente, desenvolvida em Java, por exemplo, seja escalonada para uma Nuvem de acordo com alguns requisitos, tais como serviço de maior disponibilidade ou de menor preço. Ou seja, o código legado seria aproveitado com os benefícios da Nuvem.

Outro recurso é determinar os dados que podem ser armazenados em Nuvem. Por exemplo, é possível definir as tabelas dos Bancos de Dados que podem ficar apenas na Nuvem Privada da empresa, apenas em na Nuvem Pública ou as que podem ficar hospedadas em ambas as Nuvens. Essa identificação permite às empresas começarem a migrar e utilizar os serviços de Nuvem.

A escalabilidade na Nuvem está relacionada à aplicação e/ou aos dados. Para que haja escalabilidade de maneira dinâmica, é necessário que a Nuvem se adapte às novas configurações do ambiente. É viável, por exemplo, criar novas instâncias de Máquinas Virtuais quando houver mais acessos do que o esperado, aumentando assim os recursos computacionais disponíveis. Entretanto, é necessário detectar dinamicamente esse pico de acesso, identificar o padrão de acesso que pode ocorrer, por exemplo, diariamente às 17h. Em seguida, escalonar em qual provedor de Nuvem a instância será criada, se é na Nuvem Pública ou Privada. O escalonamento também é responsável por selecionar em qual Nuvem os dados estarão localizados.

Uma instância em Computação em Nuvem é um servidor que executa aplicações. Um tipo de instância é um conjunto de recursos computacionais, tais como CPU, memória, armazenamento e rede. A grande variedade de tipos de instâncias permite que se possa escolher qual deles é mais adequado para uma determinada aplicação. No entanto, uma vasta possibilidade de tipos de instâncias pode dificultar a seleção precisa. Isto porque um provedor pode fornecer diversos tipos de instâncias, com interoperabilidade entre as Nuvens, e diversos provedores de Nuvem podem ser utilizados. Isto amplia a oferta dos tipos de instâncias,

dificultando ainda mais a seleção. Outro fator que atrapalha a seleção é a falta de padronização dos tipos de instâncias. Cada provedor define os nomes e as características dos recursos computacionais de forma independente. Para um pesquisador, cujo objetivo é apenas executar sua aplicação, escolher qual o provedor e o tipo de instância é mais adequado para sua aplicação, é uma tarefa complexa.

Como narrado, muitos provedores de Nuvem oferecem serviços de instâncias e armazenamento (ACETO *et al.*, 2013) sendo é difícil escolher qual provedor utilizar, dependendo de qual fator de escolha é mais crucial (*e.g.*, performance, custo, segurança, ou tolerância a falhas). Muitos outros aspectos podem ser considerados; por exemplo, Terry et al. (TERRY *et al.*, 2013) discute aspectos de consistência para a Nuvem de armazenamento ser incluída dentro de contratos de Acordo de Nível de Serviço (ANS, ou SLA, do inglês, *Service-Level Agreement*).

A caracterização do provedor de Nuvem (e consequentemente suas instâncias) é um dos passos para entender qual aplicação é mais apropriada para esse provedor. Diversos aspectos podem ser analisados no ambiente de Nuvem, como computação, acesso a memória e Entrada/Saída (E/S, ou I/O, do inglês *Input/Output*) (VEDAM; VEMULAPATI, 2012). Essas características estão diretamente relacionadas com as características da aplicação (CPU *bound*, Memória *bound* e E/S *bound*). É difícil adquirir especificações de hardware em Computação em Nuvem porque os provedores fazem um encapsulamento do hardware. Provedores de Nuvem têm muitas opções de instâncias e armazenamento (EXPÓSITO *et al.*, 2012)(EXPÓSITO *et al.*, 2013). Algumas tecnologias são incorporadas aos serviços de Nuvem para melhorar o desempenho, *e.g.* Disco de Estado Sólido (SSD, do inglês, *Solid State Drive*) (EXPÓSITO *et al.*, 2013). Todas essas características demonstram a complexidade da escolha de um provedor de serviço.

Dentro deste contexto, existem vários trabalhos de avaliação de desempenho em Computação em Nuvem, *e.g.* Computação de Alto Desempenho (HPC, do inglês, *High Performance Computing*) (EXPÓSITO *et al.*, 2013)(MARATHE *et al.*, 2013)(AZURE, 2014) e desempenho de E/S (EXPÓSITO *et al.*, 2013)(GHOSHAL;

CANON; RAMAKRISHNAN, 2011). Em todos esses trabalhos, os benchmarks são usados para entender os aspectos de desempenho da Nuvem (VEDAM; VEMULAPATI, 2012)(IOZONE, 2014)(DONGARRA; LUSZCZEK; PETITET, 2003)(NAS, 2014)(GILADI; AHITAV, 1995).

A grande diversidade de provedores de Nuvem traz uma substancial variedade de instâncias e preços. Assim, é difícil escolher não só qual instância utilizar, como também onde os dados serão armazenados para serem processados na Nuvem. E, como visto, a dificuldade aumenta com a disponibilidade de muitos provedores de Nuvem que oferecem diferentes tipos de instâncias e configurações. Surge então o questionamento: qual o tipo e a configuração da instância é mais adequada para uma determinada aplicação, buscando, por exemplo, o menor tempo de execução? Esse problema mostra a necessidade de um mecanismo que compare a variedade de instâncias dos provedores de Nuvem.

Para que se possa selecionar a melhor instância para a aplicação é importante que a aplicação também seja caracterizada. A análise das características da aplicação e a variação da carga de trabalho permite entender o seu comportamento e funcionamento. Desse modo, a caracterização prévia da aplicação e da instância permitirá uma seleção mais adequada.

1.2. OBJETIVO

A hipótese adotada neste trabalho é de que o recurso computacional preponderante da aplicação pode definir qual o tipo de instância é mais adequada para uma determinada aplicação sem a necessidade de executar a aplicação em todas as instâncias.

Sendo assim, se o recurso computacional dominante permite determinar qual o tipo de instância é mais adequado, então as aplicações poderão ser executadas em menos tempo ou podem obter o menor custo financeiro.

1.2.1. Objetivo Geral

O objetivo deste trabalho é determinar os procedimentos necessários para selecionar o tipo de instância em que uma determinada aplicação será executada, levando-se em consideração aspectos de custo ou desempenho.

1.2.2. Objetivos Específicos

Para cumprir o objetivo geral, foram definidos os seguintes objetivos específicos:

1. Desenvolver os passos a serem realizados para caracterizar uma aplicação;
2. Determinar os processos para efetuar a caracterização da instância;
3. Descrever os procedimentos necessários para avaliar as caracterizações realizadas e selecionar a instância mais adequada para a aplicação;
4. Propor uma metodologia utilizando os procedimentos de caracterização e seleção;
5. Avaliar a execução de uma determinada aplicação em diversas instâncias previamente caracterizadas e, assim, validar a metodologia através deste estudo de caso;
6. Propor uma infraestrutura de interligação de diversos provedores de nuvens para viabilizar a interoperabilidade e ampliar a quantidade e variedade de instâncias.

1.3. METODOLOGIA

Inicialmente, realizou-se uma revisão bibliográfica dos conceitos e das principais investigações a serem feitas relacionadas a Computação em Nuvem. A interoperabilidade das aplicações no nível de plataforma é uma área que apresenta desafios, sendo de vital importância para uma maior utilização da Computação em Nuvem nas grandes empresas (ARMBRUST *et al.*, 2009).

Foram analisados os principais mecanismos de interoperabilidade no nível de Infraestrutura como Serviço (IaaS) para que se possa criar uma plataforma de desenvolvimento utilizando múltiplos provedores de infraestrutura.

Foi proposta uma arquitetura visando a interoperabilidade entre IaaS, e implementou-se parte dela.

Com a interligação entre os provedores de IaaS, aumentou-se a quantidade de instâncias disponíveis e também foram consideradas diferentes configurações. Contudo, persiste outro problema: qual instância escolher.

Realizaram-se experimentos para analisar a viabilidade de utilizar instâncias com disco de estado sólido (*Solid-State Drive* - SSD) nos provedores de Nuvem pública.

Verificou-se que o aumento do custo de instâncias com maior poder computacional é seguido pelo ganho de desempenho e mensurou-se quão melhor é o desempenho de uma aplicação quando executada em uma instância de mais poder computacional, utilizando os provedores de nuvem pública.

Foram efetuados experimentos com uma aplicação específica, em uma nuvem privada, a fim de analisar as suas características e comportamento.

Desenvolveu-se a metodologia CSE (Categorização, Seleção e Execução). Em seguida, aplicou-se a metodologia utilizando uma aplicação específica em nuvens públicas e privadas. Analisaram-se os resultados e demonstrou-se a qualidade da metodologia.

1.4. ESTRUTURA DA TESE

No capítulo dois, apresentam-se os conceitos relacionados à Computação em Nuvem, assim como exemplos de sistemas, problema de interligação e também formas de escalonamento em Nuvem. O capítulo três descreve a arquitetura e implementação do sistema de interligação de provedores de Nuvens. No capítulo quatro, é detalhada a proposta da metodologia de Categorização, Seleção e Execução (CSE). No capítulo cinco, são apresentados os resultados da utilização da Metodologia CSE com a aplicação OpenModeller e com provedores de Nuvem pública e privada. O capítulo seis apresenta as conclusões e trabalhos futuros.

2. COMPUTAÇÃO EM NUVEM

Este capítulo apresenta uma visão geral sobre a Computação em Nuvem e na primeira seção são apresentados os conceitos de Nuvem, assim como suas principais características, modelos de implantação, tipos de nuvens. Em seguida, na segunda seção, são abordadas as tecnologias de virtualização, provedores públicos de Infraestrutura como Serviço (IaaS) e as bibliotecas e os padrões de interoperabilidade de IaaS. A terceira seção aborda a análise de desempenho em aplicações, em recursos e suas metodologias. Por fim, são apresentados os problemas em Computação em Nuvem.

2.1. CONCEITOS

Existem várias definições para Computação em Nuvem (*Cloud Computing*), não havendo um consenso. Duas delas são:

> "Computação em Nuvem refere-se tanto às aplicações entregues com serviços utilizando a Internet quanto a hardware e software dos Centros de Dados que provêem esses serviços." (tradução de (ARMBRUST *et al.*, 2009)).

> "Nuvem é um tipo de sistema distribuído e paralelo consistindo em uma coleção de computadores interconectados e virtualizados que são dinamicamente providos e apresentados como um ou mais recursos de computação unificados baseados no acordo de nível de serviço estabelecido pela negociação entre o provedor de serviço e consumidor." (tradução de (BUYYA, R. *et al.*, 2009)).

A primeira definição tem um foco mais voltado para o serviço, o que será oferecido ao cliente, enquanto na segunda o foco é voltado para a infraestrutura, como o serviço será fornecido ao cliente.

2.1.1. Principais Características

A principal característica da Computação em Nuvem (*Cloud Computing*) está relacionada à elasticidade embutida em seu Modelo Econômico. Os recursos são alocados aos usuários à medida que são solicitados. Quando o usuário não precisar mais utilizar o recurso, este é facilmente desalocado e disponibilizado para outro usuário. Neste modelo, o cliente paga pelo que realmente utilizou.

As empresas mantêm atualmente uma grande infraestrutura tecnológica de recursos computacionais visando ao pico de utilização. A capacidade dos recursos está diretamente relacionada ao pico da demanda para determinados momentos.

Os picos de demanda são ocasionados em determinadas situações. Como exemplo, serviços de hospedagem de fotos, tais como Flickr, possuem um pico depois de feriados nacionais, dias dos namorados, natal (TAURION, C., 2009). Nessas ocasiões, as pessoas tiram mais fotos e querem disponibilizá-las o quanto antes.

Esses picos não ocorrem com freqüência e os recursos passam a ser subutilizados. Estima-se que a utilização dos servidores dos Centros de Dados (*Data Center*) esteja entre 5% e 20% (ARMBRUST *et al.*, 2009). A elasticidade permite uma melhor utilização dos recursos, sendo estes alocados à medida que forem necessários. Os recursos podem ser obtidos utilizando o serviço de uma empresa de Computação em Nuvem. Dessa forma, uma empresa não precisa manter uma infraestrutura tecnológica. Os custos relacionados aos riscos das tecnologias adotadas e gerenciamento dos recursos são transferidos para o fornecedor de serviço. Os custos de subutilização e saturação dos recursos também são transferidos. O foco do negócio é mantido.

A elasticidade permite que os custos relativos aos recursos sejam diminuídos Como exemplo (ARMBRUST *et al.*, 2009), um serviço pode precisar de 500 servidores para poder suportar o pico da tarde, 100 servidores para o pico da noite e a média de utilização durante o dia é de 300 servidores. Para manter o serviço atendendo à média de utilização diária, são necessários 300 servidores, 300 x 24 = 7200 servidores/hora. Entretanto, para suportar todos os picos seriam necessários

500 servidores, ou seja, 500 x 24 = 12000 servidores/hora. Caso o pico de carga não seja corretamente estimado, ocorre um subprovisionamento. O custo pode ser diminuído se os recursos forem alocados somente quando for necessário, ou seja, pode-se ter 300 servidores sendo utilizados e mais 200 servidores sendo alocados dinamicamente, no momento de pico. Outro fator a ser levado em conta nos custos é a depreciação do recurso.

Com relação à transferência de risco, oportunidades de negócios podem ser permanentemente perdidas por conta da negação de serviço ocasionada pelo subprovisionamento (*underprovisioning*). Se um serviço não suportar picos inesperados, oportunidades podem ser perdidas gerando má reputação, perda de cliente e “dinheiro”. A transferência de risco é permitida pela elasticidade. Como exemplo, suponha-se que houve uma demanda inesperada de acessos, sendo seguida por algumas horas, a um determinado serviço e os servidores que foram provisionados não são capazes de atender. A elasticidade permite que novos servidores sejam facilmente alocados, evitando que os novos clientes recebam negação de serviço e sejam permanentemente perdidos.

A empresa que fornece o serviço de Computação em Nuvem também obtém lucro com a grande quantidade de recursos. Ao comprar uma grande quantidade de recursos, consegue preços mais atrativos do que outras empresas. Os serviços de manutenção também custam menos em grande quantidade. A diversificação de clientes diminui o risco, pois se uma empresa falir ou não estiver precisando mais do serviço, outras empresas podem necessitar.

Estas características assemelham-se às fábricas de produção de *chips* (ARMBRUST *et al.*, 2009), onde não são todas as empresas que necessitam e têm condições de construir uma fábrica para a produção do seu chip. Essas empresas podem se focar na lógica do seu negócio, transferindo riscos (de tecnologia) e custos (de gerenciamento, aquisição) para a fábrica. As fábricas obtêm lucro com a diversificação e quantidade.

Um exemplo mais plausível de Computação Utilitária (*Utility Computing)* está relacionado ao fornecimento de energia elétrica. Antigamente, todas as indústrias tinham sua própria infraestrutura tecnológica para a produção de energia. Um dos

fatores de localização da fábrica estava relacionado ao ambiente propício para a produção de energia. A prestação de serviço de energia eliminou os custos que não são o foco de seu negócio, permitindo uma maior abrangência de localização da fábrica. Esse serviço acabou chegando às empresas menores e às pessoas comuns. Entretanto, problemas com a prestação do serviço de energia podem causar prejuízo para as empresas. Por isso, várias empresas possuem um fornecimento alternativo de energia para emergência com a utilização de geradores.

2.1.2. Modelos de Implantação

A prestação de serviços computacionais está ocorrendo com o Computação em Nuvem (*Cloud Computing*). As empresas querem diminuir os custos de seus negócios. Outra vantagem é que a Nuvem é um serviço, ou seja, paga-se pelo serviço após utilizá-lo (TAURION, C., 2009). Dessa forma, os gastos são postergados. Para pequenas empresas essa é uma forma mais viável de entrar no mercado. Essas empresas podem optar pela utilização de Nuvem Pública.

Nuvem Pública (ARMBRUST *et al.*, 2009)(TAURION, C., 2009) é o serviço de Nuvem que está disponível para o público em geral, no qual se paga pelo que se utiliza.

No entanto, empresas grandes que já possuem uma infraestrutura tecnológica e não querem que o núcleo do seu negócio seja mantido por outra empresa podem utilizar os benefícios da Nuvem ao optar pela Nuvem Privada.

Nuvem Privada refere-se a Nuvem que não está disponível para o público em geral. As empresas criam sua própria infraestrutura de Nuvem para controlar os seus dados, aplicações, níveis de serviços e determinam quem pode acessá-lo.

Nuvem Híbrida é a utilização de ambos os tipos de Nuvem, Pública e Privada. Como exemplo, uma empresa pode preferir ter um Nuvem Privada para o núcleo do negócio e utilizar o Nuvem Pública para os serviços do *call-center* e de *e-mails*, diminuindo assim os custos.

A Computação em Nuvem apresenta algumas limitações que ainda precisam ser vencidas, tais como fronteiras entre as leis de cada país, interoperabilidade, desempenho, dentre outras (ARMBRUST *et al.*, 2009).

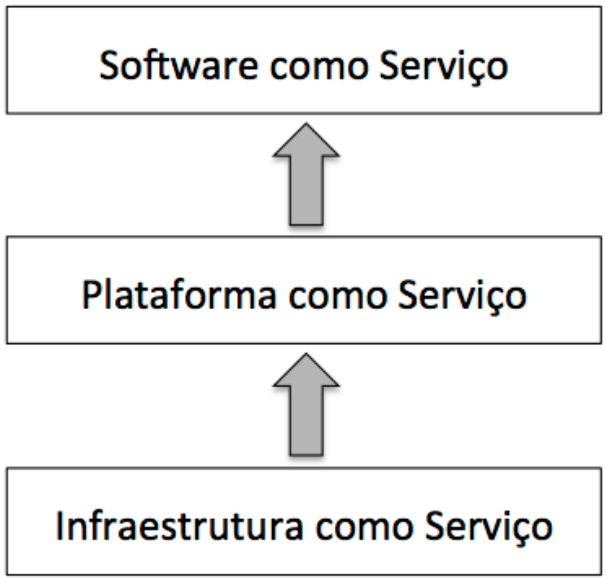
Por outro lado, Computação em Nuvem é massivamente escalável, podendo ser encapsulada como uma entidade abstrata que entrega diferentes níveis de serviço ao cliente, focado em economia de larga escala, transferindo riscos e reduzindo custos tanto para o cliente quanto para o fornecedor do serviço. Os serviços podem ser dinamicamente configurados (via virtualização ou de outra forma) e entregues sob demanda. Computação em Nuvem está fornecendo um modelo de computação desejado pela indústria, a Computação Utilitária (*Utility Computing*), por meio do qual paga-se pelo que se consome.

Na próxima seção são abordadas as Camadas de Serviços (IaaS, PaaS, SaaS) que podem ser fornecidas pela Computação em Nuvem.

2.1.3. Tipos de Serviços de Nuvem

Os serviços oferecidos pela Computação em Nuvem são comumente separados em Camadas. A Figura 1 expõe essas camadas. A camada de Infraestrutura como Serviço (*Infrastructure as a Service* - IaaS) provê um encapsulamento dos recursos, fornecendo ao cliente o controle total do recurso, sendo este obtido sob demanda. A camada de Plataforma como Serviço (*Platform as a Service* - PaaS) fornece um ambiente de desenvolvimento para que o usuário possa criar e executar sua própria aplicação na Nuvem. A camada de Software como Serviço (*Software as a Service* - SaaS) apresenta um modelo de fornecimento de software diferente do tradicional. Nesta camada, o usuário não paga por uma licença convencional, mas pelo tempo de utilização do software. A Internet é utilizada para acessar o serviço, assim, a aplicação não é instalada localmente pelo usuário. Nas próximas seções, cada uma dessas camadas serão detalhadas e vários sistemas são mostrados como exemplo.

Figura 1 - Camada de Serviços da Computação em Nuvem.



Fonte: Autoria própria.

Infraestrutura como Serviço (IaaS)

Infraestrutura como Serviço (*Infrastructure as a Service*, Iaas) tem como objetivo prover infraestrutura baseada em recursos virtuais (ou físicos) como produto para o cliente (VECCHIOLA; PANDEY; BUYYA, 2009). Quando o cliente "aluga" o recurso, o provedor de serviço fornece ao usuário total controle sobre o recurso. O hardware e software são entregues como um ambiente virtual, onde o usuário pode gerenciar o sistema operacional e desenvolver suas aplicações. A infraestrutura fornecida pelo provedor é paga pelo cliente de acordo com sua utilização e o tipo de serviço solicitado. Dentre os requisitos de hardware fornecidos pelo provedor tem-se memória, tipo e capacidade de CPU, armazenamento, tráfego de rede e dentre os requisitos de software tem-se sistema operacional, banco de dados, entre outros. O provedor de IaaS também pode fornecer diversos serviços, como balanceamento de cargas e elasticidade, dentre outros.

O usuário pode instalar os softwares *open source*, tais como banco de dados MySQL, servidor WEB Apache, como também pode utilizar uma imagem (JAMES MURTY, 2008) que contenha esses softwares. Ou ainda, o usuário também pode instalar softwares comerciais, tais como banco de dados Oracle e servidor IBM

WebSphere Portal Server utilizando sua própria licença ou pagar ao provedor pela utilização dos softwares comerciais, caso estes sejam fornecidos pelo provedor. Ao solicitar a utilização do software ao provedor, o usuário paga pelo tempo de uso, seguindo o modelo econômico da Nuvem.

O exemplo típico é o Amazon Web Services que fornece vários serviços (AMAZON, 2014d)(JAMES MURTY, 2008). Os principais são Amazon EC2 (*Elastic Compute Cloud*) (AMAZON, 2014a, p. 2)(JAMES MURTY, 2008) e Amazon S3 (*Simple Storage Service*) (AMAZON, 2014c)(JAMES MURTY, 2008) onde a infraestrutura de computação e de armazenamento são oferecidas ao público utilizando o modelo de preço de Computação Utilitária (*Utility Computing*). O Amazon EC2 fornece uma infraestrutura de computação e serviço baseada na virtualização do hardware. O usuário pode criar sua imagem (*Amazon Machine Images* - AMIs) (JAMES MURTY, 2008), salvá-la e utilizá-la para criar múltiplas instâncias a partir desta imagem. A imagem é composta pelo sistema operacional (Linux ou Windows) e pelas aplicações. Também são fornecidas imagens prontas para uso por outros usuários da comunidade ou pela própria Amazon. Caso o usuário queira utilizar o software IBM WebSphere, pode escolher a AMI de nome IBM WebSphere Application Server (32-bit), a ser hospedada no Leste dos EUA, cuja identificação (ID) é ami-316a8358. Ao criar uma instância a partir desta AMI, o usuário pagará pelo tempo de uso da instância (que pode ser *Standard*, *High-Memory*, *High-CPU* ou *Cluster Computing*) e pelo tempo de utilização do IBM Web Sphere. Outros serviços podem ser adicionados, tais como balanceamento de Cargas (*Elastic Load Balance*) para as instâncias e Auto-Dimensionamento (*Auto-Scaling*) para aumentar ou diminuir o número de instâncias de acordo com as condições que o usuário definir.

O Rackspace (RACKSPACE, 2014) era somente um servidor de hospedagem. Depois de adquirir Mosso, empresa provedora de Nuvem, tornou-se um provedor de Nuvem bastante popular devido à sua política de preços. Possui os serviços de Servidores em Nuvem (*Cloud Servers*) para a criação de instâncias virtuais e Arquivos em Nuvem (*Cloud Files*) para o armazenamento de dados. Permite criar instâncias de baixa configuração (256 MB de RAM e 10 GB de Disco) utilizando Linux como sistema operacional ao custo de 1,5 centavos de dólar, ou seja, 10,95

dólares por mês. Isso permite que os clientes façam testes com os seus serviços a baixo custo e recomendem posteriormente o Rackspace para sua empresa. Outra característica está relacionada a uma ótima documentação.

GoGrid (GOGRID, 2014) é um provedor de serviço de Nuvem que investe bastante em propaganda. Fornece Máquina Virtual de baixa configuração, entretanto seu custo é elevado se comparado ao Rackspace. O sistema operacional não influencia no valor da instância, ou seja, a utilização do Linux (que é distribuído gratuitamente), Windows ou Red Hat Enterprise acarreta no mesmo valor a ser pago por instância.

ElasticHosts (ELASTICHOSTS, 2014) permite a especificação da configuração das Máquinas Virtuais. Enquanto os demais provedores oferecem determinadas configurações para criação de instâncias, no ElasticHosts pode-se determinar qual configuração (CPU, memória, disco e transferência de dados) a instância terá. Pode-se mudar a configuração básica da instância, mudando o processador, a memória, o disco ou a largura de banda de acordo com suas necessidades. Nos outros provedores, é necessário criar uma nova instância.

Uma comparação entre o Amazon EC2, Rackspace, GoGrid e ElasticHosts é apresentada na seção 2.2.3. Outro exemplo de Nuvem Pública é FlexiScale (FLEXISCALE, 2014).

Exemplos de softwares que possibilitam a criação de Infraestrutura para a Nuvem Privada são: (a) Eucalyptus (NURMI *et al.*, 2009), que busca criar uma infraestrutura semelhante ao da Amazon; (b) OpenNebula (SOTOMAYOR *et al.*, 2009b), que permite a utilização de Nuvem Hibrida; (c) Nimbus (KEAHEY *et al.*, 2009), para criação de uma infraestrutura de Nuvem utilizando a infraestrutura da Grade, mais especificamente utilizando o Globus Toolkit (FOSTER; KESSELMAN, 1997)(FOSTER, 2006); (d) VMware vSphere (VSPHERE, 2014) (SOTOMAYOR *et al.*, 2009b); (e) OpenStack (OPENSTACK, 2014b); (f) Apache CloudStack (APACHE CLOUDSTACK, 2014) .

No nível de IaaS é possível criar um Cluster Virtual, ou seja, um conjunto de instâncias de Máquinas Virtuais com arquitetura de um Cluster. No entanto existem

alguns empecilhos, tais como overhead de E/S para escrita em disco, largura de banda de rede baixa para troca de mensagens. A Amazon está disponibilizando um serviço de cluster para Computação de Alto Desempenho (AMAZON, 2014b). Um instância possui 23 GB de memória, 33,5 EC2 Compute Units (2 processadores Intel Xeon X5570, arquitetura quad-core “Nehalem”), 1690 GB de disco virtual, plataforma de 64-bits, desempenho de E/S alta (10 Gigabit Ethernet). Esse serviço permite a execução de aplicações de alto desempenho que utilizem troca de mensagens, *Message Passing Interface* (MPI) (GROPP; LUSK; SKJELLUM, 1999).

Uma meta a ser alcançada pela computação de alto desempenho é a capacidade de conseguir executar uma aplicação paralela, com comunicação entre as tarefas, em Máquinas Virtuais, e assim, suspender e retornar as Máquinas Virtuais sem causar problemas na execução da aplicação. Outro propósito a ser buscado (SOTOMAYOR *et al.*, 2009a)(SOTOMAYOR; KEAHEY; FOSTER, 2008) está relacionado à criação de uma métrica para determinar com precisão o tempo que a máquina demora para suspender e reiniciar em conjunto com outras Máquinas Virtuais. Essa métrica é importante para a reserva avançada de recursos para a criação de Máquinas Virtuais em Nuvem Privada.

Plataforma como Serviço (PaaS)

Plataforma como Serviço (*Platform as a Service*, PaaS) provê uma aplicação ou uma plataforma de desenvolvimento na qual o usuário pode criar sua própria aplicação e executar dentro da Nuvem (VECCHIOLA; PANDEY; BUYYA, 2009). Fornece *framework* de aplicação e APIs que podem ser utilizados pelos desenvolvedores para programar ou compor aplicações na Nuvem.

Google App Engine (GOOGLE, 2014a)(VECCHIOLA; PANDEY; BUYYA, 2009) é uma plataforma para desenvolvimento de aplicações Web utilizando a infraestrutura do Google. Fornece APIs e modelo de aplicação para os desenvolvedores utilizarem serviços adicionais fornecidos pelo Google, tais como imagens, *e-mail*, dentre outros.

Azure (AZURE, 2014)(VECCHIOLA; PANDEY; BUYYA, 2009) é uma plataforma da Microsoft para o desenvolvimento de aplicações escaláveis na

Nuvem. Oferece ao desenvolvedor um ambiente de desenvolvimento, execução e controle. Permite o desenvolvimento na Plataforma .NET. Disponibiliza serviços como gerenciamento e execução de *Workflow*, orquestração de serviços web, acesso ao SQL Server, dentre outros. A aplicação é desenvolvida localmente e depois, via portal, é submetida e executada no Windows Azure.

Aneka (VECCHIOLA; CHU; BUYYA, 2009)(VECCHIOLA; PANDEY; BUYYA, 2009)(BUYYA, R. *et al.*, 2009)(CHU *et al.*, 2007), desenvolvido pela Manjarasoft, é uma plataforma para o desenvolvimento de aplicações distribuídas utilizando .NET. Possui o ambiente de execução orientado ao serviço que pode ser executado (*deployment*) em Máquina Virtual ou física. Permite que as aplicações possam utilizar diferentes modelos de programação.

CloudBees (STAX, 2014) permite o desenvolvimento de aplicações utilizando outras infraestruturas de Nuvens. É uma plataforma que oferece ao desenvolvedor serviços e ferramentas para construir, gerenciar e escalar as aplicações. Disponibiliza um ambiente que permite ao desenvolvedor construir, executar e escalar suas aplicações.

Force.com (SALESFORCE, 2014b)(TAURION, C., 2009) é uma plataforma para o desenvolvimento de aplicações. Oferece um ambiente de desenvolvimento com linguagem própria, chamado Apex Code, parecido com as linguagens Java e C#.

Software como Serviço (SaaS)

Software como Serviço (*Software as a Service*, SaaS) é a aplicação sendo fornecida como serviço aos usuários, utilizando o modelo paga pelo que usa (*pay-to-go*) (VECCHIOLA; PANDEY; BUYYA, 2009). Os usuários utilizam a Internet, geralmente o Navegador, para acessá-lo. O cliente não pode customizá-lo. Os dados são permanentemente armazenados em um servidor remoto, acessível via Internet. Estes dados também podem ser armazenados temporariamente nos dispositivos do cliente, tais como computadores pessoais, notebooks, celulares, entre outros.

No SaaS, o usuário passa apenas a utilizar o software, ficando a cargo do fornecedor a instalação, manutenção e atualizações (TAURION, C., 2009). O cliente não precisa preocupar-se com a tecnologia que o software vai operar. Neste modelo, quando o software não está sendo utilizado, o usuário não está pagando por ele. O valor do software está relacionado às suas funcionalidades e não à posse do produto. Existe diferença entre o SaaS e *Application Services Provider* (ASP). O SaaS relaciona-se à funcionalidade da aplicação; nele, o cliente não precisa ser "dono" do software e nem precisa instalá-lo. No ASP, o cliente adquire o software e instala em um provedor remoto.

Como exemplo, a SalesForce (SALESFORCE, 2014a)(TAURION, C., 2009) é empresa líder em prover Serviços CRM (*Customer Relationship Management*). Os Google Apps (GOOGLE, 2014a)(VECCHIOLA; PANDEY; BUYYA, 2009) são ferramentas de Escritório baseadas na Web e são hospedadas na infraestrutura do Google. Este fornece mais opções que os serviços gratuitos, tais como maior espaço no gmail e suporte por telefone.

2.1.4. Modelo Econômico

O Modelo Econômico de Nuvem é baseado no modelo em que se paga pelo que se usa. Em geral, uma instância é composta por unidade de processamento, memória e disco. A transferência de dados é outro serviço que pode ser cobrado. Os valores variam para cada provedor de Nuvem. Alguns oferecem transferência gratuita entre instâncias no mesmo provedor, outros cobram pela transferência entre instâncias do mesmo provedor de Nuvem localizadas em diferentes regiões, como por exemplo, a transferência de dados entre uma instância do Amazon dos EUA para o Amazon da Europa. Também pode ser cobrada a utilização do IP (público ou privado) fornecido pelo provedor, o Software utilizado e outros serviços, como Balanceamento de Cargas e Auto Dimensionamento.

Amazon fornece disponibilidade de 99,9% em seus serviços. Entretanto, como apresentado em (ARMBRUST *et al.*, 2009) o serviço pode vir a ficar sem acesso por algumas horas durante o ano. Uma possível solução é a integração entre múltiplos provedores de Nuvens. Esta integração também permite que a aplicação e os dados não fiquem restritos a uma determinada Nuvem.

Uma empresa que possua seu próprio ambiente de Nuvem pode querer que uma determinada aplicação que necessite de um alto poder computacional não fique restrita a Nuvem Privada para não comprometer sua infraestrutura. Assim, pode-se utilizar a Nuvem Pública para aumentar a disponibilidade da aplicação e reduzir a utilização dos recursos da Nuvem Privada.

Uma característica da Virtualização é o isolamento. Assim, se ocorrer problema em uma instância, a outra não será afetada. A escolha de várias instâncias ou uma instância de maior capacidade de processamento implica a escolha entre custo/disponibilidade. Utilizar vários provedores de Nuvens para aumentar a disponibilidade também contribui para essa questão. Isso implica um modelo econômico em que se paga não somente pela hora de uso do serviço, mas também pelo nível de disponibilidade requerida.

Um escalonamento para a criação das instâncias entre as Nuvens que vise ao Modelo Econômico e dê preferência à criação de instâncias na Nuvem que possui o menor preço pode auxiliar na disponibilidade. Como exemplo, a Nuvem com menor custo por instância obteria grande parte das instâncias. Quanto menor o custo da Nuvem, mais instâncias seriam criadas nele. Por outro lado, algumas instâncias seriam criadas na Nuvem cujas instâncias são mais caras. A Tabela 1 mostra exemplos de distribuição das instâncias para duas Nuvens, utilizando como base o custo da instância e o provimento de uma maior disponibilidade.

