ICE - Uma ferramenta para emulação de InterClouds baseadas em OpenStack

Tiago Ferreto[1], **Fabricio Cordella**[1], **Vinicius Gubiani**[1], **Miguel Da Silva**[1]

[1]Faculdade de Informática
Pontifícia Universidade Católica do Rio Grande do Sul (PUCRS)
Porto Alegre – RS – Brazil

tiago.ferreto@pucrs.br

{fabricio.cordella,vinicius.gubiani,miguel.silva.001}@acad.pucrs.br

Abstract. *Demand for greater availability and efficiency of applications requires the simultaneous use of several cloud providers, in a model called InterCloud. OpenStack is one of the main open platforms for cloud computing and allows the creation of InterClouds through the use of regions. Currently, the tools used to create environments for the development of OpenStack do not support the automated creation of InterClouds. This work presents the ICE tool, a tool for InterClouds emulation based on OpenStack.*

Resumo. *A demanda por maior disponibilidade e eficiência das aplicações exige o uso simultâneo de diversos provedores de nuvem, num modelo chamado InterCloud. O OpenStack é uma das principais plataformas abertas para computação em nuvem e permite a criação de InterClouds através do uso de regiões. Atualmente, as ferramentas utilizadas para criação de ambientes para desenvolvimento do OpenStack não suportam a criação automatizada de InterClouds. Este trabalho apresenta a ferramenta ICE, uma ferramenta para emulação de InterClouds baseadas no OpenStack.*

1. Introdução

A computação em nuvem é definitivamente uma das mais importantes mudanças de paradigma da última década. Este novo modelo permitiu que qualquer pessoa possa acessar sob demanda recursos computacionais em larga escala. Empresas de diversos tamanhos, de pequenas startups até multinacionais, podem se beneficiar das características oferecidas pela nuvem. Algumas das principais vantagens incluem a possibilidade de utilizar uma pequena quantidade de recursos, expandi-las conforme a necessidade, e a possibilidade de pagar somente pelos recursos que foram realmente consumidos, usando o modelo pague o que usar (*pay-per-use*) [Mell and Grance 2011].

Apesar dos ganhos evidentes, a computação em nuvem também possui alguns desafios. A utilização de um único conjunto centralizado de recursos fornecido pelo provedor da nuvem pode ocasionar uma indisponibilidade temporária do serviço, deixando milhares de usuários sem acesso aos seus recursos. Outro desafio é garantir o acesso com baixa latência a uma aplicação para usuários espalhados em diversas regiões do mundo [Grozev and Buyya 2014]. Estes desafios levaram a necessidade de utilizar simultaneamente diversos provedores de nuvem através do modelo de InterCloud, visando

aumentar a disponibilidade da aplicação e permitindo que a mesma seja migrada para regiões mais próximas dos seus usuários. A integração entre nuvens pode ser realizada através da utilização de protocolos e padrões abertos e bem definidos.

Uma das principais plataformas abertas de computação em nuvem é o OpenStack. O OpenStack pode ser utilizado para construção de infraestruturas de nuvem pública ou privada. Ele permite o provisionamento de diversos tipos de recursos, tais como: máquinas virtuais, armazenamento em bloco, armazenamento baseado em objetos, redes virtuais, balanceadores de carga, entre outros. Uma das principais funcionalidades do OpenStack é o seu suporte a mecanismos de particionamento. Através deste mecanismos, é possível segregar recursos baseados em localidade, propriedades em comum, domínios de falha ou quaisquer outras razões, de forma a aumentar a escalabilidade da infraestrutura. Um destes mecanismos é conhecido como Regiões e permite a criação de uma InterCloud usando o modelo de Federação Voluntária Centralizada.

Uma das principais dificuldades no uso do OpenStack é o seu processo de instalação, que pode ser bastante extenso e complexo. Além de impactar a criação de ambientes de produção, isso prejudica ainda mais atividades de desenvolvimento e validação da plataforma, onde é necessário realizar a instalação do software diversas vezes, usando diversas configurações. Existem atualmente algumas ferramentas (DevStack [OpenStack Foundation 2016a], Fuel [Fuel Community Project 2016], Packstack [Packstack 2016] e OpenStack-Ansible [OpenStack Foundation 2016b]) que automatizam e facilitam a instalação de um ambiente de nuvem OpenStack em uma única máquina. No entanto, nenhuma destas ferramentas permite a criação automatizada de ambientes InterCloud baseados no OpenStack compostos por múltiplas regiões e nós de computação.