Tabela 1 – Exemplo de distribuição de instância entre duas Nuvens, baseada no custo e o provimento de disponibilidade igual a 1.

Caso	Preço Instância Nuvem A (R$/h)	Preço Instância Nuvem B (R$/h)	Instâncias Nuvem A (%)	Instâncias Nuvem B (%)
1	0,10	0,10	50	50
2	0,10	0,20	75	25
3	0,10	0,30	83	17
4	0,10	0,40	88	13
5	0,10	0,50	90	10
6	0,30	0,30	50	50
7	0,20	0,15	63	38
8	0,30	0,10	83	17
9	0,25	0,20	60	40
10	0,13	0,12	54	46

Fonte: Autoria própria.

O nível de disponibilidade implica acrescentar instâncias para aumentá-la. Por exemplo, se o usuário solicitar o nível de disponibilidade igual a 0, nenhuma instância extra será criada. Se o nível for igual a 1, uma instância extra será criada. Se for igual a 2, duas instâncias extras serão criadas. A instância será criada na Nuvem que tiver menor quantidade de instâncias. Utilizando o exemplo anterior, como a grande parte das instâncias é criada na Nuvem de menor custo a instância extra será alocada na Nuvem de maior custo.

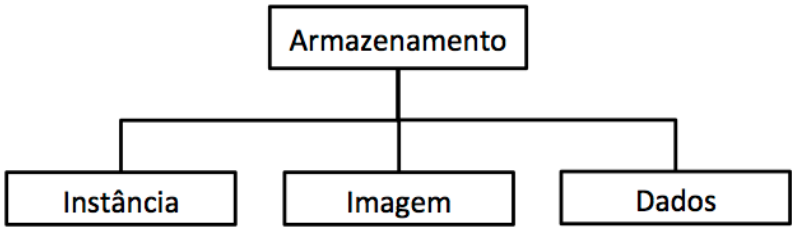
2.1.5. Tipos de Instância

Uma instância em virtualização é abstração de hardware. Na computação em nuvem, uma instância é uma abstração dos elementos de hardware de computador. As abstrações geralmente utilizadas pelos provedores de nuvem para caracterizar uma instância são: CPU, memória e armazenamento. Também podem ser atribuídos outros elementos, como por exemplo, dispositivo de rede. Em cada uma dessas abstrações é definida uma capacidade, *e.g.* 1 vCPU, 15 GB de memória, 160 GB de armazenamento local. Também pode ser definida a tecnologia da abstração, *e.g.* 320 GB de armazenamento em SSD. Um tipo de instância é um conjunto de abstrações de hardwares com sua capacidade definida. Cada provedor de nuvem possui diversos tipos de instância com seus próprios nomes, não havendo uma padronização entre os tipos.

2.1.6. Tipos de Armazenamento

No armazenamento em Computação em Nuvem, existem praticamente três classes de armazenamento, conforme Figura 2: (a) instância, é armazenada a imagem da Máquina Virtual em execução; (b) imagem, é armazenada a imagem que não está em execução, permitindo que se faça clone para criação das instâncias; (c) dados dos usuários e arquivos do sistema: os dados do usuário são os dados que o usuário pretende armazenar, e os arquivos de sistema são os arquivos de configuração do sistema operacional ou da infraestrutura.

Figura 2 - Classes de Armazenamento.



Fonte: Autoria própria.

O armazenamento em computação em nuvem pode ser efêmero ou persistente (GREGG, 2013). A Figura 3 apresenta os diagramas dos tipos de armazenamento em Nuvem. No efêmero, as instâncias são efêmeras e os dados são destruídos quando a instância é destruída, normalmente utiliza-se:

(a) armazenamento efêmero (local).

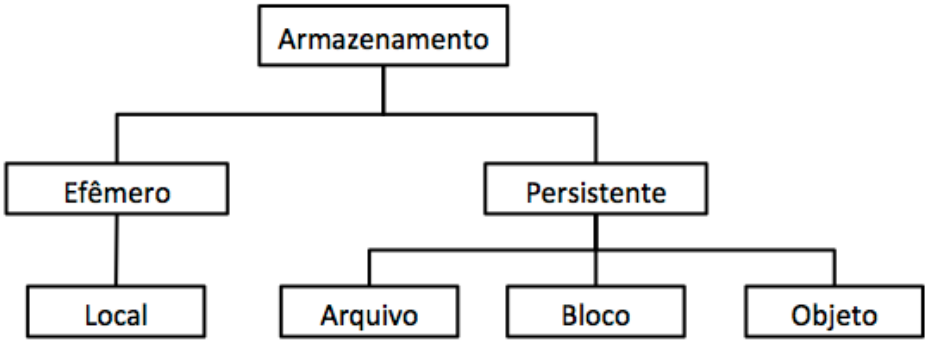
No persistente, existem três tipos de armazenamento em computação em nuvem, são eles (GREGG, 2013)(OPENSTACK, 2014b)(MESNIER; GANGER; RIEDEL, 2003):

(b) Armazenamento de Arquivo (*File Store*);

(c) Armazenamento de Bloco (*Block Store*); e

(d) Armazenamento de Objeto (*Object Store*).

Figura 3 – Diagrama do tipos de armazenamento em Nuvem.



Fonte: Autoria própria.

(a) Armazenamento Efêmero (Local)

Uma instância de nuvem efêmera geralmente é armazenada localmente utilizando os discos locais. Os dados locais também são temporários, e quando a instância é destruída, os dados também são destruídos (GREGG, 2013). Os dados são armazenados em uma estrutura hierárquica e são salvos em arquivos e diretórios (OPENSTACK, 2014b). O armazenamento local também é chamado de armazenamento em disco por alguns provedores de nuvem.

O armazenamento efêmero normalmente é utilizado para executar o sistema operacional e arquivos temporários. Em alguns provedores, pode ser também utilizado para executar aplicação de E/S de Alto Desempenho (*High Performance Computing*). O armazenamento é acessado através do sistema de arquivos do disco e sua acessibilidade é possível apenas dentro da instância. O tamanho do disco é determinado previamente pelo provedor da nuvem. Como exemplo de uso, têm-se 20 GB no primeiro disco, 50 GB no segundo disco.

(b) Armazenamento de Arquivo (File Store)

Os dados também são armazenados em uma estrutura hierárquica e também são salvos em arquivos e diretórios, entretanto os dados são persistentes (GREGG, 2013) (OPENSTACK, 2014b). Este tipo de armazenamento utiliza a interface do sistema de arquivo do sistema operacional. Os dados é acessado utilizando a rede de computadores e por protocolos, tais como, *Network File System* (NFS) e *Server Message Block* (SMB). É acessível de dentro da instância e a persistência do dado é determinada pelo usuário, ou seja, até o usuário apagar. O tamanho deste armazenamento também é determinado pelo provedor de nuvem.

O armazenamento de arquivo também pode armazenar a imagem da instância, tornando-a persistente. Entretanto, é pouco utilizado para esta finalidade devido ao seu desempenho. Este armazenamento é mais utilizado para o compartilhamento de imagens de Máquinas Virtuais (imagens que não estão em execução), permitindo que instâncias possam ser criadas a partir destas imagens.

Atualmente, este armazenamento não é apresentado ao usuário final na maioria dos provedores de nuvens.

(c) Armazenamento de Bloco (Block Store)

Os dados, as instâncias e as imagens são armazenados em volumes, também chamados de blocos (GREGG, 2013)(OPENSTACK, 2014b)(MESNIER; GANGER; RIEDEL, 2003). Normalmente, seu acesso é remoto utilizando protocolos, tais como *Fibre Channel over Ethernet* (FCoE) ou *Internet Small Computer System Interface* (iSCSI).

Este armazenamento pode ser utilizado para fornecer persistência a uma instância de Máquina Virtual. Para isto, é necessário definir no momento de criação da instância que esta será armazenada no tipo bloco.

Armazenamento em bloco também pode ser utilizado para armazenar dados de aplicações, como tabelas de banco de dados. Este armazenamento acessado através de um dispositivo de bloco, que pode ser particionado, formatado e montado. Para sua utilização, o usuário define um tamanho, cria um volume e anexa o volume na instância. Os dados são acessados a partir da instância e persiste até ser deletado pelo usuário. Seu tamanho é determinado pelo usuário no momento inicial da requisição do bloco e normalmente possui grandes tamanhos, *e.g.* 1 TB de disco. Exemplos comerciais são Amazon EBS e Rackspace Block Storage Volume.

(d) Armazenamento de Objetos (Object Store)

Um objeto (GREGG, 2013)(OPENSTACK, 2014b)(MESNIER; GANGER; RIEDEL, 2003) é uma coleção lógica de bytes que contém dado e metadado. O metadado é um conjunto de atributos (para descrever o objeto) e de políticas de segurança (para prevenção de acesso não autorizado). Cada objeto possui uma identificação única (id) e os objetos são armazenados em contêineres.

Enquanto um arquivo (file) tem metadados simples (nome do arquivo, criador, data de criação e tipo de arquivo), os metadados do objeto contêm mais informações sobre o contexto e o conteúdo dos dados. Como exemplo, um objeto de um exame médico teria atributos relacionados ao ID do objeto, tipo de arquivo, nome do paciente, diagnóstico, procedimento, data do procedimento do exame, nome do médico que fez o exame, nome do médico que solicitou o exame. Também são adicionadas as permissões de segurança, como por exemplo, se o objeto pode ser acessado por qualquer usuário.

Estes metadados permitem que objeto possa ser compartilhado com mais facilidade e com uma redução do *overhead* de gerenciamento. Também permite que os objetos possam ser escalados facilmente. Os objetos também podem ser replicados em múltiplos servidores e com localizações diferentes, aumentando a confiabilidade. Sendo assim, os objetos podem ser acessados de qualquer lugar. O acesso pode ser realizado por uma API, normalmente baseada em HTTP (*e.g.* SOAP API ou REST API).

As interfaces para a manipulação do objeto são bem conhecidas, como por exemplo, criar e deletar os objetos, escrever e ler bytes. Essa abstração permite que os dados possam ser acessados através de diferentes plataformas de sistemas operacionais.

O objetos podem ser comparados ao estacionamento com manobrista, onde se entrega a chave para o manobrista, este fornece uma identificação e é o responsável por estacionar o carro. O proprietário do veiculo não sabe onde carro foi estacionado. Para obter o carro de volta, o proprietário fornece a identificação ao manobrista, que lhe entrega o carro.

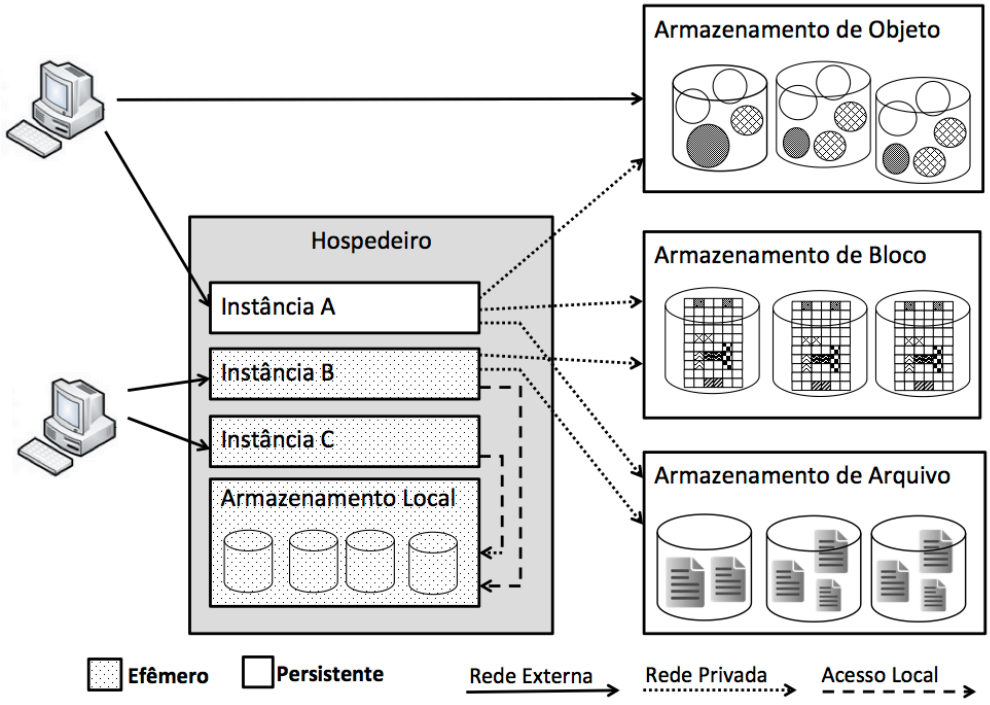
Os objetos persistem até serem deletados pelo usuário. São normalmente utilizados para dados que não sejam frequentemente atualizados, como backups, arquivos de vídeos, arquivos de áudio. Também pode ser utilizado para armazenar o estado de uma instância (*snapshot*). São utilizados também para armazenamento de grande quantidade de dados, como 10 TB. Como exemplos comerciais, têm-se Amazon S3 e Rackspace Cloud Files.

Arquitetura

A Figura 4 apresenta a arquitetura dos tipos de armazenamento em Nuvem para os dados e suas formas de acesso. Instância A é persistente e pode acessar (para leitura ou gravação) somente os dados persistentes. Estes dados podem ser armazenados em arquivo, bloco ou objeto. Para o usuário acessar os dados que estão armazenados em bloco ou arquivo, é necessário acessar primeiramente a instância via uma rede externa (*e.g.* Internet), para em seguida acessar os dados via uma rede interna do provedor. No armazenamento de objeto, os dados podem ser acessados diretamente por uma rede externa (*e.g.* Internet), ou acessados pela rede interna do provedor.

Para armazenamento de dados, em geral, não se utiliza o armazenamento em arquivos. A persistência da instância é comumente realizada em armazenamento de bloco. Para realizar um snapshot da instância, ou seja, salvar o estado da instância para, por exemplo, backup, o armazenamento do tipo objeto é o mais utilizado.

Figura 4 – Arquitetura dos tipos de armazenamentos em Nuvem para os dados e suas formas de acesso.



Fonte: Autoria própria.

As instâncias B e C são efêmeras, significa que elas estão armazenadas localmente. Os dados armazenados nestas instâncias também são efêmeros. O acesso aos dados efêmeros são realizados via o acesso local ao disco. Os dados destas instâncias também podem ser armazenados de maneira persistente, utilizando o armazenamento de arquivo, bloco ou objeto. Nota-se que, para o armazenamento local, existe concorrência das instâncias pelo acesso local, e para o armazenamento persistente existe concorrência das instâncias pela rede interna.

2.1.7. Custo

A diversidade de serviços, seja de computação ou de armazenamento, aumenta a dificuldade de calcular o custo total de utilização de uma instância. Cada nuvem possui diferentes preços para os serviços e também diferentes configurações. Em alguns provedores, não há cobrança pela transferência de dados interna, em outras o preço de cada requisição de E/S deve entrar no cálculo de custo. De maneira geral, para calcular o custo total de utilização do serviço de instância, deve-se levar em consideração o custo de computação, de armazenamento e de transferência de arquivo usando a rede. Sendo assim, temos:

$$C_{TOTAL} = C_{COMP} + C_{ARMAZ\_T} + C_{REDE}$$

onde C_{TOTAL} é o custo total, C_{COMP} é o custo de computação, C_{ARMAZ_T} é o custo de armazenamento total e C_{REDE} é o custo de rede para transferência de dados.

No custo de computação (C_{COMP}), é levado em consideração o preço do tipo de instância selecionada e o tempo de utilização desta instância. Lembrando novamente, o tipo de instância é composto por vCPU, memória e armazenamento local. O cálculo do custo de computação é apresentado como:

$$C_{COMP} = P_{INST} * TEMPO$$

onde P_{INST} é o preço do tipo instância e TEMPO é o tempo de utilização da instância.

O provedor de nuvem pode fornecer diversos serviços de armazenamento. O custo do armazenamento local já está incluído no preço da instância. Em alguns provedores, o tipo de instância é mais caro em virtude da tecnologia utilizada no armazenamento local, *e.g.* tecnologia SSD. Sendo assim, esse custo é transferido do custo de armazenamento total (C_{ARMAZ_T}) para o custo de computação (C_{COMP}), já que o custo de computação inclui o armazenamento local. Entretanto, o armazenamento local é efêmero: os dados são perdidos quando a instância é destruída.

O **custo de armazenamento total (C_{ARMAZ_T})** está relacionado ao armazenamento persistente. No armazenamento persistente, cada tipo de armazenamento possui o seu preço, variando de provedor para provedor.

Para calcular o custo total do armazenamento, deve-se levar em consideração o custo de armazenamento (C_{ARMAZ}) e o custo de requisições (C_{REQ}), obtendo assim:

$$C_{ARMAZ\_T} = C_{ARMAZ} + C_{REQ}$$

onde, C_{ARMAZ} é o custo de armazenamento e C_{REQ} é o custo de requisições realizadas no armazenamento.

O custo de armazenamento (C_{ARMAZ}) está relacionado ao preço do tipo de armazenamento (P_{TIPO_ARMAZ}), tamanho do volume de armazenamento (TAM) e o tempo de utilização (TEMPO), tendo assim:

$$C_{ARMAZ} = P_{TIPO\_ARMAZ} * TAM * TEMPO$$

O tipo de armazenamento selecionado pode ser arquivo, bloco ou objeto. Geralmente é utilizado o tipo bloco no armazenamento persistente. O preço do tipo de armazenamento (P_{TIPO_ARMAZ}) também pode variar de acordo com a tecnologia utilizada, *e.g.* HDD ou SSD, ou pela Qualidade de Serviço (*Quality of Service - QoS*), *e.g.* garantir uma quantidade de operações de E/S por segundos (IOPS).

O custo de requisição (C_{REQ}) pode ser definido, de maneira geral, da seguinte forma:

$$C_{REQ} = P_{REQ} * N_{REQ}$$

onde P_{REQ} é o preço de cada requisição e N_{REQ} é o número de requisições realizadas. Esta é uma fórmula geral pode ser adaptada de acordo com a forma de calcular de cada provedor. Como exemplo, o provedor pode definir um preço por um conjunto de requisições, também pode ter um preço mensal para garantir um QoS para um conjunto de requisições. Logo, essa fórmula pode ser adaptada para a política de preços aplicada por cada provedor.

Apesar de as requisições estarem normalmente relacionadas à rede, o custo de requisição (C_{REQ}) foi atribuído ao armazenamento, pois, de maneira geral, este custo é cobrado quando ocorre requisição no armazenamento persistente. Dessa forma, não havendo serviço de armazenamento persistente, o custo de requisição não é cobrado pelos provedores.

Sendo assim, o cálculo geral do custo de armazenamento total é definido como:

$$C_{ARMAZ\_T} = C_{ARMAZ} + C_{REQ}$$

$$C_{ARMAZ\_T} = (P_{TIPO\_ARMAZ} * TAM * TEMPO) + (P_{REQ} * N_{REQ})$$

Basicamente, existem três tipos de transferências de dados na nuvem, sendo eles: de entrada na nuvem (*data-in*), de saída da nuvem (*data-out*) e interna na nuvem (entre as instâncias das nuvens, ou a instância e outros serviços oferecidos pelo provedor de Nuvem). De maneira geral, as transferências de dados de entrada e interna na nuvem não são cobradas pelos provedores.

O custo de transferência está relacionado ao preço por dados transferidos vezes a quantidade de dados transferidos.

Novamente, observe-se que o custo de requisição (que faz parte do custo de armazenamento) está relacionado à quantidade de requisições, enquanto o custo de rede (C_{REDE}) está relacionado à quantidade de dados transferidos internamente ou externamente.

O **custo de rede (C_{REDE})** é calculado por esses três tipos de transferências, sendo definido como:

$$C_{REDE} = P_{TRAN\_ENTRADA} * TAM_{DADOS\_ENTRADA} + P_{TRAN\_SAIDA} * TAM_{DADOS\_SAIDA} + P_{TRAN\_INTERNA} * TAM_{DADOS\_INTERNO}$$

onde $P_{TRAN\_ENTRADA}$ é o preço de transferência de entrada, $TAM_{DADOS\_ENTRADA}$ é o tamanho dos dados de entrada, P_{TRAN_SAIDA} é o preço de transferência de saída, TAM_{DADOS_SAIDA} é o tamanho dos dados de saída, $P_{TRAN\_INTERNA}$ é o preço de transferência de interna, $TAM_{DADOS\_INTERNA}$ é o tamanho dos dados de internos.

Para calcular o custo total, têm-se:

$$C_{TOTAL} = C_{COMP} + C_{ARMAZ\_T} + C_{REDE}$$

$$C_{TOTAL} = (P_{INST} * TEMPO) + (P_{TIPO\_ARMAZ} * TAM * TEMPO + P_{REQ} * N_{REQ}) + (P_{TRAN\_ENTRADA} * TAM_{DADOS\_ENTRADA} + P_{TRAN\_SAIDA} * TAM_{DADOS\_SAIDA} + P_{TRAN\_INTERNA} * TAM_{DADOS\_INTERNO})$$

Com relação ao armazenamento, existe também o *snapshot*, ou seja, salvar o estado de uma instância para uso posterior. O *snapshot* pode ser utilizado para backup ou para criar clone de novas instâncias iguais. Normalmente, a imagem gerada pelo *snapshot* é armazenada em objeto. O cálculo do custo do *snapshot* (C_{SNAP}) é definido de acordo com:

$$C_{SNAP} = P_{SNAP} * TAM * TEMPO$$

onde, P_{SNAP} é o preço de armazenamento da imagem de *snapshot*, TAM é o tamanho e TEMPO é o tempo de armazenamento da imagem.

Caso o custo de snapshot seja levado em consideração, este custo poderia ser adicionado em custo de armazenamento total (C_{ARMAZ_T}). Outros tipos de serviços relacionados ao armazenamento de dados, como serviço de banco de dados, também poderiam ser adicionados ao custo de armazenamento total. Outros custos de serviços relacionados à rede, *e.g.* serviço de VPN, podem ser adicionados no custo de rede (C_{REDE}).

A análise detalhada do custo será utilizada na seção de aplicação da metodologia CSE.

2.1.8. Escalonamento

Pode-se citar ao menos três tipos de escalonamento em Computação em Nuvem: (a) escalonamento de recursos entre Máquinas Virtuais (*Virtual Machines*, VMs) localizadas em uma mesma máquina (*e.g.* utilização da memória entre duas MV); (b) escalonamento de Máquinas Virtuais e/ou aplicações entre máquinas reais (*e.g.* entre diferentes computadores do cluster); (c) escalonamento de Máquinas Virtuais e/ou aplicações entre diferentes provedores de Nuvens (BUYYA, R. *et al.*, 2009)(VECCHIOLA; PANDEY; BUYYA, 2009).

As Máquinas Virtuais que estão em uma mesma máquina real estão competindo pelos recursos da máquina real, tais como memória, CPU, acesso ao disco. Nesse nível, o escalonamento está relacionado à Máquina Virtual que irá utilizar os recursos, e quanto do recurso poderá utilizar. Como exemplo, pode-se aumentar o tamanho da memória de uma Máquina Virtual dinamicamente enquanto alguma outra máquina não está utilizando.

O escalonamento de Máquinas Virtuais entre máquinas reais (*hosts*) ocorre dentro do provedor de Computação em Nuvem. A Máquina Virtual é escalonada no conjunto de máquinas do provedor. Em geral, as entidades externas não têm

conhecimento sobre qual tipo de escalonamento está sendo aplicado dentro da Nuvem. Algumas características podem ser levadas em conta, como por exemplo, qual usuário solicitou o serviço e se este usuário possui mais de uma Máquina Virtual. Um critério de escalonamento é a alocação de Máquinas Virtuais em uma mesma máquina ou em máquinas interconectadas com baixa latência, para as Máquinas Virtuais do mesmo usuário.

No escalonamento de Nuvem ocorre uma interação entre o usuário e o *Broker* ou escalonador. O usuário comunica-se com o *Broker*/escalonador informando algumas de suas escolhas, tais como preço e tempo de uso. O *Broker*/escalonador fica responsável por escolher para qual provedor de Nuvem a Máquina Virtual será criada. Este tipo de escalonamento pode ser feito escalonando as Máquinas Virtuais e/ou aplicações. Para o escalonamento utilizando Máquinas Virtuais é necessário que haja uma padronização da Máquina Virtual. Existe uma tentativa de criação deste padrão, o Formato Virtual Aberto (*Open Virtual Format* - OVF). Assim, a instância da Máquina Virtual poderia migrar para outro provedor de Nuvem. Esse tipo de escalonamento possui atualmente dois problemas: o tempo de migração da imagem da Máquina Virtual entre os provedores devido ao seu tamanho e falta de padronização entre os provedores.

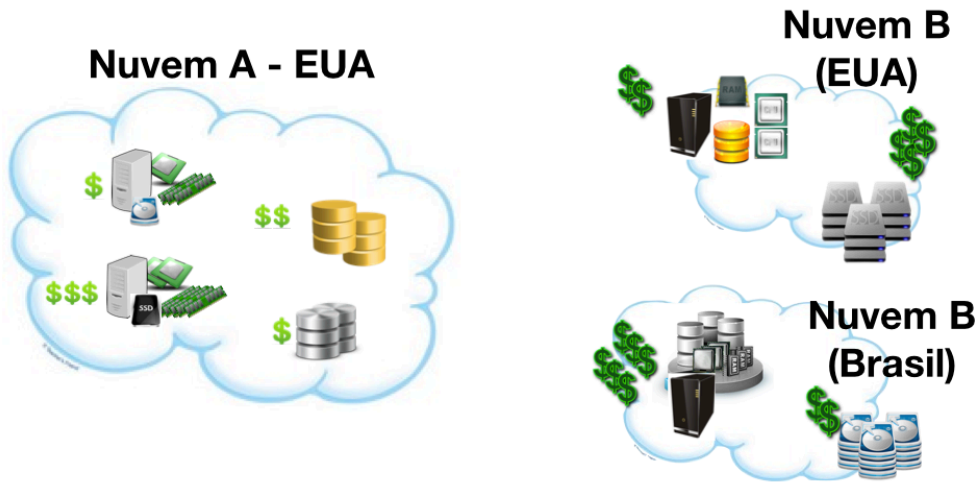
Outro problema está relacionado aos programas paralelos que estão sendo executados nas Máquinas Virtuais. Uma mensagem MPI pode não chegar ao destino e ser perdida, dependendo do modo como as máquinas são suspensas ou retomadas. Como exemplo, consideremos duas Máquinas Virtuais que possuem a mesma aplicação paralela e trocam mensagens entre si. Deste modo, uma thread da aplicação de uma Máquina Virtual A envia uma mensagem para uma thread da aplicação da Máquina Virtual B. Após a máquina A enviar uma mensagem, ambas as Máquinas Virtuais recebem “ordem” de que devem ser suspensas. As máquinas entram no estado “suspenso”. Devido ao tempo necessário para uma mensagem chegar ao seu destino, a aplicação da Máquina Virtual B pode não receber esta mensagem, por estar no estado “suspenso” e, assim, ocorrer a perda da mensagem.

O escalonamento entre provedores de Nuvem pode ser realizado no nível da aplicação, utilizando contêiner. Contêiner é projetado para ajudar a organizar e

armazenar itens que são colocados dentro dele (ECKEL, BRUCE, 2006). Contêiner é um componente que pode conter outros componentes. Uma classe contêiner é uma classe projetada para organizar múltiplas instâncias de outras classes. Um contêiner pode implementar um ambiente para a execução dos serviços. Desse modo, sua principal atribuição é inicializar os serviços quando for necessário. O contêiner fica hospedado nas MVs da Nuvem para executar e monitorar a MV e a aplicação. Esses serviços são responsáveis por executar e gerenciar as aplicações. Assim, as aplicações podem ser facilmente migradas entre diferentes Máquinas Virtuais (seja da mesma Nuvem ou de Nuvens diferentes). O problema de sincronização de mensagem pode ser realizado utilizando o checkpoint da aplicação, caso ela forneça suporte.

A Figura 5 mostra o problema quando há muitos provedores de Nuvem. Cada provedor tem diferentes instâncias, tipos de armazenamento e preços. O provedor de Nuvem pode estar estabelecido em diferentes localizações geográficas. Por exemplo, o Nuvem B pode estar localizado nos Estados Unidos (EUA) e no Brasil.

Figura 5 – Diferentes configurações de provedores de Nuvem.



Fonte: Autoria própria.

Uma instância é considerada como um servidor que executa aplicações. Ela possui CPU virtual (vCPU), memória e local de armazenamento. Cada instância é oferecida a diferentes preços.

Utilizemos o exemplo a seguir: Nuvem A tem uma instância com 4 GB de memória, 2 vCPU e 200 GB de disco local SSD e custa 2 reais. Nuvem B – Brasil tem uma instância de 64 GB de memória, 4 vCPU e 1 TB de disco local HDD e custa 4 reais. Uma breve análise e comparação entre estas instâncias nos apresenta alguns pontos a considerar. Qual instância escolher para executar a aplicação? Outro questionamento: se dobrar o custo, também será dobrado o desempenho? Essas características podem ser levadas em consideração no escalonamento em Nuvem?

O escalonamento pode levar em consideração as características de Armazenamento (*Storage*). Este pode ser local ou remoto. E há diferentes tecnologias, tais como HDD ou SSD. Sendo assim, muitas questões podem ser feitas sobre as configurações. Esse é um problema antigo em escalonamento de computadores, mas é necessário investigar utilizando as características da Nuvem. Uma característica da Nuvem é a tenacidade (*tenacity*), em que usuários podem estar compartilhando o mesmo disco, supondo ser apenas deles e que o disco não está sendo compartilhado com outros usuários.

2.1.9. Nuvem Versus Grade

Os termos Computação em Grade (*Grid Computing*) e Computação em Nuvem (*Cloud Computing*) não devem ser considerados semelhantes. Ambos provêem da Computação Utilitária (*Utility Computing*), fornecem serviços metrificados. As diferenças entre Aglomerado (*Cluster*), Grade (*Grid*) e Nuvem (*Cloud*) são expostas pelas definições de (BUYYA, R. *et al.*, 2009) (FOSTER *et al.*, 2008):

> “Cluster é um tipo de sistema paralelo e distribuído, o qual consiste em uma coleção de computadores interconectados trabalhando em conjunto como um único recurso computacional integrado” (tradução de (BUYYA, R. *et al.*, 2009)).

> “Computação em Grade é um sistema que coordena recursos que não estão sujeitos a controle centralizado, usando padrões abertos, protocolos e interfaces de propósito geral para entregar serviços de qualidades não triviais.” (tradução de (FOSTER *et al.*, 2008)).

> “Computação em Grade é um tipo de sistema paralelo e distribuído que permite o compartilhamento, seleção e agregação dos recursos ‘autônomos’ geograficamente distribuídos dinamicamente em tempo de execução (*runtime*) dependendo de sua disponibilidade, capacidade, custo, e usuários com requerimentos de qualidade de serviços” (tradução de (BUYYA, R. *et al.*, 2009)).

> “Nuvem é um tipo de sistema distribuído e paralelo consistindo em uma coleção computadores interconectados e virtualizados que são dinamicamente providos e apresentados como um ou mais recursos de computação unificados baseados no acordo de nível de serviço estabelecido pela negociação entre o provedor de serviço e consumidor” (tradução de (BUYYA, R. *et al.*, 2009)).

Em Computação em Grade (FOSTER; KESSELMAN; TUECKE, 2001)(FOSTER *et al.*, 2002)(FOSTER *et al.*, 2008), o controle dos recursos pertence ao domínio. Essa característica reflete no software local e na política de administração. Essas escolhas não são sempre úteis para usuários que precisam de diferentes sistemas operacionais, diferentes softwares. Uma grande diferença da

Nuvem: quando o usuário aluga o recurso, o provedor de serviço fornece o controle para ele (KEAHEY *et al.*, 2009).

Tanto a Nuvem quanto a Grade buscam reduzir o custo computacional, aumentar a confiabilidade e flexibilidade (FOSTER *et al.*, 2008). Os problemas são praticamente os mesmos em Nuvem e Grade, tais como implementar Computação de Alto Desempenho (*High Performance Computing* – HPC), definir métodos pelos quais consumidores encontram provedores de serviços, dentre outros. Entretanto, Nuvem e Grade operam em escalas diferentes.

Computação em Nuvem é massivamente escalável, pode ser encapsulado como uma entidade abstrata que entrega diferentes níveis de serviços para o consumidor fora da Nuvem, e os serviços podem ser dinamicamente configurados e entregues sob demanda (FOSTER *et al.*, 2008).

Enquanto a Grade fornece uma infraestrutura que entrega recursos computacionais e de armazenamento, a Nuvem entrega recursos e serviços abstratos (FOSTER *et al.*, 2008). A Nuvem pode utilizar a Grade como *backbone* e como uma infraestrutura de suporte, como por exemplo, uma infraestrutura para o gerenciamento dos recursos em diferentes domínios administrativos.

No modelo de negócio da Nuvem, o consumidor irá pagar ao provedor o que consumiu. Em Grade, o usuário tem um certo número de unidades de serviços, geralmente CPU/horas, para poder gastar (FOSTER *et al.*, 2008). Existem esforços para construir uma economia na Grade (BROBERG; VENUGOPAL; BUYYA, 2008).

A arquitetura da Grade foca-se na integração de recursos existentes, tais como hardware e software. As políticas de gerenciamento e segurança dos recursos locais continuam sob o controle do domínio administrativo. A Nuvem é composta por um grande conjunto de recursos de computação e/ou armazenamento que são acessados via um protocolo específico, utilizando a Internet. Em Nuvem, ainda não existe um padrão para as interfaces de cada nível de serviços (IaaS, PaaS e SaaS), inviabilizando a interoperabilidade entre as diversas Nuvens.

Uma grande característica da Nuvem é a abstração dos recursos (computacionais, de rede, de armazenamento, dentre outros) e o encapsulamento da aplicação (por exemplo, configuração e inicialização) por meio da virtualização (FOSTER *et al.*, 2008). As vantagens da virtualização são que: (a) várias aplicações podem ser executadas no mesmo servidor e, assim, os recursos podem ser utilizados de maneira mais eficiente; (b) as configurações de várias aplicações podem ser completamente diferentes; (c) há um aumento da disponibilidade da aplicação, já que o ambiente virtual pode ser facilmente migrado sem a interrupção do serviço. A Grade não utiliza tanto a virtualização como a Nuvem, mas há projetos, como o Nimbus, para utilizar virtualização sobre a Grade.

Grade tem suporte para diversos tipos de aplicações, que vão de Computação de Alto Desempenho (*High Performance Computing* - HPC) a Computação de Alta Vazão (*High Throughput Computing* - HTC). As aplicações de alto desempenho podem ser fortemente acopladas, sendo executadas normalmente em máquinas interconectadas com baixa latência e não são executadas na rede da Grade, como também podem ser fracamente acopladas, sendo executadas e gerenciadas por sistemas de *Workflows*.

Computação em Nuvem apresenta muitos pontos a serem explorados, seja na parte de monitoramento, seja na localização dos dados (os dados podem estar no mesmo disco local ou em outro disco conectado pela rede).

Aplicações de alto desempenho para Computação em Nuvem, que envolvem muita E/S, ainda não obtiveram muito sucesso em razão da grande latência gerada por conta da virtualização da rede (WALKER, 2008) e da localização dos dados. O escalonamento de Máquinas Virtuais em um mesmo computador, ou em computadores interconectados com baixa latência, para aplicações fortemente acopladas é uma proposta de incorporação de aplicações HPC para o ambiente de Nuvem.

Algumas características de Nuvem e Grade são comparadas e apresentadas na Tabela 2. O controle dos recursos fornecidos ao usuário, ou seja, instalação de programas e bibliotecas é restrito na Grade. Quem detém o controle do recurso é o administrador do domínio da Grade. Caso o usuário precise utilizar uma aplicação

que necessite o acesso de administrador e/ou necessite de uma determinada versão que não esteja instalada, é necessário solicitar permissão de acesso ou solicitar a instalação via administrador da Grade. Como cada domínio detém uma política administrativa, o processo de solicitação pode ser burocrático e a permissão pode ser negada em alguns domínios. Em Nuvem, no nível de IaaS, é fornecida uma abstração do recurso, onde o usuário detém total controle deste recurso, oferecendo assim uma maior flexibilidade de utilização ao usuário.

Tabela 2 - Comparação entre Computação em Nuvem e Computação em Grade.

Característica	Grade	Nuvem
Controle dos Recursos Fornecidos	Domínio	Usuário
Infraestrutura	Recursos Computacionais e de Armazenamento	Recursos e Serviços Abstratos. Pode utilizar a Grade para gerenciar
Modelo de Negócio	Colaborativo	Paga pelo que consome
Arquitetura	Integração de recursos existentes (compartilhamento) Domínio Administrativo	Conjunto de recursos de computação e armazenamento. Controle Central
Interfaces/ Interoperabilidade	Abertas/ Alta	Proprietárias/Fraca
Benefício/Malefício	Colaboração/ Complexidade	Elasticidade/Segurança
Quem utiliza? / Quem financia?	Pesquisadores / Governo	Pequena e médias empresas / Capital próprio

Fonte: Autoria própria.

A infraestrutura utilizada em Computação em Grade fornece recursos computacionais físicos (ou virtuais) ao usuário, interconectando vários processadores distribuídos geograficamente e fornecendo baixa E/S de disco. A Grade tem o intuito de fornecer grande capacidade de computação para aplicações de alto desempenho. A Nuvem é baseado em recursos e serviços abstratos. Utiliza o encapsulamento dos recursos, em geral, utilizando a virtualização. Os benefícios do encapsulamento foram discutidos anteriormente. Em geral, o *overhead* de disco é alto e a largura de banda da rede é baixa (ARMBRUST *et al.*, 2009), não sendo adequado para aplicações de E/S, que utilizem bastante o disco ou troca de

mensagem. Atualmente, a Amazon EC2 está oferecendo um serviço para Computação de Alto Desempenho (AMAZON, 2014b). A infraestrutura da Nuvem pode utilizar a infraestrutura de Grade para gerenciar os recursos. Como exemplo, o toolkit Nimbus (KEAHEY *et al.*, 2009) utiliza o gerenciamento e certificados de segurança de uma infraestrutura de Grade (Globus Toolkit (FOSTER; KESSELMAN, 1997)(FOSTER, 2006)) para fornecer o serviço de infraestrutura de Nuvem (IaaS).

O modelo de negócio em Grade é baseado na colaboração dos usuários, onde os recursos são compartilhados entre os usuários que fazem parte da Grade. Em geral, a infraestrutura da Grade é mantida por instituições de pesquisa governamentais. A Nuvem assemelha-se a Computação Utilitária (*Utility Computing*), onde o usuário paga pelo serviço que utilizou. Em geral, os usuários são pequenas e médias empresas.

A arquitetura da Grade é baseada na integração de recursos existentes, distribuídos geograficamente, visando ao compartilhamento destes. Cada domínio administrativo possui suas políticas de gerenciamento dos recursos. Grade possibilita a criação de Organizações Virtuais (*Virtual Organizations* - VO), ou seja, permite que determinados recursos, em diferentes domínios, sejam utilizados por uma determinada organização não física. Em Nuvem, os recursos de computação e armazenamento possuem um controle central.

Uma das grandes características da Grade é a utilização de interfaces abertas a fim de fornecer interoperabilidade entre os diversos domínios. Cada provedor de Nuvem possui sua própria interface de programação, tornando a interoperabilidade fraca entre estes provedores e aprisionando os usuários/clientes.

A colaboração entre os pesquisadores, no compartilhamento dos recursos, é uma vantagem da utilização da Grade. Entretanto, criar, manter e gerenciar sua infraestrutura envolve uma grande complexidade. A elasticidade e o modelo econômico (pagar pelo que usa) tornam o Nuvem atrativo para as empresas. Alguns autores questionam a segurança da Nuvem, contestando que as falhas relacionadas à segurança não foram bem exploradas (FOSTER *et al.*, 2008).

Os utilizadores do Grade são, em grande parte, pesquisadores financiados pelo governo. Existem empresas que também financiam projetos relacionados a Grade. Em Nuvem, as pequenas e médias empresas estão utilizando-o devido ao seu baixo custo inicial e por pagar depois de utilizar o serviço, ocorrendo assim a transferência do custo inicial (capital) para o custo do produto (*capex*) (TAURION, C., 2009).

2.2. TECNOLOGIAS, PROVEDORES E PADRÕES DE IAAS

Esta seção tem como objetivo apresentar as tecnologias envolvidas na virtualização e nos gerenciadores de infraestrutura de Nuvem. Em seguida, é apresentado alguns provedores de nuvem pública vigentes no mercado. Por fim, são apresentadas as bibliotecas e os padrões de interoperabilidade que estão disponíveis atualmente.

2.2.1. Virtualização

As Máquinas Virtuais começaram a ser utilizadas comercialmente nos *mainframes* da IBM em 1972, permitindo que cada Máquina Virtual execute seu próprio sistema operacional. Na década de 1990, a Intel incorporou características de virtualização no processador 80x86. O domínio da tecnologia de virtualização deixou de ser somente da IBM e acabou permitindo a popularização a virtualização (SILBERSCHATZ; GALVIN; GAGNE, 2013).

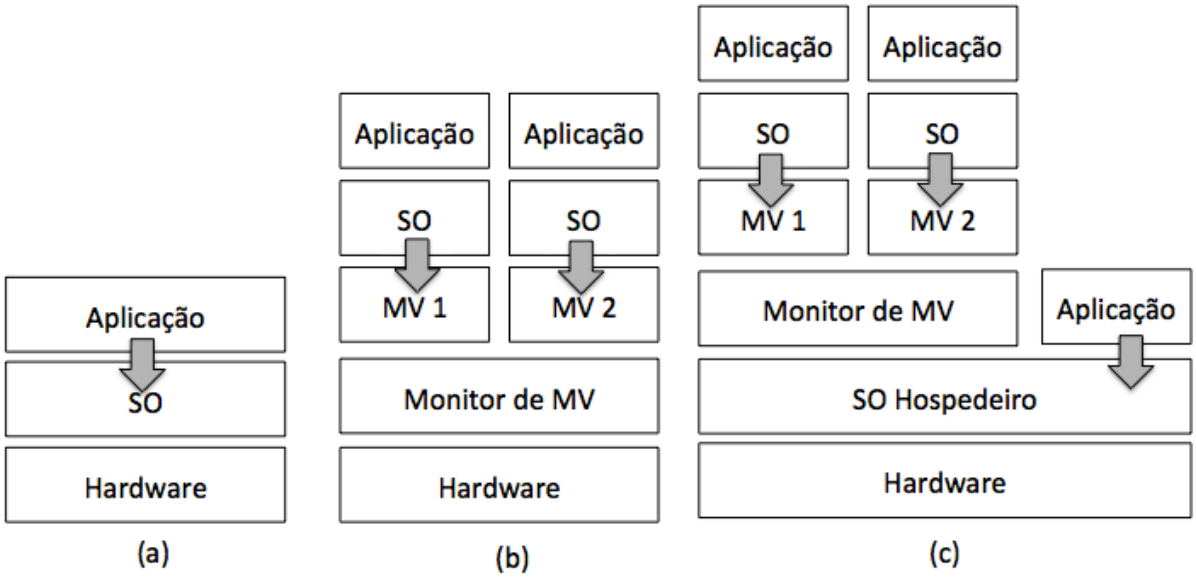
Conceitos

Maquina virtual (SILBERSCHATZ; GALVIN; GAGNE, 2013) (LAUREANO, 2006) (SMITH; NAIR, 2005) é uma abstração de hardware (CPU, memória, disco, dentre outros) que permite a criação de diferentes ambientes de execução. Cada Máquina Virtual é independente e fornece a ilusão de que cada ambiente está sendo executado independentemente. Um hardware físico permite executar diferentes sistemas operacionais concorrentemente, cada um em uma Máquina Virtual diferente. Cada sistema operacional gerencia a sua própria memória e escalonamento da sua Máquina Virtual.

O Monitor de Máquinas Virtuais (MMV) cria o ambiente execução do sistema operacional. Como técnica de virtualização, o monitor pode fornecer interfaces sem modificação (na virtualização completa) ou interfaces adaptadas (na para-virtualização) para o sistema convidado.

Os modelos de sistemas são apresentados na Figura 6. No sistema não virtualizado, a aplicação interage com o sistema operacional e este é responsável por gerenciar o hardware.

Figura 6 - Modelos de Sistemas. (a) Não virtualizado. (b) Virtualizado Tipo 1; (c) Virtualizado Tipo 2.



Fonte: Autoria própria.

No sistema do tipo 1, o sistema operacional é responsável por prover a virtualização e o monitor de Máquinas Virtuais está entre o hardware e as Máquinas Virtuais. Exemplos deste tipo de sistema são o Xen Server e o Vmware ESX. No sistema tipo 2, o Monitor de Máquinas Virtuais está entre a Máquina Virtual o sistema operacional hospedeiro. Neste tipo, o MMV é uma aplicação que está sendo executada no sistema operacional hospedeiro. Outras aplicações também podem ser executadas no sistema operacional hospedeiro. Exemplos do tipo 2 são KVM, Virtual Box e Parallels Desktop.

Vantagens e Utilização

Uma vantagem da virtualização é que um computador físico pode executar diferentes sistemas operacionais concorrentemente. Sendo assim, pode-se ter diversos tipos de servidores sendo executados simultaneamente em uma máquina física.

Outra característica é isolamento de falhas das Máquinas Virtuais. Uma Máquina Virtual é independente das demais, sendo assim, quando uma Máquina Virtual está danificada, as demais não serão afetadas. Como exemplo, se o sistema operacional de uma Máquina Virtual está sendo danificado com um vírus, devido ao isolamento, as demais Máquinas Virtuais praticamente não terão problemas com este vírus. Outro exemplo, se um servidor web precisa que o sistema operacional seja reiniciado, esta reinicialização não irá afetar as demais Máquinas Virtuais, que continuarão executando o seu sistema operacional normalmente.

Outra vantagem é a possibilidade de suspender (hibernar, *suspend*) a Máquina Virtual em execução e depois voltar a executá-la (*resume*), isto sem afetar as demais Máquinas Virtuais. Também é possível fazer cópia da Máquina Virtual em execução (*snapshot*) ou da imagem da Máquina Virtual (*clone*). A cópia da Máquina Virtual em execução (snapshot) permite que sejam criados backups do estado do sistema operacional que podem ser restaurados quando necessário. A cópia da imagem da Máquina Virtual permite que se crie um ambiente com programas instalados e configurados, e depois que esse ambiente possa ser replicado, sem a necessidade de instalar e configurar novamente os programas.

As Máquinas Virtuais também podem ser utilizadas na área da pesquisa. Utilizando as Máquinas Virtuais, pode-se modificar o sistema operacional da Máquina Virtual sem afetar a máquina física. Também permite aos usuários configurar programas sem afetar a configuração da máquina física.

A Computação em Nuvem tem utilizado a capacidade de abstração dos recursos da virtualização e o gerenciamento de Máquinas Virtuais para prover seus serviços, tanto de infraestrutura, quanto de plataforma e software.

2.2.2. Gerenciadores de Infraestrutura

Um problema da Nuvem está relacionado à Interoperabilidade. No nível de infraestrutura, cada provedor de Nuvem possui sua própria API para que possam ser desenvolvidas as aplicações em sua infraestrutura. No nível de plataforma, a interoperabilidade entre PaaS é dificultada pelos formatos de dados de cada provedor e em função das diferentes interfaces para o mesmo serviço. Uma aplicação realizada no Azure (AZURE, 2014) não é compatível com Google App Engine (GAE) (GOOGLE, 2014a) e vice-versa. É por essa razão que o cliente acaba ficando preso a um determinado provedor de plataforma de Nuvem.

Uma possível saída para evitar o “aprisionamento” da aplicação ao provedor de PaaS comercial é a utilização de uma padronização das interfaces dos serviços oferecidos pelos provedores de Plataforma e/ou uma plataforma aberta que forneça serviços de PaaS, utilizando as infraestruturas fornecidas pelos provedores de IaaS.

A integração desses provedores de IaaS para o fornecimento de uma plataforma como serviço permitirá a portabilidade das aplicações. Com a virtualização utilizada pelos provedores de infraestrutura, é possível criar um ambiente semelhante em diferentes provedores, sendo necessário fazer uma orquestração dos serviços. Existem várias tentativas de padronização de API no nível de IaaS, entretanto, os grandes provedores de IaaS não possuem o interesse na padronização. Para estes, se a aplicação e os dados ficam presos em seus serviços, o cliente também fica preso e subordinado às suas definições de preços, aumentado a sua receita. Os softwares abertos de infraestrutura tentam fornecer alternativas a este problema, tais como Eucalyptus (NURMI *et al.*, 2009) e OpenNebula (SOTOMAYOR *et al.*, 2009b).

O Eucalyptus (NURMI *et al.*, 2009) foi desenvolvido para ser um provedor de Infraestrutura aberto, com as funcionalidades semelhantes às da Amazon. É um projeto de pesquisa originado da Universidade da Califórnia, Santa Barbara. O Eucalyptus permite criar uma infraestrutura de Nuvem pública ou privada. Implementa as interfaces dos serviços EC2, S3 e EBS (*Elastic Block Storage*) (JAMES MURTY, 2008) fornecidos pela Amazon, com pouca exceções. Essa

característica permite que o cliente que desenvolveu um serviço utilizando os serviços da Amazon não fique preso à infraestrutura da Amazon, pois pode utilizar o Eucalyptus para criar sua Nuvem Privada.

OpenNebula (SOTOMAYOR *et al.*, 2009b) é um software aberto para a construção de IaaS. É desenvolvido e mantido pelo Grupo de Pesquisa de Arquitetura de Sistemas Distribuídos (*Distributed Systems Architecture Research Group*) da Universidade de Computação de Madri (*Complutense University of Madrid*). Para interoperabilidade entre as Nuvens implementa o padrão Interface de Computação de Nuvem Aberta (*Open Cloud Computing Interface* - OCCI) (OCCI-WG, 2009) e EC2-Query (JAMES MURTY, 2008). Além desses, possui dois *drivers* para acessar os seus serviços: DeltaCloud (DELTACLOUD, 2014) e LibCloud (LIBCLOUD, 2014).

OpenStack (OPENSTACK, 2014a) é um software de código aberto para a criação, configuração e gerenciamento de infraestrutura em nuvem. Foi desenvolvido pelo Rackspace em conjunto com a Nasa e utiliza a licença Apache.

2.2.3. Provedores de Nuvens Públicas

Nesta seção, apresentaremos uma comparação entre os sistemas provedores de Nuvem, visando os aspectos relacionados à migração da aplicação e dos dados entre as Nuvens. A Tabela 3 apresenta a comparação entre os provedores Amazon EC2, Rackspace, GoGrid e ElasticHosts. As informações foram colhidas do site dos provedores e com base na criação das instâncias nestes provedores.

O primeiro aspecto analisado foi a Ferramenta de Desenvolvimento. Examinou-se qual suporte/interface é fornecido aos programadores para que possam interagir com o provedor. Amazon possui 18 serviços disponíveis para o usuário (Amazon EC2, S3, Simple Reduce, dentre outros). Para cada serviço, existe uma API de programação. A API de programação pode ser baseada no protocolo REST ou SOAP, ou possuir ambos os protocolos. Possui Amazon Web Services SDK (AWS SDK), onde os clientes podem desenvolver suas aplicações utilizando as

bibliotecas do AWS em linguagens específicas, tais como Java, Python, PHP, Ruby e .NET. Essas bibliotecas implementam as API da Amazon utilizando SOAP. Não são todos os serviços que possuem as bibliotecas disponíveis. Assim, é necessário programar utilizando os protocolos SOAP ou REST diretamente. Além das bibliotecas, o serviço EC2 da Amazon fornece uma ferramenta de Linha de Comando para criar, gerenciar e apagar as instâncias. Esta ferramenta implementa o protocolo REST.

Tabela 3 - Comparação entre IaaS.

Característica	Amazon Ec2	Rackspace	GoGrid	ElasticHosts
Ferramenta de desenvolvimento	- Linha de Comando (REST) - AWS SDK (SOAP) - 18 APIs	- Cloud Servers e File API (REST) - Console Virtual (REST)	GoGrid API (REST)	Elastic Host Cloud Servers API (REST)
IP Público	IP Elástico (Pago)	Dinâmico	10 disponíveis	Dinâmico
Armazenamento de Dados (Storage)	Amazon S3	Cloud Files	Cloud Storage	Driver Virtual (Com SO)
Principal Acesso	ssh (sem senha) + Certificação	ssh (envia senha por e-mail)	ssh (Cria senha)	VNC (Habilitar ssh)
Permite Servidor Dedicado	Sim	Não	Sim	Não
Imagem	AMI	Pré-definidas	MyGSI	Pré-definidas
Balanceamento de Cargas	Elastic Load Balancing	Não	Load Balancing	(*)
Virtualização	Xen	VMware	Xen e Vmware	KVM
Localização do Data Center	EUA (N. Virginia, N. Califórnia) / UE – Irlanda / PA – Singapura	EUA	EUA-Leste/ Oeste	UK (Londres) e EUA (Santo Antonio)

(*) Informação não encontrada.

Fonte: Autoria própria.

Rackspace provê dois serviços: Servidores em Nuvem (*Cloud Servers*) e Arquivos em Nuvem (*Cloud Files*), para computação e armazenamento,

respectivamente. Ambos os serviços possuem API própria e utilizam o protocolo REST para a comunicação do cliente com o servidor. Também possui um Console Virtual baseado em REST. A Gogrid e a ElasticHosts possuem suas próprias APIs para os seus serviços. Ambos também utilizam o protocolo REST.

Quanto ao IP Público, a Amazon fornece o serviço pago chamado IP Elástico (*Elastic* IP). O IP Elástico é um IP público fornecido ao cliente, mas não está atrelado a uma única instância. Dessa forma, o cliente pode vincular e desvincular o IP público a uma instância a qualquer momento. O Rackspace e ElasticHosts fornecem por padrão um IP público para cada instância de Máquina Virtual criada. Quando a instância é destruída, o IP também é destruído. O GoGrid fornece por padrão 10 IPs públicos disponíveis para serem utilizados pelas instâncias. Caso o usuário necessite de mais, é necessário comprar mais IPs. Todos os provedores fornecem IPs privados gratuitamente.

Para o Armazenamento de Dados (*storage*), a Amazon fornece o Amazon Serviço de Armazenamento Simples (*Simple Storage Service*), mais conhecido com Amazon S3. O Rackspace fornece o serviço de Arquivos em Nuvem (*Cloud Files*) e o GoGrid o de Armazenamento em Nuvem (*Cloud Storage*). O serviço de armazenamento fornecido pelo ElasticHosts é baseado em Driver Virtual. Ou seja, solicita-se um HD virtual e o cliente é responsável por gerenciá-lo.

O "principal acesso" está relacionado ao principal mecanismo para acessar a instância de Máquina Virtual. No Amazon EC2, por padrão, a maioria das imagens utiliza Secure Shell (ssh) sem senha. No entanto, faz-se necessária a utilização do Certificado Digital para acessar as instâncias. O Rackspace utiliza o ssh como principal meio de acesso à sua Máquina Virtual. A senha da instância da Máquina Virtual é enviada para o e-mail cadastrado e também pode ser obtida na página de navegação. Esta característica dificulta a automatização de criação e gerenciamento da instância. Entretanto, é importante verificar se a API da Rackspace fornece algum mecanismo para contornar este problema. No GoGrid, a instância é criada e pode ser acessada por padrão utilizando o ssh sem senha. Assim que se faz o primeiro acesso, o usuário troca a senha. O ElasticHosts não utiliza o ssh com acesso

padrão, mas sim a Computação de Rede Virtual (*Virtual Network Computing* - VNC). Depois de acessar a instância utilizando o VNC é possível habilitar o ssh.

O aspecto “Permite Servidor Dedicado” está relacionado à execução de uma única instância no servidor. Atualmente, a Amazon fornece serviço de HPC na qual cada instância possui a capacidade de computação dedicada (AMAZON, 2014b). O GoGrid também permite que a instância seja executada em um servidor dedicado. O Rackspace e ElasticHosts não possuem o serviço de servidor dedicado para a Nuvem.

Cada provedor possui o seu formato de imagem. A Amazon possui a Imagem de Máquina da Amazon (*Amazon Machine Image* - AMI). O usuário pode utilizar as AMIs disponibilizadas pela Amazon, criar sua própria AMI, pagar pela imagem vendida por outros usuários ou utilizar as AMI disponibilizadas por outros usuários/comunidades. O Rackspace possui as imagens pré-definidas, ou seja, apresenta apenas as opções que o usuário pode escolher, disponibilizadas pelo provedor. GoGrid possui a imagem no formato GSI (GoGrid Server Image). No GSI, as aplicações são pré-configuradas e customizadas, assim como o Sistema Operacional. O usuário pode utilizar a imagem pré-definida pelo GoGrid, ou customizar a sua imagem. Existem empresas parceiras que fornecem imagens pagas. ElasticHosts só permite a utilização de imagens pré-definidas.

O Balanceamento de Cargas é a distribuição automática do tráfego (ou dos trabalhos a serem realizados) entre os recursos computacionais (ou instâncias) de acordo com uma política definida. Amazon fornece o serviço de balanceamento, denominado Elastic Load Balancing, baseado no tráfego da aplicação, sendo possível definir políticas de escalonamento das requisições utilizando a API. Rackspace não fornece o serviço de balanceamento de cargas, entretanto disponibiliza um programa que pode ser utilizado para o balanceamento de cargas. GoGrid fornece o serviço de balanceamento de Cargas. Não foram encontradas informações do ElasticHosts sobre disponibilidade de serviço ou programa para balanceamento de Cargas.

Quanto à Virtualização, Amazon utiliza o Xen Modificado como monitor de Máquinas Virtuais. O Rackspace utiliza o Xen como tecnologia de virtualização. GoGrid utiliza Xen e VMware. ElasticHosts optou pelo KVM.

Outro aspecto importante está relacionado à localização do provedor. Esta característica define as leis às quais sua aplicação e seus dados estarão sujeitos. Como exemplo, nos EUA existem restrições quanto ao uso de criptografia. Amazon possui servidores localizados nos Estados Unidos da América (EUA). As duas localizações disponíveis nos EUA são no Norte da Carolina e no Norte da Califórnia. Também disponibiliza os serviços na Irlanda (União Européia - UE) e na Singapura (Pacífico da Ásia). Rackspace e Gogrid são localizados somente nos EUA. ElasticHosts fornece o serviço no Reino Unido (Londres) e nos EUA (Santo Antonio).

2.2.4. Bibliotecas e Padrões de Interoperabilidade

A Interface de Computação de Nuvem Aberta (*Open Cloud Computing Interface* - OCCI) (OCCI-WG, 2009) é padronização de interface da Nuvem para IaaS promovida pelo OpenGrid Fórum. O OpenGrid Fórum é uma comunidade de usuários, desenvolvedores e vendedores liderando os esforços de padronização global para Computação em Nuvem. O objetivo do OpenGrid Fórum com o OCCI é que os vendedores/desenvolvedores de Grades e Nuvens possam oferecer interfaces padrões para a entrega de serviços escaláveis dinamicamente dentro de seus produtos. O OCCI é baseado em diversas APIs que fazem parte de grandes corporações de provedores de Nuvens Públicas, sendo elas: Amazon EC2 API, ElasticHosts API, FlexiScale API, GoGrid API, Sun Cloud API, dentre outras. O OpenNebula e o OpenStack implementam esta interface.

DeltaCloud (DELTACLOUD, 2014) é um projeto da RedHat que define a API de Web Services para interagir com provedores de serviços de Nuvem de maneira uniforme. No OpenNebula, sua API é implementada como um *driver* utilizando as interfaces do OCCI para interação com a infraestrutura.

LibCloud (LIBCLOUD, 2014) é uma biblioteca de cliente em Python para interagir com diversos provedores de Nuvens, como Amazon EC2, Rackspace, GoGrid, dentre outros. No OpenNebula, sua API é implementada como um *driver* utilizando as interfaces do OCCI para interação com a infraestrutura.

Esses exemplos demonstram uma tentativa de padronização das interfaces dos provedores da Infraestrutura. Os grandes provedores de Nuvens Públicas (seja em IaaS ou PaaS) não possuem interesse na padronização, enquanto outros órgãos acadêmicos e comerciais tentam estabelecer inúmeras padronizações.

2.3. AVALIAÇÃO DE DESEMPENHO

A avaliação de desempenho (JAIN, 1991)(GREGG, 2013)(JOHNSON; COUTINHO, 2011) consiste no estudo do comportamento de um sistema avaliando a eficiência ou rendimento para obter ou prover desempenho com baixo custo.

A avaliação de desempenho é considerada uma arte (JAIN, 1991) e também é subjetiva, pois o significado de bom ou ruim depende do contexto no qual está sendo aplicado (GREGG, 2013), cabendo ao avaliador entender o contexto e saber interpretar a avaliação corretamente.

O primeiro passo para analisar o desempenho é definir claramente qual é o **objetivo** da análise (JAIN, 1991)(GREGG, 2013). Este objetivo ajudará a definir as métricas, carga de trabalho, as técnicas de avaliação, dentre outros. O objetivo fornece um direcionamento nas tomadas de decisões.

As **métricas** são os critérios utilizados para medir o desempenho de um determinado sistema. Como exemplos de métricas, têm-se utilização, saturação, vazão (*throughput*), tempo de resposta. Os **parâmetros** são as características que serão variadas no estudo do desempenho. Na análise do tráfego da rede de computadores, o parâmetro pode ser o tamanho do pacote, na análise de CPU o parâmetro pode ser o tamanho de requisição. O parâmetro deve estar relacionado com as métricas, e ambos devem estar relacionados com a técnica de avaliação selecionada e com o objetivo da análise de desempenho.

A **Carga de Trabalho** (*workload*) (JAIN, 1991) são as requisições realizadas no sistema, como por exemplo números de instruções que a CPU precisa executar. Em computação, **Benchmarking** (VEDAM; VEMULAPATI, 2012) é um conjunto de programas usados para estressar características especificas do sistema com a variação da carga de trabalho e com os resultados para comparar o desempenho específico de aplicações ou de recursos computacionais.

2.3.1. Técnicas de Avaliação de Desempenho

Na avaliação de desempenho, existem basicamente três técnicas: (JAIN, 1991)(GREGG, 2013)(JOHNSON; COUTINHO, 2011): (a) modelagem; (b) simulação; e (c) experimentação.

(a) Modelagem: O uso de modelo permite a abstração de detalhes que não são relevantes para a caracterização do desempenho em alto nível. Nesta técnica identificam-se o componentes principais do sistemas e sua interação. Pode ser utilizada para predição de performance em um sistema a ser desenvolvido ou em um sistema existente. Esta técnica que possui o menor custo financeiro. A modelagem pode ser baseada em resultados da simulação ou experimentação. Como exemplos de técnicas de modelagem têm-se teoria da filas, redes de petri, processos estocásticos, dentre outros.

(b) simulação: é muito utilizada para avaliação de sistemas computacionais, em decorrência da facilidade do aprendizado e simplicidade de utilização (JOHNSON; COUTINHO, 2011). De maneira semelhante à técnica de modelagem, pode ser utilizada para avaliar um sistema a ser desenvolvido ou em um sistema existente a afim de prever o desempenho. Permite a criação e avaliação de testes experimentais. Esta técnica permite simular uma carga de trabalho em um sistema que não está em produção ou testar cargas de trabalho maiores do que a do sistema de produção. Existem os simuladores para projetar sistemas, podendo ser genéricos, como por exemplo, Arena (ROCKWELL, 2014), ou específicos, como os simuladores de rede OMnet++ (VARGA, 2001) e o Network Simulator (ns-2) (NS-2, 2014), os simuladores de computação em grade GridSim (BUYYA, RAJKUMAR;

MURSHED, 2002) e SimGrid (CASANOVA, 2001) e o simulador de computação em nuvem CloudSim (CALHEIROS *et al.*, 2011).

(c) experimentação ou aferição: é uma técnica que analisa um sistema já existente aplicando uma carga de trabalho no sistema que está em produção permitindo que diversos parâmetros sejam aferidos. As medidas podem ser obtidas pela própria aplicação, através dos métodos de captura da aplicação e seus *logs*, ou monitoradas por outras ferramentas no momento em que a aplicação estiver sendo executada. Para analisar um sistema existente, deve-se começar pela experimentação através da caracterização da carga de trabalho e do desempenho resultante (GREGG, 2013). Como exemplo de ferramentas no Linux (GREGG, 2013), têm-se *vmstat* para estatística de memória, *mpstat* para uso de CPU, *iostat* para reportar o uso de E/S, *strace* para analisar chamadas de sistemas e *sytemtap* que é uma ferramenta de *tracing* estático e dinâmico que utiliza dados aferidos pelo *kernel*.

2.3.2. Perspectivas de Desempenho

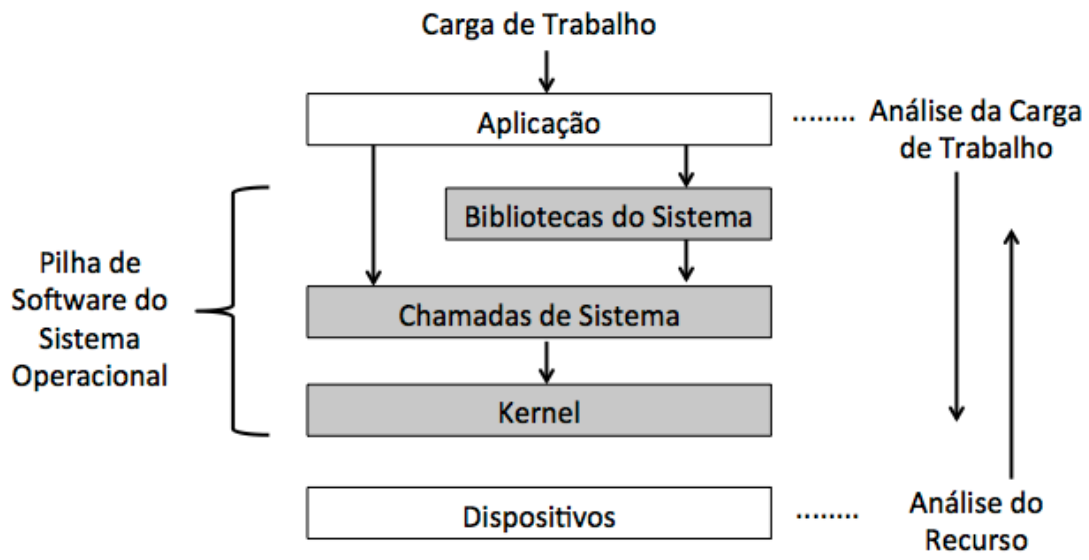
Análise de desempenho de sistemas computacionais pode ser dividida em duas perspectivas (GREGG, 2013): (a) análise de recursos; e (b) análise da carga de trabalho, conforme a Figura 7. A análise da carga de trabalho envolve o comportamento da aplicação dada uma carga. Na análise de Recursos, investiga-se o comportamento dos recursos. Em ambas as perspectivas, a pilha de software do sistema operacional pode ser utilizada para obter os resultados de desempenho.