Esse artigo apresenta a ferramenta ICE (InterCloud Emulator), uma ferramenta para a criação automatizada de ambientes InterCloud de testes baseados no OpenStack. O ICE utiliza máquinas virtuais e containers para emular uma InterCloud em uma única máquina. Cada região, localizada em uma máquina virtual independente, é criada usando a ferramenta DevStack. A ferramenta ICE também utiliza o software Rally, visando a realização de *benchmarks* simulando requisições de diversos usuários. Testes realizados mostraram que o ICE consegue provisionar ambientes InterCloud, voltado à experimentos, compostos por diversas regiões e nós de computação em um tempo um pouco maior do que o necessário para criar uma única nuvem usando o DevStack no modo AIO (All-in-One). O ICE não deve ser utilizado em produção, pois foi projetado inicialmente para ser executado em apenas um servidor físico.

O artigo está organizado da seguinte forma. A Seção 2 introduz o conceito de InterCloud e apresenta uma classificação dos seus diversos tipos. A Seção 3 apresenta o software OpenStack e os seus mecanismos de particionamento, utilizados para criação de InterClouds. A Seção 4 apresenta as diversas ferramentas existentes para criação automatizada de ambientes OpenStack. A Seção 5 descreve a ferramenta ICE, sua arquitetura e implementação interna. A Seção 6 apresenta a avaliação dos resultados obtidos no trabalho. A Seção 7 apresenta as conclusões e os trabalhos futuros que poderão ser implementados.

2. InterClouds

A centralização dos recursos computacionais promovida pela computação em nuvem simplificou significativamente o processo de alocação e uso de infraestrutura de TI para execução de serviços e aplicações. No entanto, essa mesma estratégia aumentou o risco de indisponibilidade, devido ao uso de uma única infraestrutura, assim como dificultou o desenvolvimento de aplicações que demandam baixa latência de acesso e são usados por clientes espalhados pelo mundo. A InterCloud é um modelo de nuvem que visa garantir a qualidade dos serviços como performance e disponibilidade. Ela permite a realocação sob demanda de recursos e a transferência de carga de trabalho através de diferentes infraestruturas de nuvem com base nos requisitos de cada usuário [Cases 2010].

A principal característica de uma InterCloud para os seus clientes, é a possibilidade de escolher diferentes fornecedores e infraestruturas em diversas localizações geográficas. Desse modo, eles podem fazer com que seus negócios sejam adaptáveis às políticas e alterações dos fornecedores do serviço, facilitando a expansão dos seus negócios. Além disso, garantem benefícios como a resiliência das aplicações, a capacidade de alocação em diversas regiões geográficas e a vantagem de evitar a dependência do fornecedor. Além dos benefícios aos clientes, o modelo de InterCloud também pode ser vantajoso aos provedores de nuvem. Através da ligação com outros provedores, o provedor de nuvem é capaz de expandir a sua capacidade sob demanda e atender melhor requisitos de garantia de serviço. Esse ganho em capacidade e qualidade torna o provedor de nuvem mais atrativo aos seus clientes.

As InterClouds podem ser classificadas [Grozev and Buyya 2014] em: Federação Voluntária ou Independente. No modelo de Federação Voluntária, os provedores colaboram voluntariamente entre si para trocar recursos. No modelo Independente, também conhecido como Multi-Cloud, diversas nuvens são usadas de forma agregada por uma aplicação ou *broker*. As InterClouds com Federação Voluntária podem ser ainda divididas em: Centralizadas ou Peer-to-Peer. No modelo Centralizado, existe uma entidade central para realizar ou facilitar a alocação dos recursos. No modelo Peer-to-Peer, as nuvens que compõem a InterCloud se comunicam diretamente uma com as outras para alocação do recurso, sem a existência de um centralizador.

O modelo de InterCloud Independente pode ser ainda classificado como: baseado em Serviços ou Bibliotecas. No modelo baseado em Serviços, existe um *broker* externo ou ligado diretamente ao cliente de nuvem que realiza o provisionamento de recursos nas diversas nuvens. No modelo baseado em Biblioteca, o cliente de nuvem utiliza uma biblioteca que facilita o acesso uniforme aos diversos provedores de nuvem para implementar o seu próprio *broker*. Este trabalho visa a emulação de InterClouds baseadas em OpenStack usando o modelo de Federação Voluntária Centralizada.