(a) Análise de Recursos

Nesta perspectiva, são analisados os recursos, tais como CPU, memória, disco, interfaces de redes, dentre outros. Os administradores de sistema têm a perspectiva da análise de recursos, já que são responsáveis pelos recursos. Nesta perspectiva o foco está na utilização dos recursos para identificar se este está chegando no limite ou quando chegará. Sendo assim, pode-se investigar problemas de performance, para ver qual recurso é responsável pelo problema, ou pode-se planejar a capacidade, obtendo informações para novos sistemas ou para sistemas

existentes. A utilização de benchmark pode auxiliar no planejamento de capacidade. As métricas mais utilizadas na análise de recursos são: utilização, Operação de Entrada e Saída por segundo (*Input/Output Operations Per Second* - IOPS), saturação e vazão (*throughput*).

Figura 7 - Perspectivas de Análise de Desempenho.



Fonte: Traduzido de (GREGG, 2013).

(b) Análise da Carga de Trabalho

A análise da carga de trabalho investiga a performance da aplicação através da utilização de uma carga de trabalho e verificação do comportamento da aplicação (GREGG, 2013). Os desenvolvedores de aplicação estão voltados para esta perspectiva.

É necessário entender as características da aplicação antes de inicializar os testes de desempenho da aplicação. Através da verificação das características da aplicação pode-se desenvolver um modelo de carga de trabalho para que se possa utilizar repetidamente (JAIN, 1991). Este processo é denominado de caracterização da carga de trabalho.

Algumas verificações que podem ser realizadas são (GREGG, 2013): (a) qual a função da aplicação (*e.g.* servidor web, banco de dados, aplicação científica); (b) quais operações ou requisições a aplicação realiza (*e.g.* servidor web realiza requisições HTTP); (c) como a aplicação está configurada e por quê - verificar os arquivos de configuração e se possui parâmetros de melhoria de desempenho. Também é necessário definir os objetivos e as métricas para análise. As métricas mais utilizadas nesta perspectiva são latência, vazão e utilização dos recursos.

2.3.3. Operações de performance de E/S

Existem operações de Entrada/Saída (E/S) podem influenciar diretamente na performance do sistema (GREGG, 2013). Neste tópico, as principais características para análise de performance de E/S da aplicação são analisadas, sendo elas: (a) Operação de Escrita Assíncrona x Síncrona; (b) Buffered x Direct I/O e (c) Acesso Sequencial x Acesso Randômico.

(a) Operação de Escrita Assíncrona x Síncrona

A operação de Escrita Síncrona (O_SYNC) só é finalizada quando estiver realmente escrito no local de armazenamento persistente (*e.g.* disco). Na operação de escrita assíncrona (*write-back caching*), os dados são transferidos para a memória principal e escritos em outro momento em um local persistente.

(b) Buffered x Direct I/O

Em operações de E/S do tipo *Buffered*, os dados são *bufferizados* na cache do sistema de arquivos e precisam ser processados pela CPU.

O DIRECT I/O é utilizado em aplicações que possuem seu próprio sistema de gerenciamento de cache, como banco de dados e *middlewares* (MPI). A vantagem é que os arquivos não competem pelo *buffer* (ou cache) do sistema de arquivos e também não são processadas pela CPU.

A utilização do DIRECT I/O permite que se use menos CPU e menos cache, permitindo uma melhor performance (considerando-se que a aplicação esteja gerenciando corretamente as operações de escrita e leitura de arquivos).

(c) Acesso Sequencial x Acesso Randômico

Operação sequencial acessa o dispositivo de armazenamento (*e.g.* disco) de maneira contínua. Melhor desempenho em transferência de dados com tamanho grande (bloco de 128KB).

Na operação Randômica, o acesso ao dispositivo de armazenamento não ocorre de maneira contínua, possuindo o melhor desempenho para transferência de dados com tamanho pequeno (*e.g.* bloco com tamanho de 4KB).

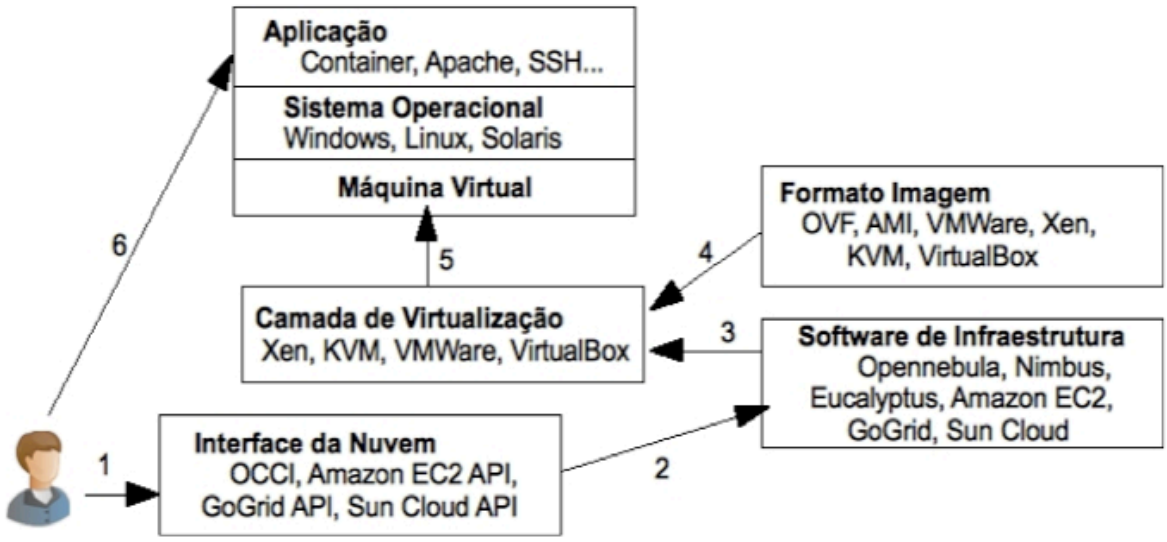
2.4. PROBLEMAS

Atualmente, os serviços de plataformas são oferecidos para a utilização de uma Nuvem específica e com limitação de linguagem. A utilização dessas plataformas causa o aprisionamento e o cliente fica vulnerável às políticas da Nuvem, uma vez que esse aprisionamento ocorre em decorrência da dificuldade de transição das aplicações de maneira transparente e da falta de interoperabilidade entre as Nuvens. Como exemplo, o App Engine (Google) (GOOGLE, 2014a) e o Azure (Microsoft) (AZURE, 2014) oferecem uma plataforma de desenvolvimento de aplicações proprietária, específica para suas infraestruturas. Aneka (VECCHIOLA; CHU; BUYYA, 2009)(VECCHIOLA; PANDEY; BUYYA, 2009)(BUYYA, R. *et al.*, 2009)(CHU *et al.*, 2007) é um software de plataforma para o desenvolvimento de aplicações em Nuvem, restrito a aplicações .NET e proprietária.

Uma forma de contornar este problema é a criação de uma plataforma aberta, que interligue as diferentes Nuvens e permita a migração da aplicação do usuário. Por conta dos serviços de plataforma (PaaS) disponíveis serem fechados, essa interligação pode ser feita no nível de infraestrutura (IaaS).

A criação de uma Máquina Virtual (Virtual Machine - VM) no nível de Infraestrutura da Nuvem é apresentada na Figura 8. O usuário acessa e gerencia os serviços disponibilizados pela Infraestrutura através de uma Interface de Nuvem (**1**). A Interface de Computação de Nuvem Aberta (*Open Cloud Computing Interface* - OCCI) é uma proposta de padronização de interface de programação para a Nuvem, sendo baseado em diversas interfaces de IaaS. A Interface da Nuvem comunica-se com o Software de Infraestrutura (**2**), sendo este responsável por criar, configurar e gerenciar a infraestrutura da Nuvem (**3**) (SOTOMAYOR *et al.*, 2009b)(KEAHEY *et al.*, 2009) podendo conter um ou mais Monitores de Máquinas Virtuais (MMVs), tais como Xen, KVM, Vmware, VirtualBox (SOTOMAYOR *et al.*, 2009b)(KEAHEY *et al.*, 2009). As Nuvens públicas possuem seu próprio software de Infraestrutura, mas existem alternativas abertas, como OpenNebula, Nimbus e Eucalyptus (SOTOMAYOR *et al.*, 2009b)(KEAHEY *et al.*, 2009). A Camada de Virtualização (CV) é composta pelo Monitor de Máquinas Virtuais (MMV), e este, é responsável por criar, suspender, reiniciar e finalizar as MVs a partir de uma imagem. Ao receber um pedido do Software de Infraestrutura (**3**), a Camada de Virtualização cria uma MV (**5**), utilizando uma determinada imagem (**4**).

Figura 8 - Criação de Máquina Virtual utilizando IaaS.



Fonte: Autoria própria.

A imagem é um arquivo que contém informações sobre a Máquina Virtual, tais como Sistema Operacional e Aplicações instaladas. A partir da imagem, o Monitor de Máquinas Virtuais (MMV) pode criar uma ou mais MVs. Cada MMV utiliza seu próprio formato de imagem. Formato de Virtualização Aberto (*Open Virtualization Format* - OVF) (DMTF, 2010) é uma especificação para uma padronização das imagens. Após a criação da MV, o usuário pode acessar **(6)** as aplicações e gerenciar o Sistema Operacional da Máquina Virtual.

3. A METODOLOGIA DE CARACTERIZAÇÃO CSE

Os provedores de Computação em Nuvem oferecem diversos tipos de serviços, sendo um deles o fornecimento de instâncias de Máquinas Virtuais. Este serviço é caracterizado como Infraestrutura como Serviço (Infrastructure as a Service – IaaS).

Diferentes tipos de instâncias são fornecidos pelos provedores, variando a quantidade de processadores, memórias, tipo de disco, dentre outros. Esta variedade de tipos dificulta a comparação entre os provedores, por não existir padronização dos tipos de instâncias.

Como exemplo de configuração de discos, estes podem ser fornecidos como disco local ou remoto, com diferentes tecnologias de disco (HDD, SSD), com garantia de Operação de Entrada e Saída por segundo (IOPS), além de possuírem diferentes tipos de interfaces (SCSI, iSCSI, SATA).

Instituições privadas e públicas também estão percebendo o benefício de utilização de Máquinas Virtuais. Como exemplo, a Universidade de São Paulo lançou a Nuvem USP, a fim de prover serviços de Computação em Nuvem e atender a demanda da comunidade universitária por recursos computacionais, fornecendo instâncias de Máquinas Virtuais para grupos de pesquisa. Essas instâncias também possuem diferentes tipos de características. A princípio, o uso é gratuito, apesar de haver unidade monetária de cobrança por uso das instâncias, variando os valores de acordo com o tipo de instância. Entretanto, pode haver uma grande demanda, e os recursos podem não ser suficientes para atende-la, sendo necessária a utilização de “cotas”. Ou seja, cada grupo de pesquisa teria um valor (em real) para utilizar os recursos da Nuvem USP.

As aplicações também evoluíram e exigem cada vez mais recursos. Se de um lado temos uma oferta de diferentes tipos de instâncias, de outro temos aplicações mais complexas e que exigem mais recursos computacionais. As aplicações podem ser categorizadas pela utilização de recursos, como:

- CPU bound: é a aplicação que utiliza bastante computação, requisitando o processador constantemente para realizar processamento.
- I/O bound: é a aplicação que utiliza bastante I/O, requisitando os recursos periféricos, seja de armazenamento ou de rede, com operações de Entrada e saída.

As aplicações podem ser totalmente CPU ou I/O bound, mas também podem ser identificadas partes da aplicação que possuem essas características. Como exemplo, no início da execução de uma determinada aplicação pode haver uma grande quantidade de operações de E/S de leitura de arquivos grandes. Em seguida, durante sua execução, esta aplicação pode executar muitos cálculos e utilizar bastante processamento. Esta fase pode ser classificada com CPU bound. Por fim, na fase final, esta aplicação pode realizar muitas operações de escrita com arquivos grandes. Esta última fase também pode ser classificada como I/O bound.

A identificação de partes da aplicação que utilizam determinados recursos em demasia permite identificar o comportamento da aplicação.

Para analisar a aplicação, é necessário que um especialista acompanhe, seja ele o desenvolvedor ou um usuário avançado, de maneira que as peculiaridades desta possam ser analisadas. Como exemplo, o aumento da carga de trabalho da aplicação; para saber como aumentar a carga, é necessário entendê-la primeiramente, seja pelo estudo do código fonte, seja por auxilio de um usuário mais avançado.

É necessário analisar e categorizar o comportamento e utilização dos recursos da aplicação, assim como analisar o comportamento e o desempenho das instâncias dos diversos provedores de Nuvem, a fim de melhorar o desempenho da aplicação, possibilitando, por exemplo, a diminuição do tempo de execução da mesma.

Diante desta grande variedade de tipos de instâncias, provedores e preços, o usuário fica indeciso quanto a onde executar a sua aplicação. Qual provedor utilizar? Qual tipo de instância e de configuração fornece o melhor desempenho para a aplicação executar em menos tempo? Qual o menor custo? Qual o melhor custo

por desempenho? O principal problema levantado neste trabalho é: Que o tipo de instância e configuração é mais adequado para uma determinada aplicação, levando-se em consideração o menor tempo de execução ou menor custo?

Todos os questionamentos levantados implicam a necessidade de procedimentos para a escolha da instância. Sendo assim, o objetivo deste trabalho é determinar os procedimentos necessários para selecionar em qual tipo de instância a aplicação será executada, levando-se em consideração custo ou desempenho. Ou seja, desenvolver uma metodologia, composta de um conjunto de procedimentos para determinar o tipo de instância mais adequado para uma aplicação específica, com o intuito de aprimorar o desempenho da aplicação. A metodologia tem o intuito de fornecer os procedimentos para auxiliar na investigação e na tomada de decisões.

A hipótese adotada neste trabalho é de que o recurso computacional preponderante da aplicação pode definir qual o tipo de instância é mais adequada para uma determinada aplicação sem a necessidade de executar a aplicação em todas as instâncias. Sendo assim, se o recurso computacional dominante permite determinar qual o tipo de instância é mais adequado, então as aplicações poderão ser executadas em menos tempo ou podem obter o menor custo financeiro.

A hipótese adotada permite que as instâncias mais adequadas para a aplicação sejam selecionadas e que o tempo de execução da aplicação possa ser diminuído. Esta hipótese avança o conhecimento científico à medida que o detalhamento da aplicação e dos recursos são afunilados e novas características podem ser incluídas nos procedimentos de caracterização, não ficando restrita às tecnologias existentes.

Neste trabalho, portanto, pretende--se demonstrar os procedimentos necessários para selecionar qual tipo de instância é melhor para uma determinada aplicação. Sendo assim, se faz necessário criar uma metodologia que permita analisar detalhadamente, as características da aplicação e também da instância.

Em seguida, é necessário definir critérios de seleção. Esses critérios podem ser incorporados à metodologia, de acordo com as necessidades do usuário.

Inicialmente, utiliza-se o critério de desempenho e de custo. O intuito é construir uma forma de seleção, de escalonamento de Máquinas Virtuais para a execuções de Aplicações para Computação em Nuvem.

As variáveis dependentes, ou seja, aquelas que apresentam os resultados dos experimentos e que são medidas, estão relacionadas ao custo e ao desempenho. As variáveis independentes, aquelas introduzidas intencionalmente para serem manipuladas nos experimentos, estão relacionadas ao tipo de instância, às aplicações específicas e aos recursos computacionais.

O objetivo inicial é definir qual tipo de instância é melhor para uma determinada aplicação. Para isso, verificou-se a necessidade de analisar com mais detalhes as características que os tipos de instância podem fornecer. Diversos recursos computacionais podem ser analisados, tais como disco, rede, memória, CPU. Além disso, diversas métricas também podem ser utilizadas para essas análises: vazão, utilização, saturação.

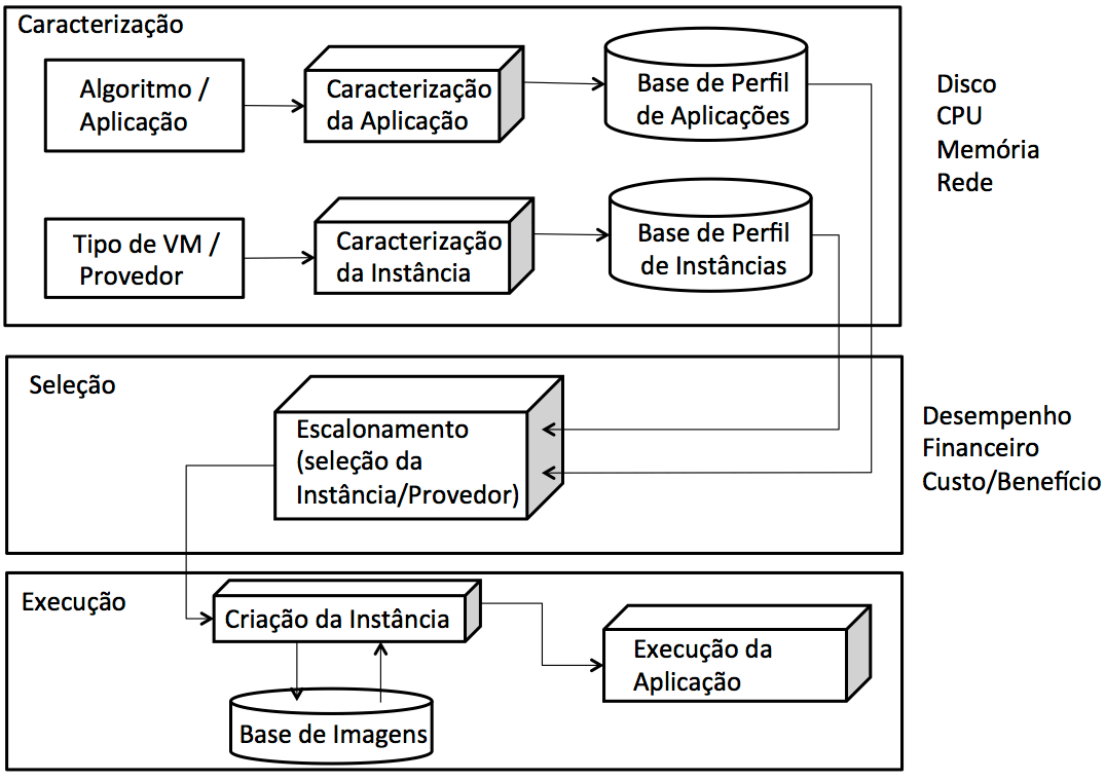
Desse modo, a metodologia deve apresentar os procedimentos necessários para que possam ser feitas as caracterizações, levando-se em consideração um ou mais recursos computacionais e permitindo-se que novos recursos possam ser adicionados de acordo com os requisitos da aplicação.

As métricas também variam de acordo com os recursos utilizados. Sendo assim, escolhem-se as métricas de acordo com os recursos. Notou-se então, que a aplicação tem uma grande influência na escolha do tipo de instância, pois a aplicação pode não estar utilizando todos os recursos que desejaríamos analisar. Como exemplo, uma aplicação que seja do tipo I/O bound de disco e utilize pouca memória, não se faz tão necessário analisar a memória, pois o desempenho é definido pelo I/O de disco. Portanto, se há muitas aplicações desse tipo, é mais interessante é analisar o I/O.

CSE (Categorização, Seleção e Execução) é uma metodologia que fornece as diretrizes para selecionar em qual provedor e instância a aplicação será executada, permitindo criar uma base de dados com o perfil da aplicação e os tipos de Máquinas Virtuais.

A Figura 9 apresenta o diagrama da metodologia, sendo composta por três fases: caracterização, seleção e execução. Cada uma das etapas será descrita em detalhes nas próximas seções.

Figura 9 - Metodologia CSE.



Fonte: Autoria própria.

3.1. Caracterização

Antes de fazer a seleção de instância em Computação em Nuvem, é necessário compreender não somente o comportamento das MVs, mas também o comportamento das aplicações.

Essa etapa fornece os procedimentos para caracterizar a aplicação e o tipo de instância. A caracterização da aplicação apresenta como é o comportamento da aplicação em relação aos recursos. A caracterização do tipo de instância indica qual a melhor utilidade, ou tipo de operação, é mais indicada para a instância. Por

exemplo, uma instância pode ser mais adequada para leitura de arquivos do que as demais analisadas.

A caracterização das aplicações permite que elas possam ser alocadas para as instâncias mais adequadas, que também foram previamente caracterizadas. A determinação dessa alocação é realizada posteriormente, na fase de Seleção.

Uma aplicação do tipo CPU *bound* também pode ser analisada quanto a outro aspecto que não seja a CPU. Como exemplo, pode-se reduzir o tempo de execução da aplicação através da análise de I/O de disco, caso a aplicação tenha partes significativas de operações de I/O. Ou seja, é possível analisar um recurso secundário mais utilizado pela aplicação para obter, por exemplo, um menor tempo de execução.

Esta etapa demonstra como a caracterização pode ajudar na etapa posterior de seleção. O objetivo da caracterização é analisar o comportamento dos recursos computacionais de um tipo de instâncias e entender o funcionamento da aplicação em relação à utilização dos recursos computacionais.

A partir da determinação das características da aplicação e o tipo de instância é possível criar uma base de dados que possa ser utilizada posteriormente. Essa base de dados pode ser feita por diferentes usuários e permite que a adição de novas análises para que esta possa ser ampliada. As caracterizações da aplicação e da MV podem ser realizadas de forma paralela.

As variáveis independentes são as variáveis que podem ser manipuladas. Essas variáveis podem estar relacionadas aos recursos computacionais, tais como CPU, I/O de rede ou de disco. As variáveis dependentes são as variáveis dos resultados. Nessa etapa, essas variáveis dependem das métricas selecionadas para cada recurso computacional.

Foram definidas 2 etapas na fase de categorização, sendo elas: (1) caracterização da aplicação; e (2) caracterização da instância. As execuções destas etapas visam criar os respectivos perfis.

Na etapa de caracterização da aplicação, analisa-se a aplicação afim de verificar o recurso preponderante. Na etapa de caracterização da instância, analisa-se a instância com o intuito de verificar o seu comportamento em relação aos recursos. Em ambas as etapas, para cada tipo de recurso, é preciso aplicar procedimentos específicos de análise e selecionar as métricas adequadas.

Nesta fase, deve-se aumentar a carga da entrada da aplicação, com a finalidade de entender o seu comportamento com relação ao recurso selecionado. Como exemplo, quantidade e tamanho de arquivos de entrada podem ser incrementados em uma aplicação cientifica, a fim de caracterizar o comportamento da aplicação com o aumento das entradas. Nessa etapa, é de suma importância um especialista da aplicação, para que seja possível definir como estressar a aplicação.

3.1.1. Caracterização da Aplicação

A caracterização da aplicação tem como finalidade entender o comportamento e o funcionamento da aplicação, bem como a relação dela com os recursos computacionais.

A caracterização da aplicação é realizada monitorando os recursos computacionais, tais como CPU, disco, dentre outros. Nesta etapa, são definidos um ou mais recursos a serem analisados, e também as ferramentas e os procedimentos de monitoramento relacionadas a estes recursos.

Um problema na análise de aplicações é que nem sempre se tem acesso ao código fonte para entender o seu funcionamento. Sendo assim, é necessário que tenhamos procedimentos de caracterização da aplicação de maneira mais genérica e que não precisem de acesso ao código fonte.

O objetivo da caracterização de aplicação é determinar os procedimentos necessários para que se possa fazer uma análise da aplicação sem a necessidade de acesso direto ao código fonte.

Objetiva-se criar um perfil da aplicação e entender como ela interage com um determinado recurso computacional. A caracterização permite que tenhamos um entendimento do funcionamento da aplicação e do seu comportamento à medida que aumentamos a carga de entrada.

A investigação da aplicação através da análise de seu comportamento e funcionamento em relação ao recurso permite criar um perfil da aplicação, sem que seja necessário analisar o código fonte. Caracterizar a aplicação sem ter o acesso ao código fonte é entender seu comportamento e funcionamento, permitindo que aplicações do tipo caixa preta (que não possuem seu código liberado) possam ser caracterizadas. Se tivermos um perfil da aplicação podemos então selecionar uma instância mais específica, que aprimore seu desempenho, na etapa de seleção.

As variáveis dependentes, aquelas que são os resultados, estão relacionadas com as métricas. Estas métricas podem ser vazão, latência, dentre outras. As variáveis independentes estão relacionadas com o recurso escolhido e com o comportamento da aplicação. Elas estão relacionadas com a carga de entrada e os parâmetros da aplicação, permitindo que sua variação possa estressá-la para que possamos entender seu comportamento à medida que variemos esta carga.

Esta etapa pode ser feita sem análise do código fonte. Isso permite que determinemos o funcionamento da aplicação sem a necessidade de entender o código. Entretanto, é necessário haver um especialista da aplicação ou um estudo prévio, para que possamos variar a carga de entrada e os seus parâmetros de maneira efetiva.

O especialista é responsável por determinar previamente o que é necessário para a variação da carga de entrada e dos parâmetros efetivamente. É importante notar que aplicações cientificas possuem certos parâmetros que são específicos e não faz muito sentido, do ponto de vista do pesquisador, aumentar ou diminuir demasiadamente o seu valor. Isso aumenta a importância de se ter um especialista da aplicação, para que este possa auxiliar no entendimento dos parâmetros e nas cargas de entrada. Dessa forma, o especialista ao viabilizar o entendimento dos parâmetros e a variação da carga de entrada da aplicação, irá colaborar para o

funcionamento da aplicação. A variação da carga de entrada é responsável pelo entendimento do comportamento da aplicação .

A etapa de caracterização da aplicação é composta por 6 sub-etapas, sendo elas: (1) identificação das características do recurso; (2) seleção das chamadas de sistemas; (3) obtenção do comportamento da aplicação; (4) definição das métricas (5) variação de carga da aplicação (*workload*); e (6) definição do perfil da aplicação.

Na **identificação das características do recurso**, primeira sub-etapa, é identificar quais operações realmente impactam o desempenho do recurso computacional.

Como exemplo, ao analisar o recurso computacional disco, é necessário analisar as características de E/S de disco. Neste exemplo, as operações de escrita síncrona ou assíncrona fazem parte do perfil da aplicação, o que deve ser analisado, na fase posterior, para caracterizá-la corretamente. Outras características são se a operação é “bufferizada” ou tem Direct I/O, se o acesso é sequencial ou randômico. Essas características determinam como foi realizada a codificação da aplicação, sem que tenhamos acesso ao código fonte.

Sendo assim, a primeira sub-etapa da fase de caracterização da aplicação é levantar quais características relacionadas ao recurso computacional que devem ser analisadas.

Na **seleção das chamadas de sistemas**, segunda sub-etapa, deve-se identificar quais chamadas de sistemas que podem informar as características dos recursos. Será determinado como as chamadas de sistema estão relacionadas aos recursos.

No exemplo de I/O de disco, as chamadas de sistemas identificadas para serem analisadas são as *open, read, write, e lseek*, pois essas chamadas estão relacionadas ao I/O.

Os parâmetros que devem ser analisados para determinar as características citadas anteriormente (no Linux) são O_SYNC, DIRECT_IO, O_NONBLOCK na

primitiva *open*. A análise dessas das chamadas de sistemas e destes parâmetros permitem determinar o perfil inicial da aplicação.

Como exemplo, para determinar se o acesso é randômico, deve-se verificar se ocorre variação da posição de acesso à memória analisando-se a chamada de sistema `lseek`. Para determinar se as operações são mais de leitura ou de escrita, verifica-se a quantidade total de acesso de leitura e escrita, chamadas de sistemas `read` e `write`, respectivamente. Para determinar se as operações de I/O são bufferizadas ou se tem acesso direto ao disco, é preciso verificar se existe a opção DIRECT_IO na primitiva `open`. Para determinar se é síncrona ou assíncrona, será examinado o parâmetro na O_NONBLOCK na primitiva `open`.

Nesta sub-etapa são identificadas quais as chamadas de sistemas que correspondem às características do recurso computacional analisado.

O objetivo da **obtenção do comportamento da aplicação**, terceira sub-etapa, é determinar o tempo e a porcentagem de tempo gasto pelas chamadas de sistemas condizentes com o recurso que está sendo analisado.

Dependendo do recurso, é necessário também analisar quanto está sendo consumido do recurso computacional. Como exemplo, se o recurso analisado for o processador, o tempo médio de uso do processador também deve ser levado em consideração para a criação do perfil da aplicação.

A verificação do comportamento da aplicação é obtida através da execução da aplicação, dos resultados das métricas definidas anteriormente e do monitoramento das chamadas de sistemas.

Assim, esta sub-etapa determina como as chamadas de sistemas estão sendo realizadas para uma condição de carga e parâmetros.

No **definição das métricas,** quarta sub-etapa, definem-se as métricas que serão utilizadas para cada recurso computacional, tais como vazão e utilização. A métrica vazão (*throughput*) é a quantidade de trabalho realizado em um determinado tempo (*e.g.* operações de escrita por segundo). A utilização do recursos refere-se à

quantidade de tempo que o recurso ficou ocupado durante um período. Outras métricas, tais como saturação, latência e erro, também podem ser utilizadas. É necessário ter conhecimento do recurso computacional para conseguir selecionar adequadamente as métricas necessárias. Como exemplo, para o I/O de disco, pode ser utilizado a vazão de escrita de arquivo.

Na **variação de carga de trabalho da aplicação (*workload*),** quinta sub-etapa, executa-se a aplicação e monitora-se o recurso selecionado, com diferentes cargas de trabalho e variações de parâmetros da aplicação. Assemelha-se à etapa anterior. A diferença é que nessa etapa têm-se várias cargas e parâmetros a serem analisados.

Essa análise da carga de trabalho permite entender o comportamento geral da aplicação e é de suma importância para que possa posteriormente ser analisada na fase de Seleção e seja feito o planejamento da capacidade que a instância possa suportar.

Como exemplo, uma aplicação pode ter como entrada arquivos de diferentes tamanhos. Neste caso, seria possível variar a quantidade e o tamanho de arquivos. É necessário definir os casos mais importantes, quais os arquivos e tamanhos são interessantes e de real utilidade para o usuário da aplicação. Sendo assim, a ajuda do especialista da aplicação pode ser de grande valia para determinar os casos a serem analisados.

A **definição do perfil da aplicação,** sexta e última sub-etapa da caracterização da aplicação, apresenta o perfil final da aplicação com as características analisadas.

3.1.2. Caracterização da instância

A caracterização da instância fornece os procedimentos para a criação do perfil de um determinado tipo de instância que possa ser posteriormente utilizado na fase de seleção.

A Computação em Nuvem possui diferentes tipos de instâncias. Instâncias são servidores virtuais que podem executar aplicações. Os tipos de instâncias possuem uma variedade de combinações de recursos virtuais, tais como CPU, memória, *storage* e rede. Isso permite que se possa escolher a combinação de recursos mais adequada para a aplicação.

Apesar de ser um benefício ter uma grande variedade de tipos de instâncias, também é mais complicado escolher qual instância utilizar. Se forem considerados os vários provedores de Nuvem, a dificuldade na escolha do tipo de instância aumenta, levando-se em consideração a configuração, custo e desempenho.

Os provedores podem fornecer diversas quantidades de memoria e CPU. Também podem oferecem uma variedade de *storage*, podendo este ser local ou remoto, com diferentes tecnologias de disco (atualmente, HDD ou SSD). A dificuldade em comparar os diversos tipos de instâncias e a falta de padronização das mesmas aumenta a dificuldade de escolha. Outro problema é que a máquina física é compartilhada por diversas instâncias, e, apesar dos provedores de computação de nuvem garantirem um Nível de Serviço, o seu desempenho pode variar com o tempo.

O objetivo desta etapa é descrever os procedimentos necessários para a caracterização da instância e suas configurações no ambiente de Computação em Nuvem; é fornecer as diretrizes para criar um perfil para os tipos de instâncias. A vantagem é que esses perfis podem ser armazenados por diferentes usuários, criando assim um banco de dados incremental.

Nesta etapa, as variáveis dependentes (resultados) são as métricas que foram utilizadas na etapa de Caracterização da aplicação. As variáveis independentes são os tipos de instâncias. Esse é um processo interativo, e mais métricas podem ser adicionadas ou removidas. Entretanto, as etapas anteriores devem ser atualizadas.

Uma maneira de obter o desempenho, de maneira mais genérica, é utilizando benchmarks. O benchmark é um software composto por programas que buscam testar o desempenho dos recursos computacionais. Normalmente, os benchmarks

são compostos por micro-benchmarks, que são programas que testam o desempenho de um recurso computacional específico.

Sendo assim, através do benchmark e outras ferramentas de monitoramento, pode-se criar o perfil do tipo de instância para um determinado recurso computacional. Se o benchmark determinar o perfil da instância, então é possível determinar qual a melhor característica da instância e também é possível criar uma base de dados com diversos perfis.

Para a criação do perfil das instâncias foram definidas as seguintes sub-etapas (procedimentos): (1) escolha dos provedores de Nuvens (2) escolha dos tipos de instâncias; (3) definição das métricas; (4) seleção de método de avaliação; (5) execução do método de avaliação; e, (6) definição do perfil do tipo de instância.

Na **escolha dos provedores de Nuvens**, primeira sub-etapa, é necessário selecionar quais os provedores de Nuvens serão utilizados. Esses provedores podem ser públicos ou privados. Atualmente, existe uma diversidade de provedores públicos. Neste primeiro passo, é necessário definir um ou mais critérios de escolha. Uma lista de sugestões dos critérios de escolha é:

(a) popularidade e tempo de mercado;

(b) Tipo de licença de Software: o provedor pode oferecer licenças de software do OpenSource ou Proprietária. O provedor pode fornecer preços mais favoráveis para determinados softwares necessários para executar a aplicação em questão;

(c) Interface de controle: verificar quais são as possibilidade de interação do usuário com o provedor de Nuvem. Exemplos são Aplicação Web/Painel de Controle, API (*Application Programming Interface*), Linha de Comando e Interface Gráfica (*Graphical User Interface* – GUI);

(d) Suporte: verificar qual possui suporte, quais os serviços são oferecidos, fórum, *chat live*, recursos online ferramentas de Diagnósticos de problemas na instância;

(e) Especificações técnicas: quais sistemas operacionais são oferecidos, o tipo de processador (32bit ou 64 bits) e se possuem computadores dedicados;

(f) Preços: verificar os preços de instância mais barata ou instância com maior poder computacional, do tráfego de dados, os tipos de planos (paga pelo que usa ou planos de fidelização) e tráfego de dados gratuitos;

(g) Outras Características: podem ser fornecidos serviços (pagos ou gratuitos) que podem ser de interesse, tais como *Backup Storage*, snapshot backup, persistência, encriptação de dados, *AutoScaling*, Balanceamento de Cargas, Monitoramento e Serviço de Hospedagem de Arquivos.

Na **escolha dos tipos de instâncias**, segunda sub-etapa, define-se quais tipos de instâncias serão analisados em cada provedor de Nuvem. Devido à grande variedade e quantidade de tipos de instâncias, inicialmente, deve-se definir o objetivo da análise e relacioná-lo com o recurso computacional. Como exemplo, analisar os diversos *storages* das instâncias que tenham o mesmo custo. A definição do objetivo direciona-se na escolha inicial das instâncias e limita a quantidade de instâncias a serem analisadas. Posteriormente, novas instâncias podem ser analisadas.

Em **definição das métricas**, terceira sub-etapa, define-se as métricas relacionadas ao recursos. Estes recursos e métricas podem ser os mesmos utilizados na etapa de Caracterização da Aplicação.

Na **seleção do método de avaliação**, quarta sub-etapa, seleciona-se o método de avaliação para o recurso computacional a ser analisado. Como exemplo, a utilização de micro-benchmark. Este micro-benchmark deve possuir as métricas selecionadas anteriormente.

Em **execução do método de avaliação**, quinta sub-etapa, executa-se o método de avaliação. Como exemplo, executar o micro-benchmark. O uso de ferramentas de monitoramento pode ser utilizado para auxiliar no entendimento de detalhes específicos, como por exemplo variações de resultados. Com a execução são obtidos os resultados das métricas.

Em **definição do perfil do tipo de instância**, sexta sub-etapa, analisa-se o resultado do benchmark, cria-se o perfil da instância apresentando as principais características, e, por fim, armazena-se o resultado em uma base de perfil.

3.2. Seleção

A fase de Seleção é responsável por selecionar o tipo de instância mais adequado para uma determinada aplicação. Esta fase permite que a caracterização do tipo de instâncias e da aplicação seja utilizada. Para que se tenha uma boa seleção, é necessário que a caracterização tenha sido bem projetada e realizada.

Na caracterização, cria-se uma base de perfis, com diferentes características. Na fase de Seleção utilizam-se os dados dos perfis como entrada aplicando-se um algoritmo/método/procedimento para a seleção. Nesta fase, diversos algoritmos podem ser utilizados. Pode-se utilizar um ou mais recursos computacionais que foram previamente analisados na fase de caracterização.

Ainda nessa fase, define-se um ou mais recursos computacionais para serem analisados e também as métricas. A caracterização é responsável pela obtenção dos dados e pela primeira análise dos dados para a criação do perfil. A seleção é responsável pela análise dos perfis e pela escolha do perfil mais adequado. Assim, é necessário definir um ou mais recursos computacionais e as métricas a serem utilizadas com base nos perfis disponíveis, ou seja, com base nos tipos de instâncias já caracterizadas.

A fase de seleção apresenta os procedimentos necessários à seleção da instância mais adequada para uma determinada aplicação, considerando fator de escolha (*e.g.* custo ou desempenho). O objetivo desta fase é apresentar os procedimentos necessário para a seleção da instância. Se o procedimento de seleção fornecer a instância mais adequada, então pode-se reduzir o custo financeiro e/ou o tempo de execução da instância.

Como resultado, tem-se a atribuição do tipo de instância para a aplicação. As variáveis que podem ser manipuladas nessa fase são recurso, métrica, perfil de

caracterização da instância. A estimativa de duração de execução da aplicação também pode ser adicionada para auxiliar no resultado.

Os procedimentos da fase de seleção são: (1) escolha da aplicação (2) escolha do recurso computacional preponderante; (3) definição de métricas e do fator de escolha; e (4) seleção da instância.

Na **escolha da aplicação**, primeira etapa, define-se qual aplicação vai ser utilizada. As características da aplicação encontra-se na base de perfil da aplicação. Ressalta-se que várias aplicações podem ser caracterizadas ao mesmo tempo. Sendo assim, nesta fase é necessário definir qual aplicação será utilizada.

Na **escolha do recurso computacional preponderante**, segunda etapa, define-se um ou mais recursos a serem utilizados. A escolha do recurso deve ser realizada a partir dos recursos previamente analisados que constam no perfil da aplicação. Como exemplo, ao analisar os recursos CPU e memória, verifica-se que a aplicação é do tipo CPU bound, logo o recurso a ser selecionado é a CPU.

Na **definição de métricas e do fator de escolha**, terceira etapa, são definidas quais métricas relacionadas com o recurso serão utilizadas, como exemplo, vazão, latência. No fator de escolha, é definida qual é a medida decisiva na escolha, como por exemplo desempenho ou custo. Outros elementos podem ser adicionados para auxiliar na seleção da instância, como por exemplo, estimativa do usuário de tempo de execução da aplicação. Neste exemplo, essa estimativa pode auxiliar na definição do custo de execução de uma aplicação nas instâncias.

Na **seleção da instância**, quarta etapa, seleciona-se qual instância é mais adequada para a aplicação. Utilizam-se os perfis de instâncias que contenham o recurso computacional e as métricas definidas anteriormente para realizar a seleção. Nesta etapa, é possível utilizar algoritmos de seleção simples, como utilizar a instância com menor custo, ou algoritmos mais complexos, como técnicas de reconhecimento de padrões ou técnicas estatísticas de análise multivalorada de dados.

De qualquer forma, para que esses algoritmos possam ser utilizados na seleção (e/ou extração de conhecimento) é necessário que os dados tenham sido corretamente coletados e analisados. Ou seja, a fase de caracterização deve ter sido feita corretamente e as métricas utilizadas nelas também. Isso demonstra a importância da fase de caracterização.

Esta etapa fornece a relação do tipo de instância para aplicação. Essa informação é utilizada posteriormente na fase de execução.

3.3. Execução

A fase de execução é responsável por criar a instância, configurá-la e executar a aplicação. Com base na escolha da instância, feita na fase de seleção, esta fase cria a instância e configura o ambiente de execução para a execução da aplicação. A fase de execução também verifica a necessidade de manter uma imagem no provedor de Nuvem.

Cada vez que uma aplicação vier a ser executada, é necessário criar o ambiente de execução. Portanto, ter uma imagem com a configuração específica para esta aplicação permite o ganho de tempo da configuração. A fase de execução também é responsável por analisar a quantidade de execuções da aplicação e por manter uma imagem no provedor de Nuvem.

A fase de execução é dividida em 4 etapas, sendo elas: (1) escolha da imagem; (2) criação da instância; (3) configuração da instância; (4) execução da aplicação.

Na **escolha da imagem**, primeira etapa, é averiguado se o provedor de Nuvem já possui uma imagem com as configurações necessárias para atender os requisitos da aplicação. Nessa etapa, define-se qual imagem será utilizada para a criação da instância. Pode ser escolhida uma imagem que não tenha as configurações parciais do ambiente necessário a execução.

Em **criação da instância**, segunda etapa, a instância é criada a partir da imagem previamente definida.

Na **configuração da instância**, terceira etapa, configura-se o ambiente para a execução da aplicação. Caso a imagem selecionada não tenha todos os requisitos para execução, esta etapa é responsável por instalá-los e configurá-los.

Na **execução da aplicação**, executa-se a aplicação na instância criada.

3.4. Resumo dos Procedimentos da Metodologia CSE

Esta seção apresenta um resumo dos procedimentos da Metodologia CSE, que foram apresentados nas seções anteriores.

1. Caracterização

1.1. Caracterização da Aplicação

1.1.1. Identificação das características do recurso;

1.1.2. Seleção das chamadas de sistemas;

1.1.3. Obtenção do comportamento da aplicação;

1.1.4. Definição das métricas;

1.1.5. Variação da carga de trabalho da aplicação (*workload*);

1.1.6. Definição do perfil da aplicação.

1.2. Caracterização da Instância

1.2.1. Escolha dos provedores de Nuvens ;

1.2.2. Escolha dos tipos de instâncias;

1.2.3. Definição das métricas;

1.2.4. Seleção de método de avalição;

1.2.5. Execução do método de avaliação;

1.2.6. Definição do perfil do tipo de instância.

2. Seleção

2.1. Escolha da aplicação;

2.2. Escolha do recurso computacional preponderante;

2.3. Definição de métricas e fator de escolha;

2.4. Seleção da instância.

3. Execução

3.1. Escolha da imagem;

3.2. Criação da instância;

3.3. Configuração da instância;

3.4. Execução da aplicação.

3.5. Conclusão

A metodologia CSE, apresentada neste capítulo, fornece os procedimentos necessários para auxiliar na determinação do recurso computacional preponderante da aplicação. Também, possibilita definir a caracterização da aplicação e os tipos de instâncias. Isso permite definir o tipo de instância mais adequado para uma aplicação específica, sem necessidade de executar a aplicação em todos os tipos de instância.

Entretanto, com a utilização de diversos provedores de Nuvem, amplia-se a quantidade de tipos de instância. Por consequência, a dificuldade de execução da aplicação aumenta, em razão das diferentes interfaces existentes para criação de instâncias, na medida em que cada provedor fornece a sua própria interface de gerenciamento. Adiciona-se a isto a necessidade de configuração da instância, necessária à execução da aplicação.

Por este motivo, existe a necessidade de uma infraestrutura que auxilie na seleção da instância e na execução da aplicação, permitindo uma interligação entre provedores de nuvens públicas e privadas, a fim de facilitar a utilização da metodologia.

4. SISTEMA DE INTERLIGAÇÃO DE PROVEDORES DE NUVENS PÚBLICAS E PRIVADAS

A interligação entre diversos provedores de Nuvens permite que o usuário não fique restrito às políticas de uma única Nuvem e também amplia as possibilidades de utilização de diversas instâncias de Máquinas Virtuais. Neste capitulo são apresentadas as arquiteturas e a implementação do InterCloud LAHPC que permite a interligação entre os provedores de Nuvens públicas e privadas.

A construção de uma plataforma visando à interoperabilidade pode ser realizada de duas maneiras: migração da Máquina Virtual ou por contêiner. A imagem é composta pelo sistema operacional e pelas aplicações. Cada Software de Virtualização possui seu próprio formato de imagem. Existe também a imagem da Máquina Virtual, esta contém o Sistema Operacional (SO), aplicações e estado da MV. O problema de migração de MVs está relacionado aos diferentes formatos de imagens de MVs utilizados por cada Software de Virtualização. Outra solução é a utilização de *contêineres*, ou seja, um conjunto de serviços. Esses serviços são monitoramento, execução, escalonamento. O contêiner é instalado em cada MV, fornecendo subsídios para que a aplicação seja executada e possibilitando a migração da aplicação de maneira transparente aos usuários.

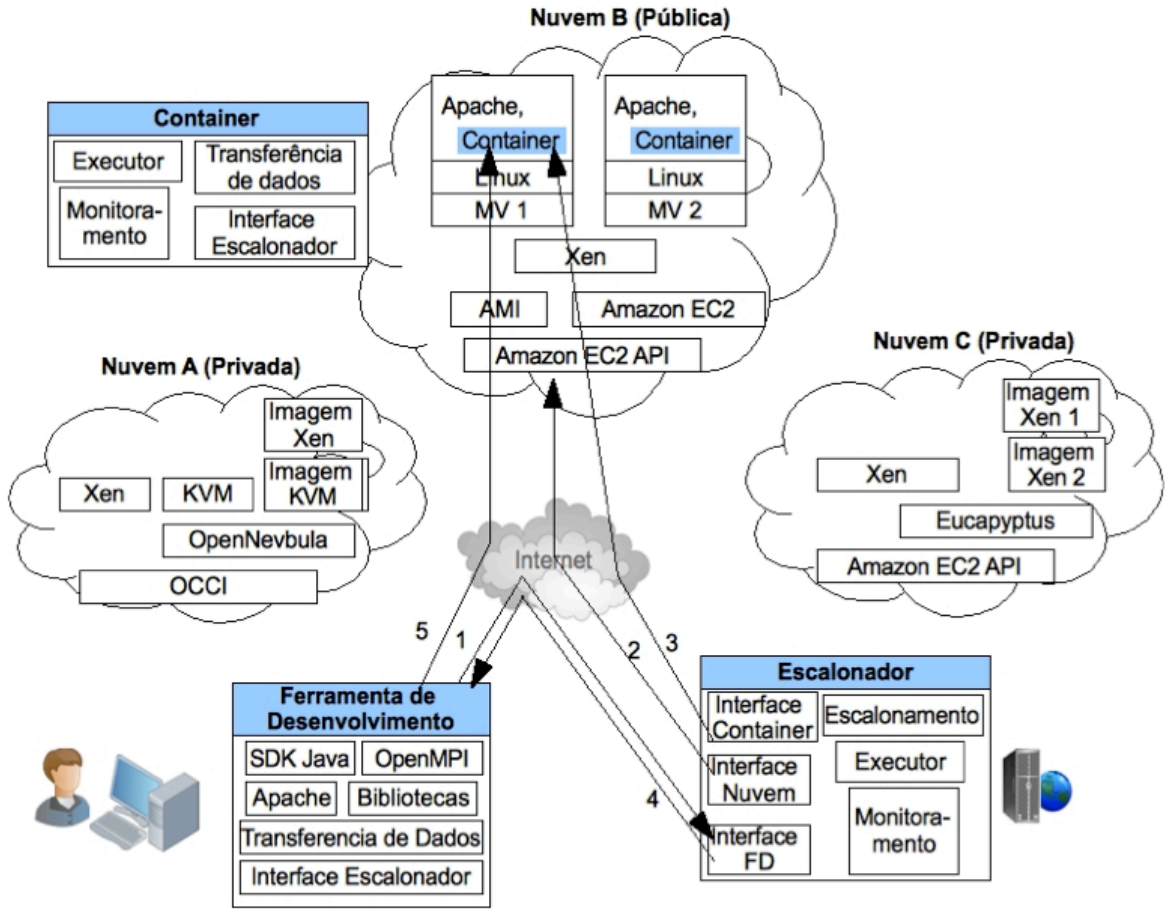
A seleção da Nuvem envolve uma avaliação dos serviços providos. Avalia-se a Qualidade de Serviço (*Quality of Service* - QoS), tais como disponibilidade, confiabilidade e desempenho. O monitoramento dos recursos auxilia na tomada de decisões para o balanceamento de cargas e para a criação de mais MVs. O usuário pode fornecer informações sobre as regras do seu próprio negócio. Como exemplo, depois dos feriados as pessoas costumam publicar suas fotos nos álbuns online. Uma empresa deste ramo pode informar ao escalonador que haverá um aumento no acesso em seu site. Assim, o escalonador pode selecionar outra Nuvem que forneça um melhor preço para uma maior quantidade de acessos. O escalonador pode separar aplicações e Banco de Dados. Aplicações podem ficar em uma Nuvem e os dados distribuídos em diferentes Nuvens, evitando o aprisionamento e problemas com leis do país na qual a Nuvem está instalada.

O usuário utiliza a Ferramenta de Desenvolvimento (FD) para a criação de suas aplicações. A FD fornece suporte a diferentes modelos de programação, tais como programação sequencial e paralela. FD é composta por compiladores, aplicações e bibliotecas, permitindo ao usuário desenvolver sua aplicação em seu computador e depois enviar para a Nuvem. A aplicação pode ser desenvolvida localmente ou remotamente. Se o usuário utilizar programas de desenvolvimento gratuitos, tal como o compilador GCC (GNU, 2014), é mais econômico desenvolver, fazer os testes do programa localmente e, por fim, executá-lo na Nuvem. Se o usuário optar pela utilização de programas de desenvolvimento pagos, tal como o compilador da Intel (INTEL, 2014), o usuário pode desenvolver seu programa remotamente em uma Nuvem que forneça o programa como serviço. O FD possui uma interface para comunicar-se com o Escalonador e um programa para transferir a aplicação (*e.g.* cliente FTP).

O usuário já deve ter contas em diferentes Nuvens e precisa informar ao escalonador os dados para acessá-las (*e.g.* certificado digital). O escalonamento é baseado no QoS, como disponibilidade, confiabilidade, desempenho e regras de negócios. O Executor é responsável por solicitar a criação das MVs. O Escalonador pode ser executado em uma máquina dedicada com acesso à Internet ou em uma MV de uma Nuvem. Escalonador possui interfaces para se comunicar com a FD, Contêiner e com diversas Nuvens Públicas. Implementa funções para criar, suspender, reiniciar e destruir MVs.

Contêiner é projetado para ajudar a organizar e armazenar itens que são colocados dentro dele (ECKEL, BRUCE, 2006). Uma classe contêiner é uma classe projetada para organizar múltiplas instâncias de outras classes. Um contêiner pode implementar um ambiente para a execução dos serviços. Sendo assim, sua principal responsabilidade é inicializar os serviços quando necessário. O contêiner fica hospedado nas MVs da Nuvem para executar e monitorar a MV e a aplicação. A transferência de dados é realizada por uma aplicação (*e.g.* servidor FTP). O Executor é responsável por executar o código transferido pela FD. Monitoramento envia informações de memória, CPU e rede para o Escalonador através da Interface Escalonador.

Figura 10 - Arquitetura de Plataforma para Computação em Nuvem.



Fonte: Autoria própria.

A execução de uma aplicação do usuário segue os seguintes passos, conforme apresentado na Figura 10: (**1**) depois de compilada a aplicação, a FD solicita ao Escalonador a Nuvem e a Máquina Virtual a ser utilizada; (**2**) o Escalonador aplica o escalonamento, analisando os preços de cada Nuvem, o QoS, selecionando qual Nuvem a ser utilizada; com o intuito de aumentar a interoperabilidade, alguns compiladores e aplicações podem ser instalados na MV utilizando o *Contêiner*. O Executor prepara o ambiente solicitado pela FD, utilizando a interface da Nuvem (*e.g.* Amazon EC2 API), cria as MVs a partir de uma imagem (*e.g.* AMI) e instala o Contêiner; (**3**) o Escalonador comunica-se com o Contêiner através da Interface *Contêiner* e inicializa o Monitoramento; (**4**) o Escalonador informa à FD a Nuvem escolhida e as informações necessárias para iniciar a Transferência de Dados; (**5**) a FD faz a Transferência dos Dados, que inclui o

programa da aplicação e os dados relacionados, para o Contêiner. Ao terminar a transferência, o Contêiner inicia a execução da aplicação.

Após o início da execução, o Escalonador continua monitorando o Contêiner a fim de averiguar a Qualidade de Serviço prestada. Novas informações podem ser adicionadas ao Escalonador, por exemplo, modificação de preços das Nuvens, acréscimo ou remoção de novas regras de negócio via usuário. Com base nessas informações, realiza-se novamente um escalonamento para a aplicação. Nessa fase de reescalonamento leva-se em consideração a viabilidade de migração do código da aplicação ou da MV. Para a migração do código da aplicação, é necessário observar a compatibilidade da aplicação com a versão do Sistema Operacional. Para a migração da MV, é necessário que a Nuvem execute o mesmo formato de imagem da MV. Em ambos os tipos de migração, o tempo para suspender/migrar/reiniciar a aplicação ou a MV é levado em consideração pelo escalonamento.

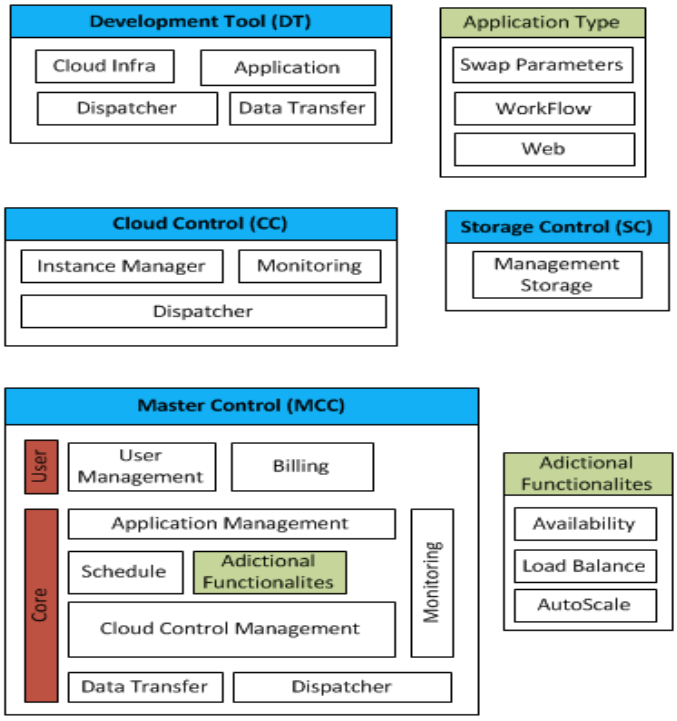
4.1. Descrição Aprimorada da Arquitetura proposta

O foco desta seção é propor uma arquitetura com integração entre duas Nuvens (*InterCloud*), usando tanto Nuvem pública quanto privada.

Essa seção descreve os módulos e componentes que fazem parte da arquitetura. Os componentes que fazem parte da arquitetura são: Ferramenta de Desenvolvimento (*Development Tool* - DT), Controle da Nuvem Mestre (*Master Cloud Control* - MCC), Controle da Nuvem (*Cloud Control* - CC) e Controle de Armazenamento (*Storage Control* - SC), conforme mostrado na Figura 11.

Ferramenta de Desenvolvimento (*Development Tool* - DT) é o *front-end* do desenvolvedor. Nesta ferramenta, o usuário inicia e gerencia ambientes compostos por vários provedores de Nuvens. Os usuários submetem as aplicações para executar na Nuvem. DT tem duas principais funções: (a) gerenciar, inicializar e monitorar a infraestrutura de Nuvem, (b) *front-end*, servir de *front-end* para os usuários submeterem suas aplicações.

Figura 11 - Arquitetura do InterCloud.



Fonte: Autoria própria.

O módulo *Cloud Infra* é usado para criar a infraestrutura da Nuvem. Em seguida, tarefas podem ser executadas nas instâncias da Nuvem.

O módulo Aplicação (*Application*) é responsável por submeter as aplicações e provê suporte para os seguintes tipos de aplicações: (a) Troca de Parâmetros (*Swap Parameters*), onde uma aplicação é executada várias vezes com diferentes parâmetros; (b) *Workflow*, onde as tarefas são executadas com fluxos preestabelecidos; e (c) Web, onde a aplicação pode ser utilizada por navegadores web ou outros serviços Web.

O DT utiliza o módulo Expedidor (*Dispatcher*) para comunicar-se com o componente Controle de Nuvem Mestre (*Master Cloud Control* - MCC) e o módulo Transferência de Dados (*Data Transfer*) para transferir dados para as instâncias de Máquinas Virtuais ou outros serviços de Nuvem.

O componente Controle de Nuvem (*Cloud Control* - CC) é responsável, por criar, gerenciar e monitorar as instâncias dos provedores de Nuvens. Cada provedor de Nuvem que provê serviços de computação tem esse componente. O Controle de Nuvem também é responsável por monitorar se o Controle de Nuvem Mestre (*Master Cloud Control* - MCC) está funcionando corretamente e por assumir em caso de falha. Esse componente tem os módulos Gerenciador da Instância (*Instance Manager*), Monitoramento (*Monitoring*) e Expedidor (*Dispatcher*).

Gerenciador de Instância (*Instance Manager*) é responsável por criar e deletar instâncias. O módulo Monitoramento (*Monitoring*) é responsável pelo monitoramento das instâncias e por checar se o MCC está funcionando. O CC usa o módulo Expedidor (*Dispatcher*) para comunicar-se com o Controle de Nuvem Mestre (*Master Cloud Control* - MCC) e com o provedor de Nuvem.

O componente Controle de Armazenamento (*Storage Control* - SC) é responsável pelo armazenamento das aplicações e de dados e também por coordenar a localização das aplicações e dos dados. O módulo Gerenciador de Armazenamento (*Management Storage*) é responsável por esse gerenciamento.

Controle de Nuvem Mestre (*Master Cloud Control* - MCC) é o principal componente. É responsável por gerenciar todos componentes, especialmente todos os componentes do Controle de Nuvem (*Cloud Control* - CC). O MCC tem as funcionalidades do Controle de Nuvem no qual ele está instalado. MCC é dividido em dois grupos lógicos: Usuário (*User*) e Usuário Principal (*Core User*).

A camada Usuário (*User*) tem o módulo Gerenciador de Usuário (*User Management*), que é responsável por adicionar usuários e armazenar seus certificados/*tokens*. Os certificados/*tokens* são mecanismos de segurança para utilizar os provedores de Nuvens. O módulo Faturamento (*Billing*) é responsável por gerenciar o tempo de uso dos usuários e enviar para eles o faturamento.

As principais características do sistema estão na camada *core*. Ela tem os módulos Gerenciador de Aplicação (*Application Management*), Escalonador (*Schedule*), Gerenciador de Controle de Nuvem (*Cloud Control Management*), Transmissão de Dados (*Data Transfer*), Expedidor (*Dispatcher*) e Monitoração

(*Monitoring*). Os módulos de Disponibilidade (*Availability*), Balanceamento de Cargas (*Load Balance*) e AutoEscala (AutoScale) podem ser adicionados para poder prover funcionalidades adicionais.

O Gerenciador de Aplicações (*Application Management*) é responsável pelo gerenciamento das aplicações. Usando o DT, usuários podem submeter aplicações para ele, o qual é responsável por identificar o tipo de aplicação (*Swap Parameters*, *Workflow* ou *Web*) e gerenciá-las.

No tipo de aplicação *Workflow*, esse módulo analisa as precedências das tarefas. Inicialmente, as tarefas sem precedências serão enviadas ao Gerenciador de Controle de Nuvem (*Cloud Control Management*) para serem executadas. Uma vez completadas, o Gerenciador de Aplicações (*Application Management*) recebe a notificação e submete as tarefas que têm precedência com as tarefas concluídas. Esse passo é repetido até que todas as tarefas tenham sido concluídas.

No tipo de aplicação de Troca de Parâmetros (Swap Parameters), módulo divide a aplicação em tarefas para serem executadas com diferentes parâmetros. Depois, ele envia essas tarefas para o módulo Gerenciador de Controle de Nuvem (*Cloud Control Management*).

No tipo de aplicação Web, o Gerenciador de Aplicação (*Application Management*) analisa as requisições necessárias para a construção do ambiente (número de máquinas, servidor Web, linguagem de programação) e submete-as ao Gerenciador de Controle de Nuvem (*Application Management*).

Escalonador (Schedule) é responsável por escalonar as tarefas, verificando qual provedor de Nuvem pode ser utilizado. Ele pode focar baixo custo e alto desempenho.

O Gerenciador de Controle de Nuvem (*Cloud Control Management*) é responsável por gerenciar os componentes de Controle de Nuvem (*Cloud Control*) dos provedores de Nuvens. Eles são enviados para o Controle de Nuvem (*Cloud Control*) para criar e deletar instâncias de suas Nuvens.

O Monitoramento (*Monitoring*) é responsável por monitorar as operações do Controle de Nuvem e verificar as operações das instâncias.

O módulo Expedidor (*Dispatcher*) tem três funções principais dentro do componente MCC. Ele é responsável por enviar mensagens para os provedores de Nuvem (como Rackspace e Amazon), comunicar-se com as instâncias e outros componentes da arquitetura (DT e CC).

O módulo Transferência (*Data Transfer* - DT) tem a mesma função do componente DT do MCC.

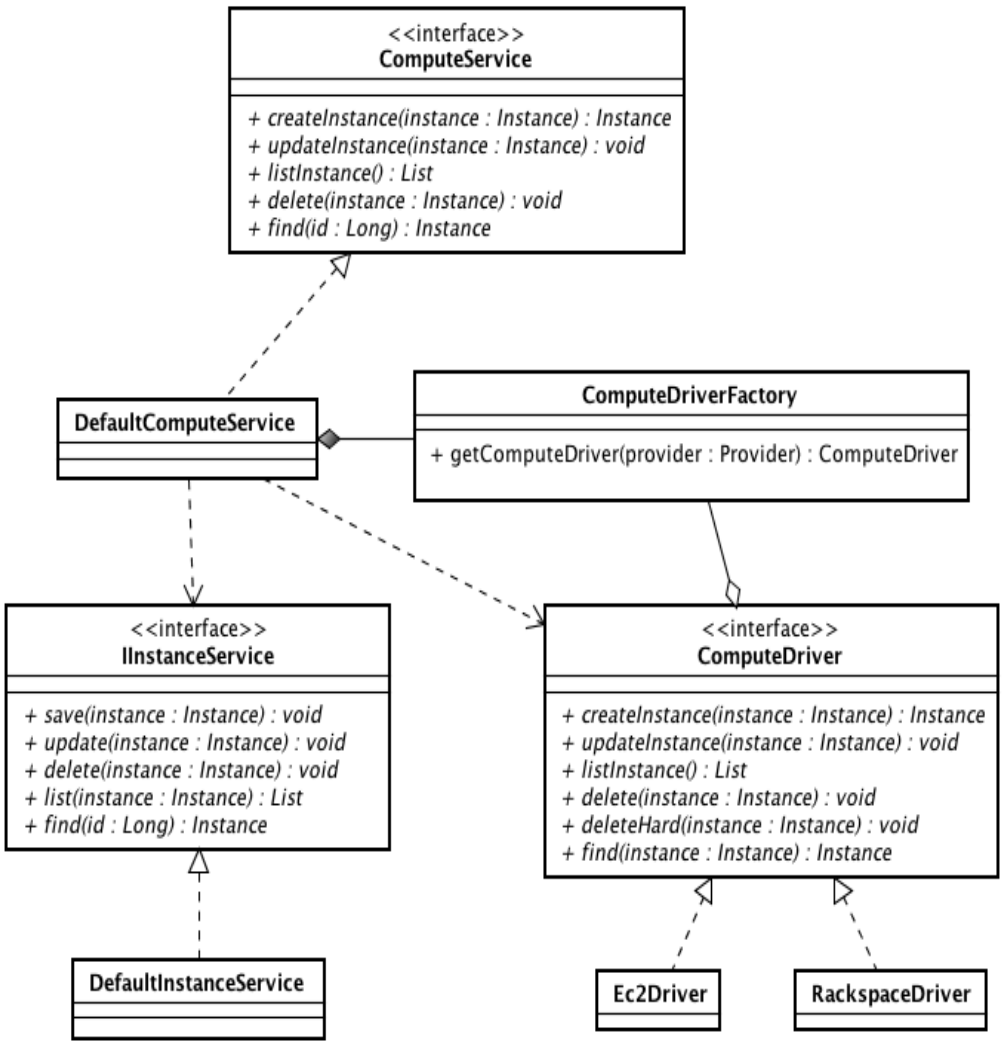
Funcionalidades adicionais podem ser adicionadas dependendo do tipo de aplicação, como por exemplo, (a) Disponibilidade (Availability) é responsável por disponibilidade para a aplicação. A aplicação e os dados podem ser replicados para alta disponibilidade; (b) Balanceamento de Cargas (Load Balance) é responsável pela distribuição de requisições entre as instâncias. O MCC pode criar uma ou mais instâncias para realizar o balanceamento de cargas; e (c) AutoEscala (AutoScale) define as regras para a escalabilidade automática das aplicações e a criação de novas instâncias. Como exemplo, quando a aplicação Web recebe um pico de requisições, a aplicação pode ser duplicada em outra instância durante esse pico.

Os seguintes passos são necessários para executar a aplicação: (a) o Gerenciador de Aplicação (*Application Management*) requisita a criação e configuração da instância para o Gerenciador de Controle de Nuvem (*Cloud Control Management*); (b) o Gerenciador de Controle de Nuvem (*Cloud Control Management*) se comunica com o Escalonador (*Schedule*), e então cria e configura as instâncias; (c) após a execução, o Gerenciador de Controle de Nuvem (*Cloud Control Management*) informa ao Gerenciador de Aplicação (*Application Management*) que a tarefa foi completada. O Gerenciador de Aplicação (*Application Management*) submete a outra tarefa pendente.

4.2. Descrição da Implementação da Arquitetura

Esta seção descreve a implementação do Provedor de Nuvem (*Cloud Provider*) e de Conexão de Instância (*Instance Connection*). Nesse estudo foram implementados vários módulos das camadas. Os módulos não implementados são: Monitoramento (*Monitoring*), Gerenciamento do Usuário (*User Management*), Fatura (*Billing*) e Funções Adicionais (*Additional Functions*). O tipo de aplicação implementada foi a Troca de Parâmetros (*Swap Parameters*). A Figura 12 apresenta o diagrama de classe para a comunicação entre os provedores.

Figura 12 - Diagrama de Classes da Implementação da Arquitetura.



Fonte: Autoria própria.

A interface ServiçoDeComputação (*ComputeService*) é responsável por definir os métodos primários para o gerenciamento das instâncias dos provedores das Nuvens. Essa mostra os métodos comuns para criação, atualização, listagem remoção e busca das instâncias. Os métodos definidos são:

- `createInstance()`: cria as instâncias no Provedor de Nuvem;
- `updateInstance()`: atualiza informações relacionadas a instância (*e.g.* senha de acesso via protocolo ssh);
- `listInstance()`: lista todas as instâncias criadas;
- `deleteInstance()`: destrói a instância especificada de forma assíncrona, ou seja, não aguardando a resposta de que foi realmente finalizada;
- `findInstance()`: busca as informações relacionadas a um determinada instância (*e.g.* usuário criador da instância).

O método adicional em ComputeDriver é:

`deleteInstanceHard()`: destrói a instância especificada de forma síncrona, ou seja, aguarda a confirmação de que a instância foi realmente destruída.

Foram desenvolvidos os *drivers* de comunicação com os provedores Rackspace (`RackspaceCloudProvider`) e Amazon EC2 (`Ec2CloudProvider`). Os *drives* podem ser utilizados para a comunicação com os provedores afim de permitir a criação de uma infraestrutura privada. Como exemplo, o OpenNebula cria uma infraestrutura de Nuvem Privada e provê uma API para o gerenciamento das instâncias. A plataforma apresentada integra a Nuvem privada criada com o OpenNebula com a Nuvem pública, como Amazon e Rackspace.

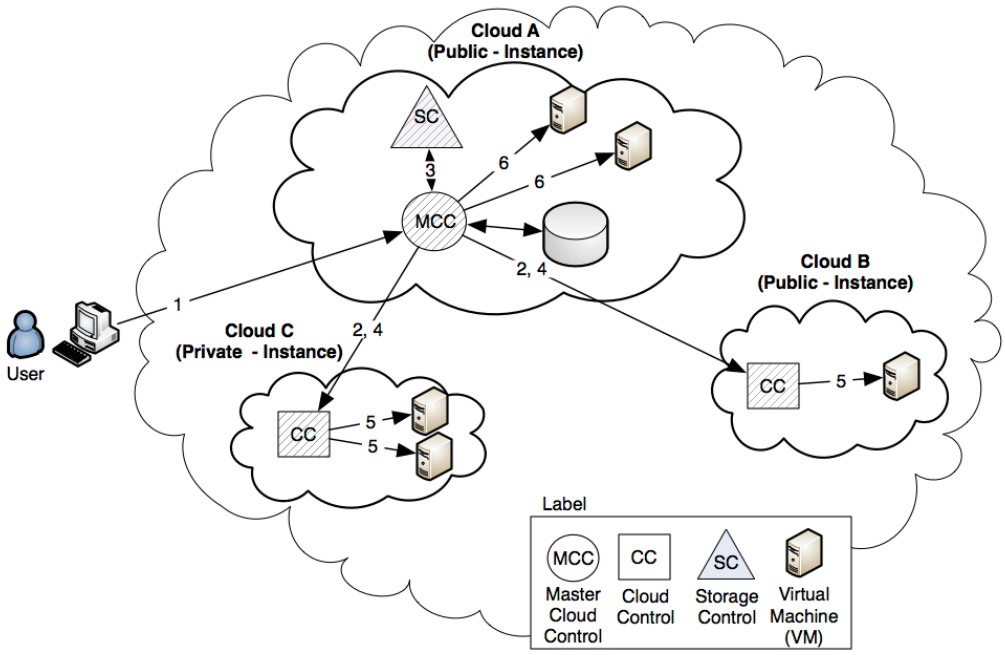
Para se comunicar com as instâncias, os principais métodos identificados são: `exec`(), para executar comandos nas instâncias, `copyToRemote`() e `copyToLocal`(), para copiar os dados.

4.3. Criação da Infraestrutura

Esta seção descreve a criação da infraestrutura. Essa infraestrutura é necessária para a submissão das aplicações.

A fim de criar a infraestrutura para prover a integração entre provedores de Nuvens, o usuário utiliza o componente Ferramenta de Desenvolvimento (*Development Tool* – DT). O usuário informa quais os provedores de Nuvem a serem utilizados, tão quanto as credenciais / *tokens* que fornecem o acesso para esses provedores. O usuário também informa qual o armazenamento (*storage*) será utilizado e qual o principal provedor de Nuvem será considerado o principal. Então, a criação da estrutura é iniciada, conforme a Figura 13.

Figura 13 - Criação da Infraestrutura.



Fonte: Autoria própria.

(1) DT cria uma instância no provedor de Nuvem principal, instala o componente MCC e informa qual provedor de Nuvem será utilizado.

(2) MCC cria uma instância em outro provedor de Nuvem e instala o componente CC nessas instâncias.

(3) MCC envia a localização do local de armazenamento para o Controle de Armazenamento (*Storage Control*). A localização pode ser na mesma instância em que o MCC está instalado ou em outro serviço de armazenamento. Nesse exemplo, a localização é na mesma instância do MCC.

(4) Quando requisitado pelo Controle de Nuvem Mestre (MCC),

(5) Controle de Nuvem (Cloud Control – CC) cria a instância.

(6) O MCC também desempenha as funções do Controle de Nuvem (CC) que gerencia a Nuvem. Nesse passo, o MCC cria a instância em sua própria Nuvem.

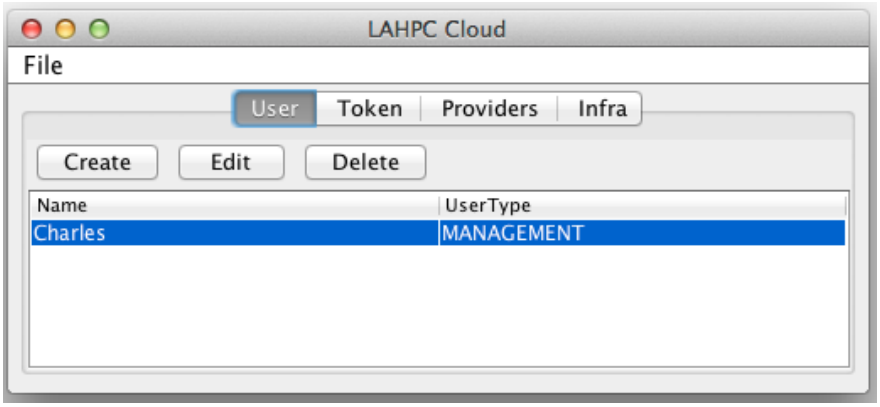
4.4. Exemplo de Criação de Infraestrutura

Nessa seção, é apresentada a interface do usuário e os passos necessários para a criação da infraestrutura utilizando Nuvens privadas (OpenNebula) e públicas (Rackspace e Amazon Ec2), implementando-se a arquitetura proposta.

Criou-se uma interface gráfica (GUI) para o Usuário Administrador com 4 abas (Usuário, Token, Provedor e Infra), como mostra a Figura 14. Todas as abas podem criar, editar ou deletar. A aba Usuário é responsável por definir se o usuário é administrador ou se é um usuário básico. A aba Token possui informações sobre o acesso de segurança que é usado para se comunicar com os provedores (como id, chave, certificação). A aba Provedor contém informações sobre os provedores de serviços (Amazon, Rackspace, OpenNebula), assim como os Drivers de acesso (Ec2 ou Rackspace) e o Token. Na aba Infra, cria-se a Infraestrutura de Nuvem Virtual. Selecionam-se os provedores para a criação e assim pode-se construir a infraestrutura.

Foi implementada a persistência em um banco de dados e a criação da infraestrutura. O teste de funcionalidade, utilizando a Amazon Ec2, Rackspace e OpenNebula, é descrito a seguir.

Figura 14 – Interface Gráfica para Criação de Usuário.

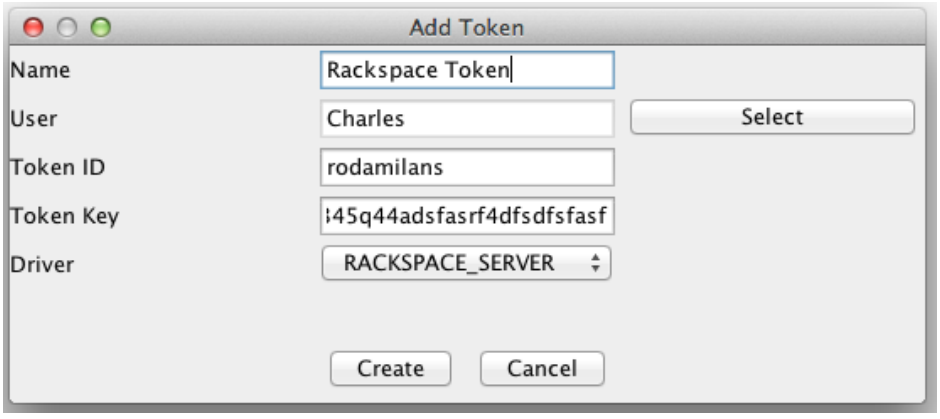


Fonte: Autoria própria.

No primeiro passo é necessário criar o usuário administrador (*admin user*) para gerenciar o sistema, sendo este responsável por criar, editar e deletar o Usuário, Token, Provedor e Infraestrutura (Infra). A Figura 14 mostra a GUI do sistema com o usuário sendo criado.

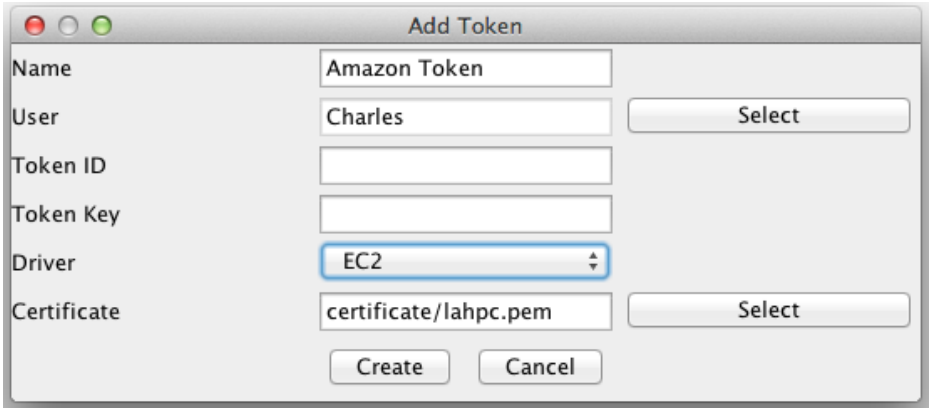
Depois, adicionam-se os Tokens para ter permissão de interagir com o provedor de Nuvem, e assim, poder criar MVs. Existem duas formas para o acesso: (a) Através de um Login (id) e Senha (Chave), conforme a Figura 15, ou (b) utilizando certificado (Figura 16).

Figura 15 - Interface Gráfica para criação de Token com Login e Senha.



Fonte: Autoria própria.

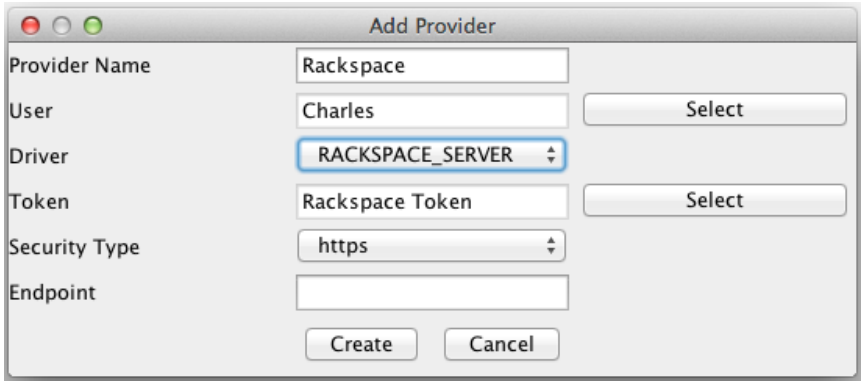
Figura 16 - Interface Gráfica para criação de Token com certificado.



Fonte: Autoria própria.

No terceiro passo, adiciona-se os servidores e são especificados as características de configuração dos provedores, tais como protocolo de acesso (http ou https) e endereço (*endpoint*), conforme a Figura 17. Se deseja utilizar uma região específica, pode-se mudar o *endpoint* para a região desejada. Neste exemplo, adicionou-se o provedor de Rackspace para o Usuário (User) Charles, utilizando o driver de acesso RACKSPACE_SERVER, com o Token previamente criado (Rackspace Token) com o protocolo *https*.

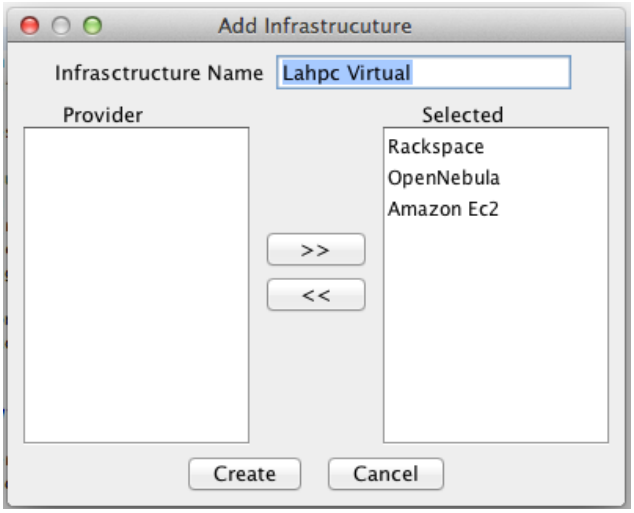
Figura 17 - Interface Gráfica para adicionar provedor de Nuvem.



Fonte: Autoria própria.

No quarto passo, foram selecionados os provedores que irão fazer parte da Infraestrutura Virtual, Figura 18. O primeiro da lista é o MCC e os demais são CCs. Finalmente, cria-se a infraestrutura com o botão Iniciar (*Start*), Figura 19.

Figura 18 - Interface Gráfica para Criação de Infraestrutura.



Fonte: Autoria própria.

Figura 19 - Interface Gráfica para Iniciar (Start) ou Parar (Stop) a Infraestrutura.

Fonte: Autoria própria.

4.5. Conclusão

Este capítulo apresentou uma proposta de arquitetura para interligação de provedores de Nuvens públicas e privadas. Além disso, foi implementado a API para a comunicação com os provedores públicos da Amazon e Rackspace. Está API

também pode ser utilizada nos provedores de Nuvem privadas que utilizem, por exemplo, a infraestrutura de gerenciamento OpenNebula.

O próximo capítulo apresenta estudos de casos para a avaliação de nuvens públicas e privadas, e também para a aplicação da Metodologia CSE.

5. RESULTADOS E ESTUDO DE CASO

Este capítulo apresenta um estudo de caso sobre a análise de desempenho de nuvens públicas e também um estudo de caso com a utilização da metodologia CSE para sua validação. O objetivo dos próximos capítulos é apresentar e comparar o desempenho das nuvens públicas e mostrar a viabilidade do uso da metodologia CSE em nuvens públicas e privadas, utilizando uma aplicação na área de biodiversidade. Os procedimentos da metodologia são aplicados empregando a aplicação OpenModeller e infraestruturas de nuvens públicas e privadas.

5.1. ANÁLISE DE DESEMPENHO DE NUVENS PÚBLICAS

O presente trabalho avalia o desempenho de I/O na Amazon EC2. A análise de desempenho também é realizada no Rackspace. Amazon EC2 e Rackspace foram selecionados por serem os provedores de Infraestrutura como Serviço (IaaS) mais populares. Os detalhes desse trabalho podem ser vistos em (RODAMILANS; BARUCHI; MIDORIKAWA, 2014).

Foram utilizados dois tipos de instâncias, *small* e *large*, e dois tipos de armazenamento em cada provedor de nuvem, disco (efêmero) e bloco (persistente). A tecnologia de armazenamento SSD também é analisada. O benchmark IOZone é utilizado para avaliação dos armazenamentos usando operações de modos sequenciais (escrita e reescrita) e randômica (escrita e leitura). Todas as operações são síncronas pois este é o tipo mais utilizado pelas aplicações de banco de dados. O paralelismo foi alcançado utilizando *multithreading* e é analisado com a variação de 1 até 5 *threads*.

A categorização da instância e do armazenamento em nuvens públicas permite selecionar qual o provedor de nuvem a ser utilizado. Esta seleção pode ser realizada focando em desempenho ou custo. A escolha da instância mais cara e com hardware mais poderoso não é garantia de que o desempenho será melhor.

5.1.1. Descrição dos tipos de Instâncias e armazenamento dos Provedores

Essa seção mostra um resumo sobre os tipos de instâncias e de armazenamento do Amazon EC2 e Rackspace. As informações sobre as instâncias estão resumidas na Tabela 4.

Tabela 4 - Descrição dos tipos de instâncias selecionadas

Provedor	Amazon EC2		Rackspace	
Tipo de Instância	ec2.small	ec2.large	rack.small	rack.large
Nome da API	m1.small	hi1.xlarge	1GB RAM	15 GB RAM
Arquitetura	64 bits	64 bits	64 bits	64 bits
vCPUs	1	16	1	6
RAM (GB)	1,7	60,05	1	15
Armazenamento (GB)	160 (1 HDD)	2,048 (2 SSD)	40 (1 HDD)	620 (1 HDD)
Preço (dólar/hora)	0,06	3,10	0,06	0,90
ECU	1	35	*	*
Virtualização	Xen 64-bit	Xen 64-bit	Xen 64-bit	Xen 64-bit

* Não disponível

Fonte: Autoria própria.

Tipos de Instâncias

Amazon EC2 e Rackspace oferecem uma variedade de tipos de instâncias, tornando difícil a comparação das instâncias entre os provedores. Dentre todos os tipos de instâncias disponíveis, foram escolhidos dois grupos distintos: *small* e *large*. O grupo *small* consiste em instância com hardware simples (1 virtual CPU ou vCPU), com uma pequena capacidade de disco de armazenamento e tem o mesmo preço entre os provedores. O grupo *large* contém a segunda melhor instância de cada provedor, com diferentes vCPU, armazenamentos e preços.

Amazon EC2 oferece muitos tipos de instâncias, de propósito geral até otimizada para armazenamento. A instância M1 Small (m1.small, nomeada ec2.small neste trabalho) tem a seguinte configuração: 1,7 GB de memória, 160 GB de disco local e 1 ECU (EC2 Compute Unit) de processamento. A Amazon WS

informa que um processador com 1 ECU de poder computacional é equivalente a capacidade de CPU de um processador Opteron de 1,0 a 1,2 GHz ou um processador Xeon 2007. Esse tipo de instância custa U$ 0,06 dólares por hora.

Para instâncias *small*, o provedor não detalha a arquitetura específica do processador. De acordo com /proc/cpuinfo, ele informa que o processador tem um núcleo físico e é um processador Intel Xeon E5430 @2,66 GHz .

No grupo *large*, a instância High I/O Quadruple Extra Large (hi1.4large, chamada aqui de ec2.large) foi escolhida. Possui dois discos SSD, de 1 TB cada, como armazenamento de disco local. Essa instância é usada para prover alta performance de E/S. Além disso, tem 8 núcleos (16 vCPUs com *hyperthreading*) com 35 ECUs e 60,5 GBytes de memória. A Amazon limita apenas duas instâncias ec2.large simultâneas por usuário, atém de ter um custo mais alto por hora (U$ 3,10).

Rackspace também tem muitas instâncias, que são nomeadas pelo tamanho de memória. O tipo de 1 GB de RAM (chamada neste trabalho de rack.small) é um recurso com 40 GB de armazenamento local com 1 vCPU e 1 GB de memória. Essa é a segunda instância mais simples do Rackspace e custa U$ 0,06 por hora (o mesmo preço da ec2.small).

A outra instância do Rackspace selecionada é chamada de 15 GB de RAM (chama aqui de rack.large). Ela é um recurso com 630 GB de armazenamento local, 6 vCPU e 16 GB de memória. É a segunda melhor instância do Rackspace (custa U$ 0,90 por hora). Rackspace não tem uma métrica de performance de CPU para comparar suas próprias instâncias como a Amazon ECU.

Aspectos de Custo

Amazon oferece diferentes opções de compra: (1) sob-demanda (*on-demand*): o pagamento é feito por hora de uso e a computação é obtida imediatamente; (2) instâncias spot: é feito um leilão de instâncias não usadas pelo Amazon Ec2; (3) instâncias reservadas: o usuário paga uma taxa por um período de tempo e obtém descontos por um ou três anos de uso. O Rackspace tem somente

instâncias sob-demanda, portanto é usada a compra sob demanda neste trabalho, para fins de comparação.

Cada instância será alocada para usuários de maneira não dedicada (muitas Máquinas Virtuais por nó físico). Amazon tem uma adicional opção de alocação dedicada com pagamento por hora, mas o Rackspace tem somente essa alocação com pagamento por mês.

Tipos de Armazenamento

Os provedores selecionados oferecem o mesmo tipo de armazenamento com diferentes nomes. Esse trabalho adota os seguintes tipos de armazenamento descritos na Tabela 5: (1) armazenamento em disco, os dados do usuário não são perdidos quando uma instância é reiniciada e normalmente eles são armazenados em um dispositivo de disco local. Também é chamado de armazenamento não-persistente. O armazenamento local da Amazon é chamado de efêmero e no Rackspace é chamado de disco; (2) armazenamento em bloco é um volume remoto que pode ser ligado a uma instância como dispositivo de armazenamento de bloco. Esse volume é acessado através da rede e os dados são persistentes. É Chamado de Elastic Block Store (EBS) na Amazon e volume no Rackspace; (3) o armazenamento de objetos gerencia os dados como um objeto e provê uma interface de serviço web para acessar os arquivos. Esse armazenamento é acessado pela rede. Simple Storage Service (S3) é o nome comercial da Amazon para armazenamento em objeto e é conhecido como Cloud Files no Rackspace. O armazenamento em objeto não é utilizado neste trabalho.

Tabela 5 - Tipos de Dispositivos de Armazenamento

Tipo de Armazenamento	Modo de Acesso	Nome na Amazon	Nome no Rackspace
Disco	local	Efêmero ou Instance Store	Disco (Disk)
Bloco	rede	EBS	Volume
Objeto	rede	S3	Cloud Files

Fonte: Autoria própria.

A Tabela 6 apresenta os dispositivos de armazenamento selecionados e a Tabela 7 mostra os custos dos armazenamentos. Os preços estão em dólar e foram coletados em Outubro de 2013. O preço do armazenamento é o custo adicional de utilização do dispositivo de armazenagem selecionado. O disco (local) não possui nenhum custo adicional. Amazon permite o armazenamento EBS com no mínimo 1 GB de tamanho e Rackspace com no mínimo 100 GB de tamanho. Esse tamanho mínimo reflete o custo mínimo. Amazon também oferece o provisionamento de Operações de Entrada e Saída por Segundo (*Input/Output Operations Per Second* - IOPS). Este é um armazenamento de alta performance para aplicações de E/S intensiva.

Tabela 6 - Descrição dos armazenamentos

Provedor	Nome do Armazenamento	Descrição	Tipo de Armazenamento
Amazon	ec2.root	EBS Padrão no root device	bloco
	ec2.stand	EBS Padrão (Standard)	bloco
	ec2.iops	1000 IOPS	bloco
	ec2.ssd	disco SSD local	disco
Rackspace	rack.local	disco HDD local	disco
	rack.sata	Volume Padrão	bloco
	rack.ssd	Volume SSD	bloco

Fonte: Autoria própria.

As opções de armazenamento da Amazon são (1) ec2.root, os dados e o sistema operacional são armazenados no mesmo armazenamento de bloco; (2) ec2.stand, os dados estão em um armazenamento de bloco e o sistema operacional em outro armazenamento de bloco; (3) ec2.iops, o provedor assegura o mínimo de desempenho de IOPS para o armazenamento. Podem ser garantido 1,000 IOPS para um volume de 100 GB; e (4) ec2.ssd, essa opção usa o discos SSD para o armazenamento na Amazon. Este disco pode ser usado somente em instâncias ec2.large.

As opções de armazenamento no Rackspace são (5) rack.local, os dados são armazenados no disco local e os dados são efêmeros; (6) rack.sata, o

armazenamento é externo e o disco é SATA; (7) rack.ssd, o armazenamento é externo e usa o dispositivo SSD para armazenamento

Tabela 7 - Custo dos Armazenamentos

Provedor	Nome do Armazenamento	Preço do Armazenamento (dólar/GB/mês)	Preço E/S (dólar)
Amazon	ec2.root	0,100	0,100 *
	ec2.stand	0,100	0,100 *
	ec2.iops	0,125	0,100 **
	ec2.ssd	0,000	0,000
Rackspace	rack.local	0,000	0,000
	rack.sata	0,150	0,000
	rack.ssd	0,700	0,000

* por 1 milhão de requisições de E/S
** por IOPS/mês provisionado

Fonte: Autoria própria.

5.1.2. Configuração dos Experimentos e Metodologia de Avaliação.

A avaliação de performance de E/S dos provedores da Amazon e do Rackspace foi realizada com os tipos de instância ec2.small, ec2.large, rack.small e rack.large (Tabela 4) e com os armazenamentos ec2.root, ec2.stand, ec2.iops, ec2.ssd, rack.local, rack.sata e rack.ssd (Tabela 6).

Nesta avaliação foi usada o *benchmark* IOzone com o sistema de arquivo local (ext3) nos dispositivos de discos efêmeros e de blocos, e os resultados numéricos são a vazão em KBytes/segundos.

Os parâmetros usados no benchmark são -i0 -i2 -o -r -t, onde (a) -i0, significa funcionamento de operações de escrita e leitura no modo sequencial; (b) -i2, executa operações de leitura e escrita no modo randômico; (c) -o, conjunto de operações assíncronas; (d) -r, define o tamanho de gravação (*record*) em KBytes; (e) -s, o tamanho do arquivo; e (f) -t, o número de *thread* ou processos.

O *benchmark* IOzone foi compilado em cada tipo de instância com o compilador GNU C 4.4.7 sem qualquer parâmetro adicional para a melhoria de desempenho.

A avaliação de E/S foi conduzida em dois casos de testes distintos: primeiro, um teste sequencial de escrita seguido pelo teste sequencial de reescrita (opção -i0), e o segundo, um teste de escrita randômica seguido pelo teste de leitura randômica (opção -i2). Todas as operações foram realizadas no modo síncrono. A razão para essa escolha está no fato de o acesso randômico e as operações síncronas serem vastamente utilizados em ambientes de banco de dados.

O tamanho de gravação (record) é de 4 KBytes. Essa é a opção padrão para a maioria dos sistemas operacionais. O tamanho de arquivo escolhido é de 3,4 GBytes para todos os experimentos. Na avaliação de instâncias *small*, a memória da Máquina Virtual é estressada e o arquivo tem ao menos o dobro do tamanho de memória. Como o ec2.small tem mais memória do que o rack.small, então 1,7 GB foi escolhido como base para definir o tamanho do arquivo para a execução do *benchmark*. O mesmo tamanho de arquivo foi utilizado na avaliação de instâncias *large*. O objetivo é saber se a melhor instância realmente melhora o desempenho observado pela aplicação. A memória não foi estressada nas avaliações de instâncias *large*.

O tamanho do armazenamento do bloco foi configurado para 100GB porque esse é o tamanho de bloco mínimo no Rackspace. Essa avaliação utilizou 1000 IOPS, por ser o valor máximo para o tamanho do bloco escolhido na Amazon.

Todos os experimentos da Amazon foram feitos na região us-east-1 (Virgínia do Norte), dentro da zona de disponibilidade east-1a e os experimentos do Rackspace foram feitos na região de Chicago (ORD).

Na análise *multithreading*, o número de threads foi aumentado para avaliar o desempenho do paralelismo nos tipos de instâncias e armazenamento. O número de vCPU em instâncias pequenas é 1 o número de *thread* foram aumentados de 1 para 5, de forma a para verificar se poderia haver uma melhoria desempenho.

O sistema operacional selecionado foi o Centos 6.4 para todos os experimentos. A imagem da MV que foi usada na Amazon foi a oficial Official CentOS 6.4 x86_64 do AWS Marketplace (AMI ID ami-bf5021d6). A imagem usada no Rackspace foi a CentOS 6.4 padrão.

Nessa avaliação, o benchmark IOzone foi executado cinco vezes e o desvio padrão foi calculado nos experimentos com tipos de instâncias small. A memória cache foi limpada antes de executar cada teste.

5.1.3. Resultados

Nesta seção são apresentados os principais resultados dos experimentos realizados. Para detalhes completos, consultar (RODAMILANS; BARUCHI; MIDORIKAWA, 2014).

Instâncias *Small* – Operações de Escrita Randômica

Atualmente, aplicações com *multithreads* estão se tornando cada vez mais comuns. Foram avaliados os acessos paralelos no dispositivos de armazenamento em ambiente de Computação em Nuvem.

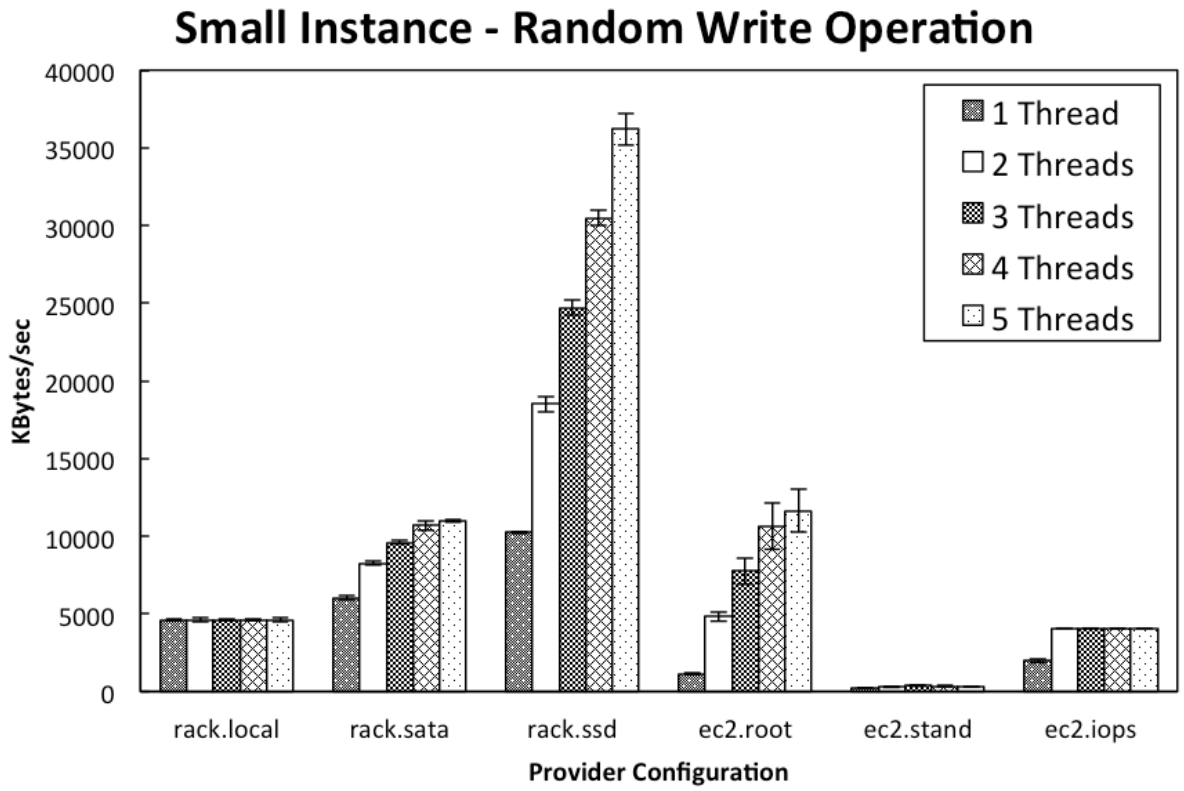
Figura 20 apresenta a largura de banda máxima obtida (medida em KBytes por segundo) para operações de escrita com variação do número de acessos a threads. O desvio padrão é mostrado em cada experimento. Instâncias *small* têm somente uma vCPU na Amazon EC2 e no Rackspace. Nesses experimentos, o número de threads aumenta, variando de 1 a 5, e todas as threads são executadas em paralelo.

Para a operação de escrita randômica, o desempenho do disco do rack.local parece não ter influência da variação do número de threads. A largura de banda observada foi quase a mesma, em torno de 4,600 KBytes/segundo. Na Amazon EC3 com volume IOPS (ec2.iops), o aumento do número de threads também teve pequeno efeito porque o acesso ao volume é limitado a somente um número de IOPS provisionado. Nesses experimentos, Amazon EC2 garante até 1.000 IOPS

para volume externo e o tamanho da página usado para operação de escrita randômica é de 4 KBytes.

ec2.stand teve o pior resultado de desempenho e a variação do número de threads não é capaz de produzir qualquer benefício da performance.

Figura 20 - Instância Small com aumento do número de *threads* para operações randômicas síncronas.



Fonte: (RODAMILANS; BARUCHI; MIDORIKAWA, 2014).

A análise comparativa entre configurações de provedores diferentes mostra que o aumento do número de threads é capaz de obter significativos ganhos de desempenho nos experimentos com rack.sata, ec2.root e rack.ssd, aprimoramento de 82%, 920% e 254%, respectivamente.

O melhor resultado de desempenho para operações de escrita randômica é obtido pelo Rackspace com volume SSD, que obteve uma taxa de transferência de 36.000 KBytes por segundo com 5 threads.

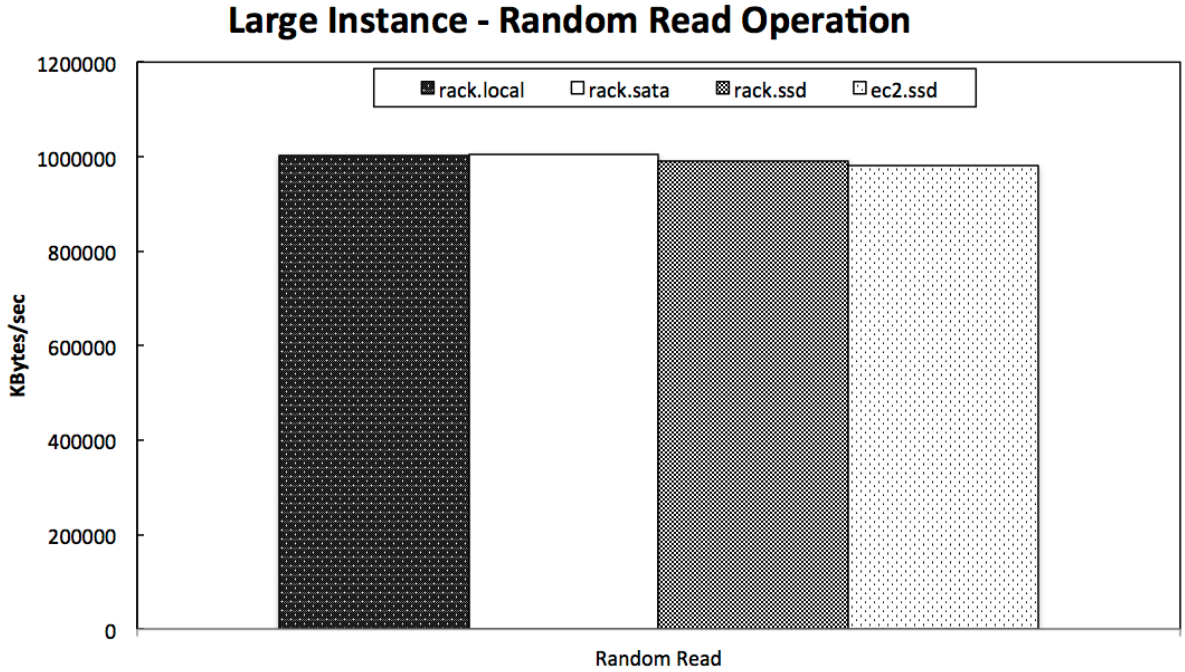
Esses resultados mostram que o Rackspace com volume externo é melhor que o volume local. Esse fato sugere que a concorrência do disco local entre usuários de nuvem é mais significativa para a perda de desempenho em relação ao desempenho de escrita aleatória do que o uso da rede para acessar o volume externo ou *storage* multi camada (SSD, SATA).

O número de threads não pode ser ignorado e é necessário verificar quantas threads podem aumentar o desempenho para cada aplicação especifica.

Instância Large – Leitura Randômica com uma *thread*

A Figura 21 apresenta os as larguras de banda de leitura experimentais para instâncias *large*. Todas as instâncias e armazenamentos avaliados apresentaram resultados similares. O volume SATA Rackspace obteve 1.005.052,69 KBytes/segundos, com 2,5% de melhoria comparada com as outras configurações. Então pode-se concluir que o disco local do Rackspace é a opção mais barata com resultados de largura de banda similares, uma aplicação que apresenta um padrão de acesso de leitura aleatória pode ser alocado em uma Máquina Virtual com disco local no Rackspace.

Figura 21 - Desempenho da instância Large da Amazon EC2 e do Rackspace focado no tipo leitura randômica.



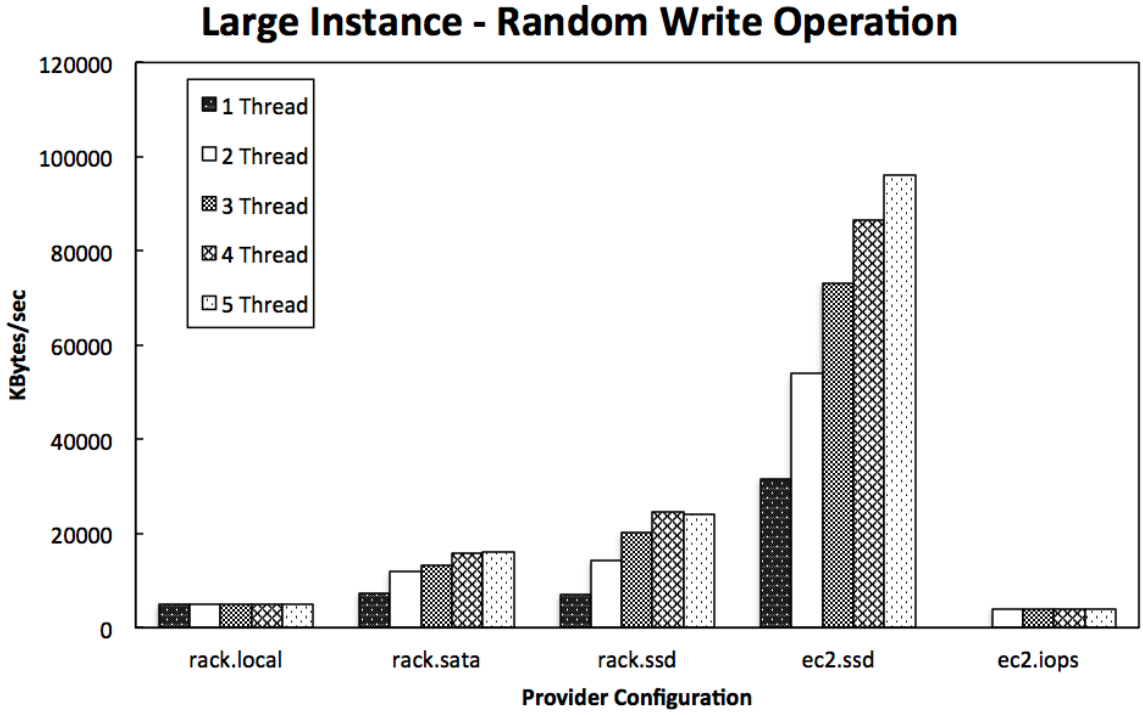
Fonte: (RODAMILANS; BARUCHI; MIDORIKAWA, 2014).

Instância Large – Escrita Randômica com múltiplas threads

Essa seção foi repetida para a análise da influência da aplicação *multithreading* em termos de desempenho do disco. Para efeitos de comparação de instâncias small, limitou-se o número de *threads* para cinco.

A Figura 22 apresenta a largura de banda de escrita randômica obtida com a variação do número de threads em instâncias large. Novamente, o Rackspace com disco local não apresenta melhoria de performance significativa com o aumento do número de threads e obtém 5.094,15 KBytes/segundos. Por outro lado, o disco SSD em ambos provedores apresentam melhorias de performance. As configurações do Rackspace e do Amazon EC2 obtiveram 342% e 304% de melhoria, respectivamente, comparando 1 *thread* com 5 *threads*. Mas Amazon Ec2 com SSD obtém o melhor resultado para 5 threads, com a vazão (*throughput*) de 96.000 KBytes/segundos. Mais uma vez, a Amazon EC2 com IOPS provisionado não melhora a performance como discutido anteriormente em *Instâncias Small – Operações de Escrita Randômica*. O Amazon EC2 com IOPS provisionado com 1 *thread* não é apresentado no gráfico porque sua execução foi abortada.

Figura 22 - Desempenho de instâncias *large* com operações de escrita randômica e variação do número de *threads*.



Fonte: (RODAMILANS; BARUCHI; MIDORIKAWA, 2014).

5.1.4. Análise específica dos resultados

Nesta seção, são apresentadas as análises dos resultados das (a) instâncias small; (b) instâncias large; (c) variação de threads em instâncias small; (d) variação de threads em instâncias large. Uma análise mais detalhada dos resultados está em (RODAMILANS; BARUCHI; MIDORIKAWA, 2014).

Instâncias Small

Essa avaliação sobre instâncias *small* mostrou que o Rackspace com volume local possui melhor desempenho para operações sequenciais de escrita e reescrita. Uma outra vantagem do disco local é que o seu custo é gratuito, enquanto os outros volumes são cobrados. Para operações randômicas de escrita e leitura, a melhor escolha para performance é Rackspace com volume externo SSD. Isso significa que um custo adicional pode ser usado para obter desempenho de E/S extra. Se os custos são um problema, o Rackspace usando disco local é um boa escolha também.

Variação de threads em instâncias Small

Essas avaliações mostraram que o Rackspace com volume SSD externo é a melhor configuração para operações randômicas de leitura e escrita síncrona em instâncias pequenas nesses experimentos. O aumento do número de *threads* pode melhorar significativamente o desempenho nesta configuração. Essas avaliações confirmaram que a rede não é o gargalo no Rackspace para essas operações. Então o uso de volume externo pode ser a melhor escolha para a melhora de desempenho de I/O.

A Amazon EC2 com a configuração de IOPS provisionado é a melhor escolha do que as outras configurações no provedor Amazon. O aumento do número de *threads* não melhora o desempenho quando as operações de E/S já estão na taxa máxima de IOPS.

Instâncias Large

A primeira conclusão que pode ser derivada dessa análise é que o Amazon EC2 com disco SSD local provê melhor performance para operações de escrita,

reescrita e escrita randômica. A segunda conclusão é que operações randômicas têm resultados similares para todas as configurações de dispositivos e a melhor escolha é o Rackspace com disco local porque é a configuração mais barata. A análise de custo revela que o Rackspace com disco local é a melhor opção na maioria dos casos.

Variação de threads em instâncias Large

Essa avaliação mostrou que a Amazon Ec2 com disco SSD local é a melhor escolha de desempenho em operações de leitura e escrita randômica. Esses resultados também confirmam que o Rackspace com disco local e Amazon EC2 com IOPS limitados não aumentam o desempenho com o aumento do número de threads. A melhor escolha de custo/benefício em operação de leitura randômica com instâncias *large* é apresentada pelas instâncias com disco local do Rackspace com 4 threads.

5.1.5. Análise Geral

A Tabela 8 e Tabela 9 apresentam um sumário das análises para o melhor provedor e configurações de armazenamento para o desempenho de E/S com 1 thread e muitas threads, respectivamente. Essas tabelas ajudam a escolher qual o provedor de armazenamento é melhor para operações de E/S randômicas e sequenciais.

Tabela 8 - Melhor provedor e configuração de provedor para desempenho de E/S para 1 *thread*.

Operação		1 *Thread*	
		Small	*Large*
Sequencial	Escrita	Rackspace - disco local	Amazon EC2 - disco SSD
	Reescrita	Rackspace - disco local	Amazon EC2 - disco SSD
Randômica	Leitura	Rackspace - bloco SSD	Amazon EC2 - disco SSD
	Releitura	Rackspace - bloco SSD	Rackspace - disco local *

* desempenho similar, custo mais barato

Fonte: Traduzido de (RODAMILANS; BARUCHI; MIDORIKAWA, 2014).

A análise comparativa entre tipos diferentes de instâncias mostra que o dispositivo de volume externo pode ser melhor que o disco local (Tabela 8). O disco local chegou a ser o gargalo uma vez que é compartilhado por múltiplos usuários e a largura de banda da rede pode prover capacidade suficiente em oposição ao acesso de disco local.

As análises do aumento de número de threads mostram que: (a) instâncias caras não podem ser a melhor escolha em casos de somente uma *thread*, *e.g.* Rackspace com disco local com operações sequenciais de escrita e releitura para instâncias pequenas e leitura randômica para instâncias *large*; (b) Amazon EC2 com volume IOPS provisionado não aumenta muito o desempenho para mais de 2 *threads* por causa da limitação da taxa de IOPS; (c) O aumento do número de *threads* é melhor para desempenho de dispositivo SSD (Tabela 9); Rackspace com volume SSD obtém melhor desempenho com o aumento do números de *threads* para instâncias *small* (embora tenha somente uma vCPU); Amazon EC2 com SSD local como armazenamento de 16 vCPUs obtém melhor desempenho para instâncias usando muitas threads.

Tabela 9 - Melhor provedor e configuração de provedor para desempenho de E/S para várias *threads*.

Operação		Várias *Threads*	
		Small	*Large*
Randômica	Escrita	Rackspace - bloco SSD	Amazon EC2 - disco SSD
	Leitura	Rackspace - bloco SSD	Amazon EC2 - disco SSD

Fonte: Traduzido de (RODAMILANS; BARUCHI; MIDORIKAWA, 2014).

5.1.6. Conclusões

Computação em Nuvem, além de fornecer uma Infraestrutura como Serviço no qual o consumidor “paga pelo que usa”, também oferece alto desempenho de E/S para HPC (*High Performance Computing*) e aplicações Comerciais. Considerando-

se que Amazon e Rackspace são os principais provedores públicos que oferecem infraestrutura como serviço, foram utilizados neste trabalho.

Foram analisados os desempenhos de diferentes E/S de armazenamento e tipos de instâncias nos provedores de nuvem Amazon e Rackspace. Os diferentes tipos de instâncias e de armazenamento foram avaliados em ambos os provedores. O benchmark IOzone foi usado em todos os experimentos com modo de acesso de escrita e leitura sequencial, e modo de acesso de leitura e escrita randômico, usando operações síncronas. O número de threads foi aumentado para avaliação de melhoria de desempenho.

Os resultados do desempenho têm mostrado que os armazenamentos e a variação do número de threads apresentam significativas diferenças nos provedores da Amazon e Rackspace. O trabalho também revelou que o uso de armazenamento de bloco externo pode prover melhor desempenho do que disco local e a rede não é o gargalo de desempenho, *e.g.* o estudo de caso do Rackspace com instância *small* com bloco SSD para operações de leitura e escrita randômica com uma thread. Além disso, foi caracterizado qual provedor de nuvem pode possuir melhor desempenho para cada operação de acesso a disco e essa avaliação revelou que o disco local pode oferecer melhor desempenho e custo mais baratos para instâncias *large*, *e.g.* o estudo de caso do Rackspace com instâncias *large* com disco local para operações randômicas com uma thread. Finalmente, a análise do armazenamento SSD mostrou que o número de threads aumenta significativamente o desempenho do disco SSD na Amazon EC2 e no bloco SSD no Rackspace.

Em geral, as respostas para as questões são: (a) nem sempre a instância mais cara apresenta alto desempenho, e (b) o desempenho de instâncias *large* deve ser considerado caso-por-caso dependendo do padrão de uso de recurso da aplicação.

O próximo passo dessa pesquisa é analisar o desempenho de uma aplicação real (*e.g.* openModeller (MUÑOZ *et al.*, 2011), uma aplicação de análise de biodiversidade).

5.2. APLICAÇÃO DA METODOLOGIA CSE - OPENMODELLER

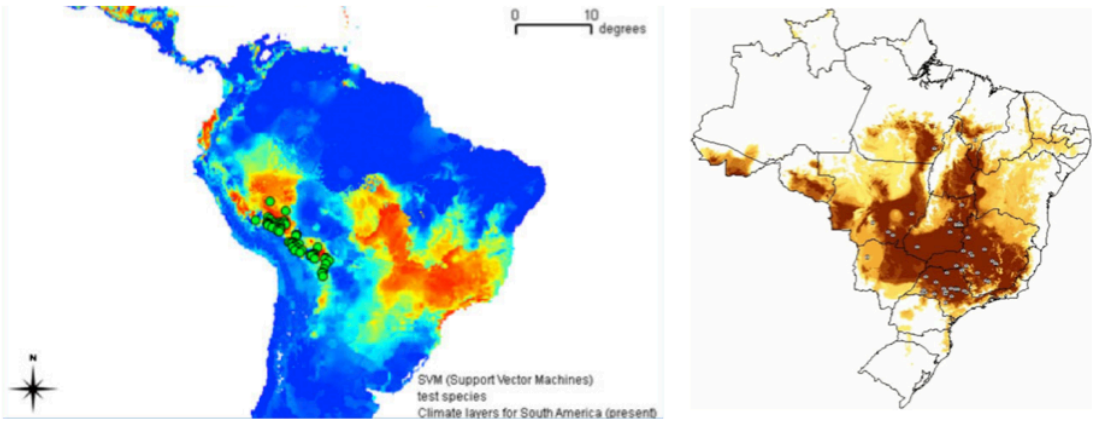
Esta seção apresenta a aplicação da metodologia CSE. Na primeira parte, é apresentada a descrição da aplicação utilizada, o OpenModeller. A segunda parte descreve o ambiente de testes. Na terceira, aplica-se a metodologia. Na quarta e última parte, realiza-se a análise dos resultados.

5.2.1. Descrição da Aplicação

A modelagem de distribuição potencial da espécie permite predizer a distribuição de espécies a partir de informações quanto à ocorrência da espécie de uma região e informações geográficas, tais como clima, solo, altitude (MUÑOZ *et al.*, 2011). A análise de distribuição da espécie, por sua vez, permite detectar problemas de saúde pública causados por vetores de doenças, extinção de espécies, assim como, planejar novas áreas de conservação.

Diversos algoritmos podem ser utilizados para projetar a evolução de espécies, tais como GARP, Maximum Entropy (Maxent), Support Vector Machines (SVM). Porém, para cada algoritmo, o usuário necessita aprender o seu funcionamento e também estudar o software de utilização do algoritmo. OpenModeller (MUÑOZ *et al.*, 2011) é um framework que provê um ambiente para a realização de experimentos de modelagem de nicho ecológico, oferecendo um único software para a utilização de diversos algoritmos de modelagem. A Figura 23 apresenta resultados de distribuição de espécies utilizando o OpenModeller.

Figura 23 - Resultados de experimentos de distribuição de espécies utilizando o OpenModeller.



Fonte: (MUÑOZ *et al.*, 2011).

O OpenModeller é um projeto desenvolvido pela Escola Politécnica da Universidade de São Paulo (Poli/USP) (POLI/USP, 2014), Instituto Nacional de Pesquisas Espaciais (INPE) (INPE, 2014), Centro de Referência em Informação Ambiental (CRIA) (CRIA, 2014) e outras instituições colaboradoras.

5.2.2. Descrição do Ambiente de Testes

Os testes experimentais foram conduzidos utilizando-se infraestruturas de nuvens públicas e privadas. As nuvens públicas utilizadas foram Amazon EC2 e Rackspace, suas configurações foram descritas na 5.1.1.

As nuvens privadas utilizadas foram Nuvem USP e Nuvem LAHPC. A Nuvem USP (NUVEMUSP, 2014) é uma nuvem desenvolvida para a comunidade da Universidade de São Paulo (USP).

O Nuvem LAHPC é um ambiente de testes do laboratório de pesquisa de Arquitetura e Computação de Alto Desempenho (*Laboratory of Architecture and High Performance Computing* - LAHPC) da Escola Politécnica da Universidade de São Paulo (POLI/USP). Para os testes, foi utilizada uma das máquinas reais do ambiente de nuvem para a criação das Máquinas Virtuais, conforme descrito na Tabela 10. A arquitetura da máquina real é 64 bits, com processador de Intel Xeon E5-2620 de 2 GHZ, memória RAM de 32 GB. Esta máquina é composta por 3 dispositivos de armazenamento: 2 Unidades de Disco Rígido (*Hard Disk Drive* - HDD) com 1 TB de capacidade (em cada unidade) e interface SATA III; 1 Unidade de Estado Sólido (*Solid-State Drive* - SSD) com 120 GB de capacidade e interface SATA III. O sistema operacional utilizado foi o Linux Centos 6.4.

Tabela 10 - Descrição da máquina real utilizada na Nuvem LAHPC.

Característica	Descrição
Arquitetura	64 bits
Processador	Intel Xeon E5-2620 de 2 GHz (6 núcleos - 12 *hyperthreads*)
Memória	32 GB
Armazenamento	2 HDD de 1 TB (cada) com SATA III 1 SSD de 120GB com SATA III
Virtualização	KVM 64-bit

Fonte: Autoria própria.

Tipos de Instâncias

Nos experimentos da aplicação da metodologia CSE, a descrição das instâncias utilizadas em cada provedor de nuvem é apresentada na Tabela 11. Os tipos de instâncias utilizados na Amazon EC2 e no Rackspace foram ec2.small e rack.small. Estes foram descritos em detalhes na seção na seção 5.1.1.

Tabela 11 – Descrição das instâncias utilizadas no testes da aplicação do CSE.

Provedor	Amazon EC2	Rackspace	Nuvem USP	Nuvem LAHPC
Tipo de Instância	ec2	rack	usp	lahpc
Nome da API	m1.small	1GB RAM	VM Web Linux	small
Arquitetura	64 bits	64 bits	64 bits	64 bits
vCPUs	1	1	1	1
RAM (GB)	1,7	1	1	1
Armazenamento (GB)	160	40	20	20
Preço (dólar/hora)	0,06	0,06	0,00 *	0,00
Virtualização	Xen 64-bit	Xen 64-bit	Xen 64-bit	KVM 64-bit

* Possui um valor simbólico de R$ 0,07. Entretanto, não está sendo comercializada.

Fonte: Autoria própria.

No provedor Nuvem USP, foi utilizada a instância denominada VM Web Linux (chamada neste trabalho de usp), por ser a instância mais simples deste provedor e por ter configurações semelhantes às das instâncias ec2 e rack. A instância usp possui a arquitetura de 64 bits, com 1 vCPU e 1 GB de memória RAM e 20 GB de capacidade de armazenamento efêmero. Possui um valor simbólico de R$ 0,07 que não está sendo cobrado até o presente momento (2014). Acredita-se que com o aumento da demanda, possam surgir cotas de utilização e a cobrança do valor seja efetuada. Entretanto, as instâncias desta nuvem privada não estão sendo comercializadas. A virtualização utilizada é o Xen. O sistema operacional utilizado foi o Linux Centos 6.3.

Na Nuvem LAHPC, utilizou-se a instância chamada small (denominada neste trabalho de lahpc), que possui a arquitetura de 64 bits. Possui 1 vCPU e 1 GB de RAM, semelhantes às configurações das demais instâncias. Não possui custo e

utiliza o Monitor de Máquinas Virtuais (*hypervisor*) KVM de 64 bit. O sistema operacional utilizado foi o Centos 6.4.

Nas instâncias Amazon Ec2, Rackspace e Nuvem USP usou-se o Linux Centos 6.4 de 64-bit e na Nuvem LAHPC utilizou-se Linux Centos 6.2 de 64-bit. Na Amazon Ec2, foram utilizadas as seguintes características: identificação da imagem ami-bf5021d6, a região foi Leste EUA (Virgínia Norte). No Rackspace, foi utilizada a região de Chicago. A Nuvem USP e Nuvem LAHPC não possuem diferentes tipos de regiões para serem selecionadas.

Descrição dos Armazenamentos

A descrição dos armazenamentos utilizados nestes experimentos é apresentada na Tabela 12.

Nos provedores de nuvem pública (Amazon e Rackspace), o nome do armazenamento e descrição dos provedores são semelhantes aos da Tabela 6. As configurações das instâncias utilizadas são as mesmas, entretanto, mudou-se o nome do armazenamento de rack.local para rack.root e rack.sata para rack.hdd) e acrescentou-se o local de instalação do sistema operacional para facilitar a comparação com as nuvens privadas.

Tabela 12 - Descrição dos armazenamentos

Tipo de Nuvem	Provedor	Nome do Armazenamento	Descrição	Tipo de Armazenamento
Pública	Amazon	ec2.root	SO no EBS padrão	bloco
		ec2.stand	EBS Padrão (Standard)	bloco
		ec2.iops	1000 IOPS	bloco
	Rackspace	rack.root	SO no Disco HDD local	local
		rack.hdd	Volume Padrão	bloco
		rack.ssd	Volume SSD	bloco
Privada	Nuvem LAHPC	lahpc.root	SO no Disco HDD local	local
		lahpc.hdd	Disco HDD local	local
		lahpc.ssd	Disco SSD local	local
	Nuvem USP	usp.root	SO no Disco HDD local	local

Fonte: Autoria própria.

Os nome de armazenamento terminado com .root significa que o Sistema Operacional foi instalado no próprio sistema de armazenamento onde serão realizados os experimentos. O rack.root, laphc.root e usp.root possuem o tipo de armazenamento efêmero (local), enquanto o ec2.root utiliza o persistente (bloco).

O ec2.stand e ec2.iops são blocos anexados (*attached*) ao ec2.root (que possui o sistema operacional). Estes blocos são persistentes e se comunicam com ec2.root via rede. O ec2.stand oferece o armazenamento persistente padrão para bloco da Amazon (*Elastic Block Store* - EBS). O ec2.iops também é um EBS, mas oferece qualidade de serviço (QoS), garantindo um número de Operações de Entrada/Saída por segundo (IOPS). Nestes experimentos foram utilizados 1000 IOPS.

O rack.hdd e o rack.ssd são blocos anexados (*attached*) ao rack.root. Ambos são persistentes, sendo que o primeiro utiliza a tecnologia HDD e o segundo a tecnologia SSD.

Na Nuvem LAHPC, foram anexados dois armazenamentos locais ao lahpc.root: um utilizando a tecnologia HDD (lahpc.hdd) e o outro com a tecnologia SSD (lahpc.ssd).

Custo

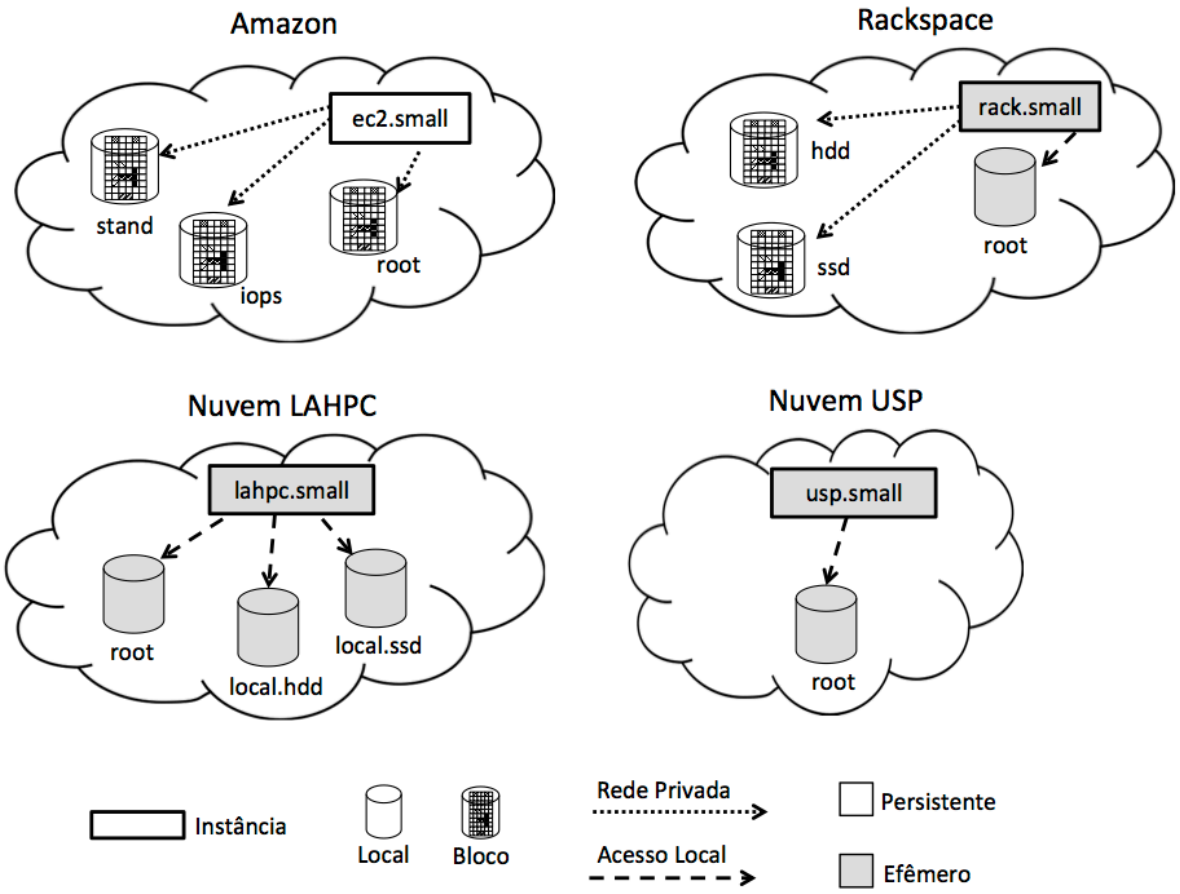
Os custos de armazenamento dos provedores públicos (Amazon e Rackspace) são detalhados na Tabela 7. Ressaltando, mais uma vez, que o nome de armazenamento rack.local foi mudado para rack.root e rack.sata para rack.hdd nos experimentos desta seção. Os provedores privados utilizados (Nuvem LAHPC e Nuvem USP) não possuem custos para o usuário.

Configuração do Ambiente

A configuração do ambiente para a realização dos experimentos com a metodologia CSE é apresentada na Figura 24. Na Amazon Ec2, as instância ec2.small é armazenada de maneira persistente no tipo de bloco denominado root. Os outros blocos denominados stand e iops são anexados na instância. Os dados dos

experimentos são armazenados em cada um dos blocos, para em seguida serem executados. O acesso aos blocos é realizado utilizando a rede interna da Amazon.

Figura 24 - Configuração do ambiente dos experimentos do CSE.



Fonte: Autoria própria.

No Rackspace, a instância efêmera é armazenada localmente em root. Os blocos com a tecnologia hdd e ssd são anexados na instância. Para acessar tais dados blocos, utiliza-se a rede privada e para o disco local utiliza-se o acesso local.

Na Nuvem LAHPC, a instância e os armazenamentos são efêmeros. A instância é armazenada em um disco local root. Outros discos locais foram anexados à instância com as tecnologias HDD (local.hdd) e SSD (local.ssd). O acesso a todos os disco é realizado utilizando-se o acesso local.

Por fim, a Nuvem USP possui uma instância efêmera que é armazenada localmente em root.

5.2.3. Metodologia CSE

Nesta seção, apresenta-se a execução dos procedimentos da Metodologia Categorização, Seleção e Execução (CSE), conforme apresentado na seção 4.4, para a sua validação. Utilizou-se a aplicação OpenModeller com o algoritmo Bioclim e o parâmetro padrão de corte de desvio padrão de 0,674.

5.2.3.1. Caracterização

Os procedimentos para a caracterização são apresentados abaixo.

Etapa 1. Caracterização da Aplicação

Esta seção apresenta as análises da caracterização da aplicação para o OpenModeller .

Etapa 1.1. Identificação das características do recurso

Nesta etapa, deve ser definido um ou mais recursos computacionais a serem analisados. Para este estudo de caso, foi selecionado o recurso de disco.

Nessa etapa, é necessário ter conhecimentos específicos sobre o recurso, a fim de poder definir quais são as características mais importantes que causam impacto no desempenho (fator de escolha selecionado). As operações de Entrada/Saída (E/S) são analisadas. As principais características de análise da aplicação para a performance de E/S são: (a) Operação de Escrita Assíncrona x Síncrona; (b) Buffered x Direct I/O e (c) Acesso Sequencial x Acesso Randômico.

Etapa 1.2. Seleção das chamadas de sistemas

Nessa etapa, analisam-se quais são as chamadas de sistemas correspondentes com as principais características (de desempenho) identificadas anteriormente. Para analisar as chamadas de sistemas pode-se utilizar o programa *strace*.

a) Operação de Escrita Assíncrona x Síncrona

Através da chamada de sistema *open*, verifica-se se a chamada é síncrona. Por padrão, a operação é assíncrona. Caso tenha o parâmetro O_SYNC, significa que a operação é síncrona no Linux.

b) Buffered x Direct I/O

No Linux, verifica-se na chamada de sistema *open* se existe a opção da diretiva de DIRECT I/O, denominada O_DIRECT.

c) Acesso Sequencial x Acesso Randômico

A operação de leitura randômica pode ser feita analisando-se as chamadas de sistemas de leitura (*read*), escrita (*write*) e *lseek*. A chamada *lseek* é responsável por alterar a posição relativa (*offset*), permitindo um reposicionamento de onde serão realizadas as operações no arquivo.

Na caracterização de Acesso Randômico analisa-se o *lseek* em conjunto com as operações de *read* ou *write*. Como exemplo, se na analise das chamadas de sistemas forem encontradas operações *lseek* seguidas por operações de escrita ou leitura, significa que o acesso está sendo randômico.

Etapa 1.3. Obtenção do comportamento da aplicação

Verificou-se, através das chamadas de sistemas, que o openModeller possui operações do tipo assíncrona, “bufferizada”, com acesso sequencial. As leituras ocorrem no início da aplicação e as escritas no final da aplicação.

Etapa 1.4. Definição das métricas

Essas métricas determinam se as operações são mais de escrita e leitura, permitindo uma melhor seleção do recurso posteriormente. As métricas selecionadas e utilizadas foram:

- Tamanho total de bytes (escritos e lidos)
- Tempo total de execução
- Tempo total de execução das operações (de escrita e de leitura)
- Total de operações (de escrita e leitura)
- Vazão (*Throughput*) de operações de escrita e leitura
    - *e.g.* Operações de escrita / Tempo Total de execução

Para a obtenção dessas métricas foram utilizados os programas time e SystemTap (stap), conforme apresentado em Tabela 13.

Tabela 13 - Métricas de E/S de disco e programas para obtê-las.

Métrica	Operação	Programa (Linux)
Tamanho	Leitura (KB)	stap iostatic.stp
	Escrita (KB)	stap iostatic.stp
Tempo	Execução (s)	time
	Leitura (s)	stap iotime.stp
	Escrita (s)	stap iotime.stp
Total de Operações	Leitura	stap iostatic.stp
	Escrita	stap iostatic.stp
Vazão	Leitura	tempo de Execução/tempo leitura
	Escrita	tempo de Execução/tempo escrita

Fonte: Autoria própria.

Etapa 1.5. Variação da carga de trabalho da aplicação (*workload*)

Esta parte necessita de conhecimentos específicos sobre a utilização da aplicação por parte do especialista. É importante compreender quais são as entradas necessárias para, em seguida, alterá-las e analisar o seu impacto.

Nesta etapa, é analisado o comportamento da aplicação variando-se a carga de trabalho (*workload*). As métricas definidas anteriormente também são analisadas.

Utilizou-se o algoritmo bioclim do OpenModeller. O OpenModeller possui alguns tipos de camadas para a análise de ocorrência de uma espécie em um determinado local:

a. Informações sobre a localização da espécie;
b. Região geográfica a ser analisada;
c. Fatores geográficos (tais como clima, vegetação, solo) a serem analisados.

No OpenModeller o aumento das informações a serem analisadas implica maior processamento e maior operações de E/S. Sendo assim, para a análise da variação de carga do openModeller, variou-se a quantidade de informações a serem analisadas, aumentando assim as camadas. O aumento no número de camadas implica o aumento da quantidade de dados a serem lidos. Utilizou-se 1, 4 e 22 camadas relacionadas à região e fatores geográficos a serem analisados.

A Tabela 14 mostra as métricas analisadas, sendo elas tamanho das camadas (arquivos de entradas), tamanho total da leitura e escrita realizadas. Com 1 camada, o total de operações de leitura é de 67.853 enquanto o de escrita é de apenas 492, ou seja, têm-se 137 vezes mais operações de leitura do que de escrita. O aumento de camadas implica o aumento da leitura, tendo aproximadamente 4000 vezes mais leitura do que escrita. Observando-se os tamanhos dos arquivos lidos e escritos, o tempo e a vazão, confirma-se que o Bioclim é de leitura.

Tabela 14 - Resultados da variação da carga de trabalho com o OpenModeller.

Métrica	Operação	1 Camada (102 MB)	4 Camadas (603 MB)	22 Camadas (2.458 MB)
Tamanho	Leitura (KB)	371.604	2.565.940	10.892.506
	Escrita (KB)	61	78	107
Tempo	Execução (s)	5,930	68,360	128,270
	Leitura (s)	1,181	7,538	26,525
	Escrita (s)	0,002	0,002	0,019
Total de Operações	Leitura	67.853	493.419	2.135.358
	Escrita	492	498	534
Vazão	Leitura	11.442,327	7.217,949	11.715,356
	Escrita	0,012	0,137	0,341

Fonte: Autoria própria.

Na análise da escrita, nota-se que o tamanho total da escrita continua pequeno, com o incremento das camadas. Foram escritos em arquivos 107 KB utilizando 22 camadas (tamanho 2.458 MB). Essa característica indica que o dimensionamento do tamanho total do disco é determinado pelas camadas de entrada.

Etapa 1.6. Definição do perfil da aplicação

As análises do Bioclim do OpenModeller demonstraram que o perfil da aplicação é assíncrono, "bufferizado", sequencial e com mais leitura do que escrita, conforme Tabela 15. Esse perfil servirá de base para a escolha das instâncias de Máquinas Virtuais.

Tabela 15 - Perfil do algoritmo BioClim do OpenModeller.

PERFIL BIOCLIM	
Característica	Descrição
Tipo de Escrita	Assíncrona
Buffer cache	Sim
Tipo de Acesso	Sequencial
Operação Predominante	Leitura

Fonte: Autoria própria.

Etapa 2. Caracterização da Instância

Nesta etapa, são analisadas as instâncias para a criação de seu perfil.

Etapa 2.1. Escolha dos provedores de Nuvem

Os provedores de Nuvens públicas escolhidos foram a **Amazon Ec2** e o **Rackspace** devido à sua popularidade e tempo de mercado. A Amazon foi pioneira em fornecer Infraestrutura como Serviço (IaaS) no ramo de Computação em Nuvem.

Os provedores privados selecionados foram a **Nuvem USP** e o **Nuvem LAHPC**. A nuvem USP é uma iniciativa da Universidade de São Paulo (USP) para

fornecer instâncias de Máquinas Virtuais para alunos, professores e pesquisadores. A Nuvem LAHPC (Cloud LAHPC) é um ambiente de Nuvem criado no Laboratório de Arquitetura e Computação de Alto Desempenho (LAHPC) do Departamento de Engenharia da Computação e Sistemas Digitais (PCS) da USP para a realização de testes. Estes provedores foram selecionados por não terem custo de utilização.

Etapa 2.2. Escolha dos tipos de instâncias

O objetivo desta análise é saber qual provedor fornece o melhor desempenho, considerando a E/S de disco.

Sendo assim, utilizaram-se as instâncias com pequeno poder computacional, denominadas small. A Tabela 11 apresenta a descrição das instâncias utilizadas em cada provedor de Nuvem.

Diferentes formas de armazenamento também podem ser selecionadas. O armazenamento por ser local ou remoto e possui diferentes tecnologias, tais como HDD e SSD.

Os tipos de instâncias selecionadas no Rackspace foram:

- rack.root: disco local, na própria instância;
- rack.hdd: volume remoto em hdd;
- rack.ssd: volume remoto em ssd.

Os tipos de instâncias selecionadas na Amazon EC2 foram:

- ec2.root: armazenamento remoto em hdd;
- ec2.stand: volume remoto padrão em hdd;
- ec2.iops: volume remoto com IOPS em hdd.

Os tipos de instâncias selecionadas na Nuvem LAHPC foram:

- lahpc.root: disco local em hdd;
- lahpc.hdd: volume local em hdd;
- lahpc.ssd: volume local em ssd.

O tipo de instância selecionada na Nuvem USP foi:

- usp.root: disco local em hd.

Etapa 2.3. Definição das métricas

Embora as caracterizações de aplicação e de instâncias possam ocorrer paralelamente, neste estudo de caso utilizou-se o perfil das instâncias para definir quais seriam a operações de E/S a serem utilizadas, que foram assíncronas, "bufferizadas" e sequenciais. A métrica selecionada foi vazão de leitura e escrita.

Etapa 2.4. Seleção de método de avaliação

Para a avaliação de desempenho do disco, selecionou-se o uso de micro-benchmark. O benchmark de I/O de disco escolhido é o IOzone, com operações assíncronas, "bufferizadas" e sequenciais.

Etapa 2.5. Execução do método de avaliação

Nesta etapa, executou-se o micro-benchmark IOzone nas instâncias previamente selecionadas 5 vezes, obtendo-se média. Os resultados de vazão de escrita, reescrita, leitura e releitura obtidos nessa etapa para cada tipo de instância são apresentados na Tabela 16. O desvio padrão é apresentado em Figura 25.

Tabela 16 - Resultados do IOzone em KBytes/sec.

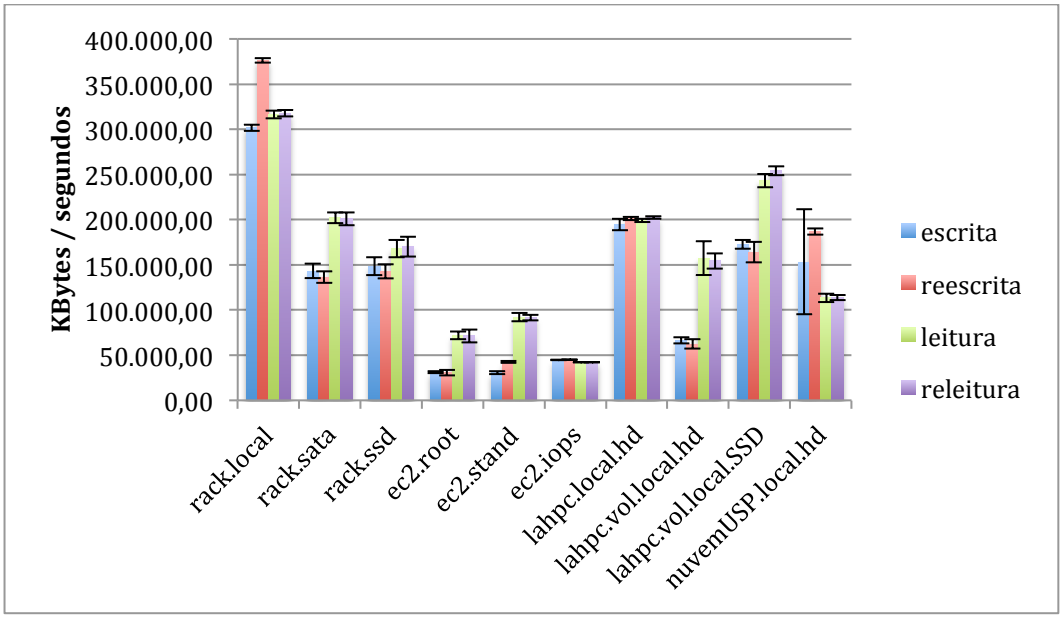
Tipos de Instâncias	escrita	reescrita	leitura	releitura
rack.root	301.837,77	376.383,46	316.296,75	317.882,77
rack.hdd	143.180,45	136.356,26	201.990,85	200.791,76
rack.ssd	148.481,24	142.893,01	167.931,88	170.178,41
ec2.root	31.274,04	30.624,65	71.842,90	71.246,24
ec2.stand	30.748,18	42.745,36	92.128,75	91.724,35
ec2.iops	44.609,98	44.824,11	41.893,41	41.902,11
lahpc.root	194.765,38	201.311,90	199.133,29	202.238,90
lahpc.hdd	66.434,30	62.364,93	157.394,00	154.325,66
lahpc.ssd	172.842,40	164.117,27	243.167,48	254.113,60
usp.root	153.135,53	186.872,68	113.355,42	113.848,71

Fonte: Autoria própria.

Etapa 2.6. Definição do perfil do tipo de instância

Os perfis dos tipos de instâncias em relação ao disco são apresentados na Figura 25. A principal característica de cada instância: (a) rack.local: reescrita; (b) rack.hdd: leitura e releitura; (c) rack.ssd: leitura e releitura; (d) ec2.root: leitura e releitura; (e) ec2.stand: leitura e releitura; (f) ec2.iops: possui desempenho similar para as operações; (g) lahpc.root: possui desempenho similar para as operações; (h) lahpc.hdd: leitura e releitura; (i) lahpc.ssd: leitura e releitura; (j) usp.root: escrita e reescrita.

Figura 25 - Caracterização das Instâncias.



Fonte: Autoria própria.

5.2.3.2. Seleção

Etapa 1. Escolha da aplicação

Nessa etapa, seleciona-se uma aplicação da base de perfil de aplicações. A aplicação escolhida é a OpenModeller com o algoritmo BioClim de 22 camadas e parâmetros de entrada padrão. Vale ressaltar que este estudo de caso está caracterizando apenas uma aplicação, entretanto, várias aplicações podem ser

caracterizadas simultaneamente, sendo necessário selecionar qual aplicação será utilizada primeiramente.

Etapa 2. Escolha do recurso computacional preponderante

Nesta etapa seleciona-se o recurso computacional que obteve um maior impacto com relação a aplicação. Devido a realização somente da análise de um recurso, este recurso foi o selecionado. O recurso escolhido é o disco.

Etapa 3. Definição de métricas e fator de escolha

A métrica selecionada é a vazão (*throughput*) de operações de leitura, devido ao perfil da aplicação. O fator de escolha é o de desempenho de E/S de disco.

Etapa 4. Seleção da instância

Nesse estudo de caso, optou-se pelo algoritmo mais simples e selecionou-se a instância que possui o melhor desempenho de operações de leitura. De acordo com a Tabela 16 é a instância rack.root.

5.2.3.3. Execução

Etapa 1. Escolha da imagem

No Rackspace não havia nenhuma imagem pronta para execução. Selecionou-se então a instância referente ao rack.local (de 1 GB de RAM).

Etapa 2. Criação da instância

Criou-se a instância rack.root.

Etapa 3. Configuração da instância

Foram instaladas as dependências do OpenModeller.

Etapa 4. Execução da aplicação

Executou-se a aplicação 5 vezes. O tempo de execução da aplicação no rack.root foi de 240,87 segundos com desvio padrão de 30,12.

5.2.4. Análise dos Resultados

Executou-se a aplicação nas outras instâncias. A instância rack.root ficou em segunda posição em tempo de execução (240,87 segundos e desvio padrão de 30,12), fincando atrás somente do lahpc.hdd (228,42 segundos e desvio padrão de 2,78), conforme a Tabela 17. Analisando-se o desvio padrão, tanto o rack.local quanto lahpc.hdd possuem um grau de similaridade entre si. Sendo assim, a seleção da instância rack.root foi satisfatória.

Tabela 17 - Execução do OpenModeller em diversas instâncias.

Tipo de Instância	Média (s)	Desvio
rack.root	240,87	30,12
rack.hdd	1.639,62	108,65
rack.ssd	441,74	45,22
ec2.root	1.023,35	384,37
ec2.stand	1.248,82	507,89
ec2.iops	483,43	10,95
lahpc.root	754,26	9,28
lahpc.hdd	228,42	2,78
lahpc.ssd	1.005,68	10,74
usp.root	585,66	23,86

Fonte: Autoria própria.

6. CONCLUSÃO

Diversos provedores de Computação em Nuvem fornecem o serviço de instâncias. Cada provedor possui os seus tipos de instâncias e diferentes configurações, o que amplia a quantidade e variedade de ofertas de instâncias. Por um lado, isto fornece uma maior possibilidade de encontrar a instância mais adequada para uma determinada aplicação. Todavia, essa grande quantidade de instâncias dificulta a escolha e, além disso, cada provedor disponibiliza suas instâncias utilizando padrões próprios. Sendo assim, é necessário caracterizar tais instâncias para que se possa fazer uma análise do perfil de cada uma. Também é necessário caracterizar as aplicações para tornar possível a seleção da instância mais adequada para determinadas aplicações.

Este trabalho apresentou um estudo detalhado sobre Computação em Nuvem e suas características, bem como as diferenças entre Computação em Nuvem e em Grade. Também foram mostrados o modelo de negócio, as características de escalonamento em Nuvem, dentre outros fatores que influenciam a escolha da instância mais conveniente. Ainda, foram apresentados os provedores de Nuvem e a variedade de oferta de suas instâncias.

A pesquisa demonstrou a importância da caracterização das instâncias e das aplicações. Examinou-se que a caracterização forneceu qualidade aos atributos que foram posteriormente utilizados na realização da seleção da instância mais adequada para uma determinada aplicação. Dessa forma, verificou-se a necessidade de caracterizar de maneira efetiva tantos as instâncias do provedores de Nuvem quanto as aplicações.

Esta obra também revelou os procedimentos necessários para determinação do recurso preponderante através da categorização, assim como os procedimentos necessários para a seleção da instância mais adequada para uma determinada aplicação. Além disso, apresentou os métodos essenciais para a execução da aplicação, considerando a possibilidade de existência prévia de uma imagem com a configuração dos requisitos da aplicação.

Foi apresentada, também, a Metodologia de Caracterização, Seleção e Execução (CSE). Esta metodologia visa fornecer as diretrizes para o processo de caracterização de aplicações e instâncias, levando em consideração seus atributos, seleção da instância mais adequada para a aplicação e, por fim, execução da aplicação na instância selecionada. Cada da etapa foi minunciosamente detalhada.

Para viabilizar a interoperabilidade e interligação das Nuvens, apresentou-se uma arquitetura e uma implementação das suas principais funcionalidades desta arquitetura. Esta interligação ampliou a quantidade de instâncias que um sistema pode utilizar, além de permitir a utilização de nuvens públicas e privadas.

As contribuições deste trabalho incluem (a) desenvolvimento da metodologia CSE; (b) demonstração da importância da caracterização e do recurso preponderante da aplicação; (c) avaliação de desempenho do disco em diversas Nuvens; (d) caracterização e avaliação de desempenho do OpenModeller; e (e) arquitetura de interligação de nuvens públicas e privadas, bem como implementação de suas principais funcionalidades.

Nota-se que a contribuição mais notável foi o desenvolvimento da Metodologia Categorização, Seleção e Execução (CSE). Os procedimentos apresentados nessa metodologia são de suma importância para a análise das diversas nuvens. Ela fornece as diretrizes para a categorização, seleção da instância mais adequada para sua aplicação, e execução da aplicação.

Outra contribuição deste trabalho foi a criação, demonstração e validação dos procedimentos relacionados à caracterização das instâncias e das aplicações, permitindo uma posterior seleção da instância adequada para um determinada aplicação. A caracterização fornece informações apuradas, permitindo que se faça uma seleção mais criteriosa das instâncias, e assim, por exemplo, uma melhora no desempenho da execução da aplicação.

A utilização da metodologia CSE para a caracterização das instâncias criou um perfil para diversos tipos de instâncias, de diferentes provedores de Nuvens, apresentando qual a operação de disco (escrita, reescrita, leitura e releitura) é mais eficaz em cada uma delas. Além disso, facilitou a comparação entre elas. Nesse

ponto, importante observar que a caracterização realizada também pode ser utilizada em outros trabalhos relativos a desempenho.

A caracterização do *framework* OpenModeller, utilizando os procedimentos da Metodologia CSE, forneceu o perfil da aplicação relacionada ao disco. Os procedimentos da metodologia determinaram as características do algoritmo BioClim, sendo elas: escrita assíncrona, utilização de *buffer cache*, acesso sequencial e operação predominante de leitura. Essas características permitem uma seleção de instâncias mais adequadas para o algoritmo.

Outra contribuição a ser mencionada refere-se à arquitetura de interligação de nuvens públicas e privadas, bem como a implementação de suas principais funcionalidades. Esta proposta apresentou a arquitetura e os seus principais componentes para a implementação de um sistema que permite a interoperabilidade entre diversos provedores de nuvem. A implementação das principais funcionalidades serve como prova de conceito para a interligação entre os diversos provedores de nuvens públicas e privadas.

A metodologia proposta pode ser utilizada por interessados em avaliar as diversas instâncias, de diversos provedores de nuvem ou em um único provedor. Como exemplo, uma nuvem privada de uma instituição fornece tipos de instâncias variados. A utilização da metodologia para categorizar a instância permite ao usuário da nuvem selecionar qual é mais adequada para sua aplicação. Uma outra vantagem a ser explorada é quanto ao provedor de nuvem informar as características principais das suas instâncias, ou seja, o perfil delas, e quais são os melhores tipos de aplicação para cada instância, facilitando a escolha dos usuários sem que estes precisem fazer a caracterização.

A caracterização da aplicação e a posterior divulgação de suas características permite que os pesquisadores direcionem o seu tempo ao que é mais significante para eles, que são os resultados que elas geram. A caracterização das instâncias permite a criação de uma base de dados global, em que diversos usuários possam utilizar e acrescentar os resultados das caracterizações.

A vantagem desta arquitetura é que ela fornece os componentes com as funcionalidades básicas para a criação da interligação de diversos provedores de nuvens. Outros componentes, com funcionalidades extras, também podem ser adicionados para fornecer "mais características". Isto direciona os pesquisadores que queiram desenvolver seus próprios sistemas de interoperabilidade.

Para trabalhos futuros, menciona-se: (a) análise de outros recursos computacionais, como por exemplo CPU e memória, utilizando a metodologia; (b) análise de outros algoritmos de modelagem do OpenModeller e também outras aplicações; e (c) adaptação desta metodologia para a caracterização de provedores de Plataforma como Serviço.

REFERÊNCIAS BIBLIOGRÁFICAS

ACETO, G. *et al.* Cloud monitoring: A survey. *Computer Networks*, v. 57, n. 9, p. 2093–2115, 2013.

AMAZON. *Amazon Elastic Compute Cloud (EC2)*. Disponível em: <http://aws.amazon.com/ec2/>. Acesso em: 2 fev. 2014a.

AMAZON. *Amazon High Performance Computing*. Disponível em: <http://aws.amazon.com/ec2/hpc-applications/>. Acesso em: 2 fev. 2014b.

AMAZON. *Amazon Simple Storage Service*. Disponível em: <http://aws.amazon.com/s3/>. Acesso em: 2 fev. 2014c.

AMAZON. *Amazon Web Services*. Disponível em: <http://aws.amazon.com/>. Acesso em: 2 fev. 2014d.

APACHE CLOUDSTACK. *Apache CloudStack: Open Source Cloud Computing*. Disponível em: <http://cloudstack.apache.org/>. Acesso em: 2 fev. 2014.

ARMBRUST, M. *et al.* Above the clouds: A Berkeley view of cloud computing. *EECS Department, University of California, Berkeley, Tech. Rep. UCB/EECS-2009-28*, 2009.

AZURE. *Windows Azure*. Disponível em: <http://www.windowsazure.com/>. Acesso em: 2 fev. 2014.

BARHAM, P. *et al.* Xen and the art of virtualization. In: THE 19TH ACM SYMPOSIUM ON OPERATING SYSTEMS PRINCIPLES, 2003, Bolton Landing. *Anais*... Bolton Landing: ACM, 2003. p. 177.

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, 2008.

BUYYA, R. *et al.* Cloud computing and emerging IT platforms: Vision, hype, and reality for delivering computing as the 5th utility. *Future Generation Computer Systems*, v. 25, n. 6, p. 599–616, 2009.

BUYYA, R.; MURSHED, M. Gridsim: A toolkit for the modeling and simulation of distributed resource management and scheduling for grid computing. *Concurrency and computation: practice and experience*, v. 14, n. 13-15, p. 1175–1220, 2002. Acesso em: 20 fev. 2014.

CALHEIROS, R. N. *et al.* CloudSim: a toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms. *Software: Practice and Experience*, v. 41, n. 1, p. 23–50, 2011.

CASANOVA, H. Simgrid: A toolkit for the simulation of application scheduling. In: FIRST IEEE INTERNATIONAL SYMPOSIUM ON CLUSTER COMPUTING AND THE GRID (CCGRID'01), 2001, Brisbane, Australia. *Anais*... Brisbane, Australia: IEEE, 2001. p. 430–437.

CHU, X. *et al.* Aneka: Next-generation enterprise grid platform for e-science and e-business applications. In: IEEE INTERNATIONAL CONFERENCE ON E-SCIENCE AND GRID COMPUTING, 2007, Bangalore. *Anais*... Bangalore: IEEE, 2007. p. 151 – 159.

CRIA. *Centro de Referência em Informação Ambiental*. Disponível em: <http://www.cria.org.br/>. Acesso em: 2 fev. 2014.

DAHARSH, J. *Problems I have had with GoGrid cloud hosting*. Disponível em: <http://www.daharsh.net/tech/2009/03/problems-i-have-had-with-gogrid-cloud.htm>. Acesso em: 2 fev. 2014.

DELTACLOUD. *Deltacloud*. Disponível em: <http://incubator.apache.org/deltacloud/>. Acesso em: 2 fev. 2014.

DMTF. *Open Virtualization Format (OVF) Specification, version 1.1*. Disponível em: <http://dmtf.org/sites/default/files/standards/documents/DSP0243_1.1.0.pdf>. Acesso em: 2 fev. 2014.

DONGARRA, J. J.; LUSZCZEK, P.; PETITET, A. The LINPACK benchmark: past, present and future. *Concurrency and Computation: practice and experience*, v. 15, n. 9, p. 803–820, 2003.

ECKEL, BRUCE. *Thinking in Java*. 4. ed. Boston: Prentice Hall, 2006.

ELASTICHOSTS. *ElasticHosts*. Disponível em: <http://www.elastichosts.com/>. Acesso em: 2 fev. 2014.

EXPÓSITO, R. R. *et al.* Analysis of I/O Performance on an Amazon EC2 Cluster Compute and High I/O Platform. *Journal of Grid Computing*, p. 1–19, 2013.

EXPÓSITO, R. R. *et al.* Performance analysis of HPC applications in the cloud. *Future Generation Computer Systems*, 2012.

FLEXISCALE. *FlexiScale Public Cloud*. Disponível em: <http://www.flexiant.com/products/flexiscale/>. Acesso em: 2 fev. 2014.

FOSTER, I. *et al.* Cloud computing and grid computing 360-degree compared. In: GRID COMPUTING ENVIRONMENTS WORKSHOP (GCE '08), 2008, Austin, TX. *Anais*... Austin, TX: IEEE, 2008. p. 131.

FOSTER, I. Globus toolkit version 4: Software for service-oriented systems. *Journal of Computer Science and Technology*, v. 21, n. 4, p. 513–520, 2006.

FOSTER, I. *et al. The physiology of the grid: An open grid services architecture for distributed systems integration*. . .: Open Grid Service Infrastructure WG, Global Grid Forum, 2002.

FOSTER, I.; KESSELMAN, C. Globus: A metacomputing infrastructure toolkit. *International Journal of High Performance Computing Applications*, v. 11, n. 2, p. 115, 1997.

FOSTER, I.; KESSELMAN, C.; TUECKE, S. The Anatomy of the Grid: Enabling Scalable Virtual Organizations. *International Journal of High Performance Computing Applications*, v. 15, n. 3, p. 200, 2001.

GHOSHAL, D.; CANON, R. S.; RAMAKRISHNAN, L. I/O performance of virtualized cloud environments. In: SECOND INTERNATIONAL WORKSHOP ON DATA INTENSIVE COMPUTING IN THE CLOUDS, 2011, New York. *Anais*... New York: ACM, 2011. p. 71–80.

GILADI, R.; AHITAV, N. SPEC as a performance evaluation measure. *Computer*, v. 28, n. 8, p. 33–42, 1995.

GNU. *GNU Compiler Collection (GCC)*. Disponível em: <http://gcc.gnu.org/>. Acesso em: 2 fev. 2014.

GOGRID. *GoGrid*. Disponível em: <http://www.gogrid.com/>. Acesso em: 2 fev. 2014.

GOOGLE. *Google App Engine*. Disponível em: <https://developers.google.com/appengine/>. Acesso em: 2 fev. 2014a.

GOOGLE. *Google Apps for Business*. Disponível em: <http://www.google.com/enterprise/apps/business/>. Acesso em: 2 fev. 2014b.

GREGG, B. *Systems Performance: Enterprise and the Cloud*. 1. ed. Boston: Pearson Education, 2013.

GROPP, W.; LUSK, E.; SKJELLUM, A. *Using MPI - Portable Parallel Programming with the Message Passing Interface*. 2. ed. Cambridge: MIT Press, 1999.

HAGEN, W. VON. *Professional Xen Virtualization*. Indianapolis: Wiley, 2008.

INPE. *Instituto Nacional de Pesquisas Espaciais*. Disponível em: <http://www.inpe.br/>. Acesso em: 2 fev. 2014.

INTEL. *Intel Compilers*. Disponível em: <http://software.intel.com/en-us/intel-compilers/>. Acesso em: 2 fev. 2014.

IOZONE. *Filesystem Benchmark*. Disponível em: <http://www.iozone.org/>. Acesso em: 2 fev. 2014.

JAIN, R. *The art of computer systems performance analysis*. .: Wiley Computer Publishing, 1991.

JAMES MURTY. *Programming Amazon Web Services: S3, EC2, SQS, FPS, and SimpleDB*. Sebastopol: O'Reilly, 2008.

JOHNSON, T. DE M. E S. M.; COUTINHO, M. C. *Avaliação de Desempenho de Sistemas Computacionais*. Rio de Janeiro: LTC, 2011.

KEAHEY, K. *et al.* Sky computing. *IEEE Internet Computing*, v. 13, n. 5, p. 43–51, 2009.

KVM. *KVM: Kernel-based Virtualization Driver (White paper)*. Disponível em: <http://www.linux-kvm.org/>. Acesso em: 2 fev. 2014.

LAUREANO, M. *Maquinas Viruais e Emuladores Conceitos, Teénicas e Aplicações*. .: Novatec Editora, 2006.

LIBCLOUD. *LibCloud*. Disponível em: <http://incubator.apache.org/libcloud/>. Acesso em: 2 fev. 2014.

MARATHE, A. *et al.* A comparative study of high-performance computing on the cloud. In: 22ND INTERNATIONAL SYMPOSIUM ON HIGH-PERFORMANCE PARALLEL AND DISTRIBUTED COMPUTING (HPDC '13), 2013, New York. *Anais*... New York: ACM, 2013. p. 239–250.

MESNIER, M.; GANGER, G. R.; RIEDEL, E. Object-based storage. *IEEE Communications Magazine*, v. 41, n. 8, p. 84–90, ago. 2003.

MUÑOZ, M. E. DE S. *et al.* openModeller: a generic approach to species' potential distribution modelling. *GeoInformatica*, v. 15, n. 1, p. 111–135, 2011.

NAS. *NAS Parallel Benchmarks*. Disponível em: <http://www.nas.nasa.gov/publications/npb.html>. Acesso em: 2 fev. 2014.

NS-2. *The Network Simulator (NS-2)*. Disponível em: <http://www.isi.edu/nsnam/ns/>. Acesso em: 2 fev. 2014.

NURMI, D. *et al.* The Eucalyptus open-source cloud-computing system. In: 9TH IEEE/ACM INTERNATIONAL SYMPOSIUM ON CLUSTER COMPUTING AND THE GRID, CCGRID '09, 2009, Shanghai. *Anais*... Shanghai: IEEE, 2009. p. 124–131.

NUVEMUSP. *Nuvem USP*. Disponível em: <https://wiki.uspdigital.usp.br/nuvem/Paginas/Home.aspx>. Acesso em: 2 fev. 2014.

OCCI-WG. *Open Cloud Computing Interface (OCCI) Specification, doc15731, version 5*. Disponível em: <https://forge.ogf.org/sf/go/doc15731>. Acesso em: 2 fev. 2014.

OPENSTACK. *OpenStack*. Disponível em: <http://openstack.org/>. Acesso em: 2 fev. 2014a.

OPENSTACK, F. *OpenStack Operations Guide*. . .: OpenStack Foundation, 2014b. Disponível em: <http://docs.openstack.org/trunk/openstack-ops/content/index.html>. Acesso em: 1 fev. 2014.

PARKHILL, D. F. *The Challenge of the Computer Utility*. .: Addison-Wesley Reading, 1966.

POLI/USP. *Escola Politécnica da Universidade de São Paulo*. Disponível em: <http://www.poli.usp.br/>. Acesso em: 2 fev. 2014.

RACKSPACE. *Rackspace*. Disponível em: <http://www.rackspace.com/>. Acesso em: 2 fev. 2014.

ROCKWELL. *Arena Simulation*. Disponível em: <http://www.arenasimulation.com/Arena_Home.aspx>. Acesso em: 2 fev. 2014.

RODAMILANS, C. B.; BARUCHI, A.; MIDORIKAWA, E. T. Experiences Applying Performance Evaluation to Select a Cloud Provider. In: 8TH INTERNATIONAL CONFERENCE ON COMPUTER ENGINEERING AND APPLICATIONS (CEA-14), 2014, Tenerife, Spain. *Anais*... Tenerife, Spain: World Scientific and Engineering Academy and Society (WSEAS), 2014.

SALESFORCE. *Salesforce*. Disponível em: <http://www.salesforce.com/>. Acesso em: 2 fev. 2014a.

SALESFORCE. *SalesForce Platform*. Disponível em: <http://www.salesforce.com/platform/>. Acesso em: 2 fev. 2014b.

SILBERSCHATZ, A.; GALVIN, P. B.; GAGNE, G. *Operating system concepts*. 9. ed. .: John Wiley & Sons, 2013. v. 8.

SMITH, J.; NAIR, R. *Virtual Machines: Versatile Platforms for Systems and Processes*. 1. ed. .: Morgan Kaufmann, 2005.

SOTOMAYOR, B. *et al.* Resource leasing and the art of suspending virtual machines. In: 11TH IEEE INTERNATIONAL CONFERENCE ON HIGH PERFORMANCE COMPUTING AND COMMUNICATIONS (HPCC ’09), 2009a, Seoul. *Anais*... Seoul: IEEE, 2009. p. 59–68.

SOTOMAYOR, B. *et al.* Virtual infrastructure management in private and hybrid clouds. *IEEE Internet Computing*, v. 13, p. 14–22, 2009b.

SOTOMAYOR, B.; KEAHEY, K.; FOSTER, I. Combining batch execution and leasing using virtual machines. In: 17TH INTERNATIONAL SYMPOSIUM ON HIGH PERFORMANCE DISTRIBUTED COMPUTING (HPDC ’08), 2008, Boston. *Anais*... Boston: ACM, 2008. p. 87–96.

STAX. *Stax Networks*. Disponível em: <http://www.stax.net/>. Acesso em: 2 fev. 2014.

TAURION, C. *Cloud Computing - Computação Em Nuvem. Transformando O Mundo Da Tecnologia Da Informaçao*. Rio de Janeiro: BRASPORT, 2009.

TERRY, D. B. *et al.* Consistency-based service level agreements for cloud storage. In: 24TH ACM SYMPOSIUM ON OPERATING SYSTEMS PRINCIPLES, 2013,

Farmington, Pennsylvania. *Anais*... Farmington, Pennsylvania: ACM, 2013. p. 309–324.

VARGA, A. The OMNeT++ discrete event simulation system. In: EUROPEAN SIMULATION MULTICONFERENCE (ESM'2001), 2001, Prague, Czech Republic. *Anais*... Prague, Czech Republic: Eurosis, 2001. p. 185.

VECCHIOLA, C.; CHU, X.; BUYYA, R. Aneka: a software platform for .NET-based Cloud computing. *High Performance & Large Scale Computing, Advances in Parallel Computing. IOS Press, Amsterdam*, 2009.

VECCHIOLA, C.; PANDEY, S.; BUYYA, R. High-performance cloud computing: A view of scientific applications. In: 10TH INTERNATIONAL SYMPOSIUM ON PERVASIVE SYSTEMS, ALGORITHMS, AND NETWORKS (ISPAN '09), 2009, Kaohsiung. *Anais*... Kaohsiung: IEEE, 2009. p. 4–16.

VEDAM, V.; VEMULAPATI, J. Demystifying Cloud Benchmarking Paradigm-An in Depth View. In: IEEE 36TH ANNUAL COMPUTER SOFTWARE AND APPLICATIONS CONFERENCE (COMPSAC), 2012, Izmir. *Anais*... Izmir: IEEE, 2012. p. 416–421.

VIRTUALBOX. *VirtualBox*. Disponível em: <http://www.virtualbox.org/>. Acesso em: 2 fev. 2014.

VMWARE. *VMware Virtualization Software*. Disponível em: <http://www.vmware.com/>. Acesso em: 2 fev. 2014.

VSPHERE. *VMware vSphere*. Disponível em: <http://www.vmware.com/products/vsphere/>. Acesso em: 2 fev. 2014.

WALKER, E. Benchmarking amazon EC2 for high-performance scientific computing. *USENIX Login*, v. 33, n. 5, p. 18–23, 2008.