3. OpenStack

O OpenStack [OpenStack Foundation 2016c] é um software open source para construção de nuvens públicas ou privadas, constituído por uma série de componentes interrelacionados que trabalham para oferecer um ambiente completo de nuvem. Os componentes do OpenStack gerenciam os recursos de computação, armazenamento e rede de um data center com o objetivo de auxiliar na construção de infraestruturas de nuvem escaláveis, independentemente de seu tamanho. O OpenStack utiliza e suporta

em sua arquitetura o modelo de Federação Voluntária Centralizada, porém é possível também o uso do modelo de InterCloud Independente baseado em bibliotecas como a jclouds [Apache Software Foundation 2016], a qual suporta diversos provedores de nuvem, incluindo o OpenStack.

O projeto foi iniciado pelo trabalho de duas organizações: Rackspace Hosting e NASA que uniram forças e disponibilizaram suas plataformas internas de armazenamento e computação em nuvem como um projeto integrado. Atualmente, o OpenStack conta com o apoio financeiro de dezenas de empresas como DELL, HP e IBM, além do trabalho de milhares de desenvolvedores distribuídos em 132 países, visando a criação de um padrão aberto para plataformas de nuvem. O objetivo a longo prazo é criar uma plataforma open source que atenda as necessidades de provedores de nuvens públicas e privadas.

Cada componente disponibiliza mecanismos de interconexão fracamente acoplados [Fifield et al. 2014] de maneira que os recursos da plataforma sejam geridos pelos usuários através de um painel de controle web, desenvolvido como uma camada separada utilizando esses mecanismos, com mínima interação com o provedor de serviços. Este tipo de decisão de projeto permite que as funcionalidades do OpenStack sejam estendidas e modificadas, facilitando também a integração com softwares de terceiros. O OpenStack permite a integração de diferentes tecnologias para a construção de nuvens, promovendo grande flexibilidade. Os principais componentes do OpenStack são:

Nova Controla a infraestrutura computacional da nuvem como um todo, gerenciando a memória, o processamento e a alocação dos recursos.

Neutron Permite o gerenciamento da rede, criação de redes, configuração de VLANs e regras de firewall.

Swift É um sistema de armazenamento escalável e redundante, o qual gerencia possíveis falhas replicando e garantindo a integridade e a disponibilidade dos dados.

Horizon Provê uma interface gráfica amigável, através de um navegador, para a administração de serviços.

Keystone Fornece o serviço de autenticação e autorização, que reúne informações dos usuários e quais os serviços que os mesmos podem utilizar.

Glance Gerencia e disponibiliza as imagens das máquinas virtuais que podem ser instanciadas e usadas.

O OpenStack possui alta flexibilidade para criação de arquiteturas de nuvens com diferentes graus de tamanho e disponibilidade. Para isso, o OpenStack permite o uso de mecanismos de particionamento. Os mecanismos providos pelo OpenStack são: Regiões (Regions), Zonas de Disponibilidade (Availability Zones), Agregados de Hosts (Host Aggregates) e Células (Cells).

Regiões são usadas para segregar uma nuvem. Cada região possui uma interface de acesso independente e não existe coordenação entre regiões. Regiões podem compartilhar alguns componentes, tais como: Keystone para gerência de identidade, Horizon para o painel de serviços, e Swift Proxy para armazenamento baseado em objetos centralizado. O usuário deve explicitamente selecionar uma região para alocar seus recursos. Alguns casos de uso de regiões incluem: (i) uma nuvem com múltiplos sites dispersos geograficamente, no qual o usuário deseja escalonar VMs para um site específico através de um único portal; e (ii) provedor de nuvem com suporte a recuperação de desastres.

Zonas de Disponibilidade são usadas para agrupar um subconjunto de recursos dentro de uma região. Estes recursos são normalmente agrupados baseado em dependências compartilhadas, tais como: fonte de energia, dispositivos de rede, etc. Recursos dentro de uma zona de disponibilidade possuem boa conectividade e baixa latência, enquanto recursos em diferentes zonas de disponibilidade provêm alta disponibilidade. Zonas de disponibilidade podem ser visíveis aos usuários, isto é, um usuário pode selecionar uma zona de disponibilidade específica para escalonar uma instância. Algoritmos de escalonamento podem ser usados para alocar réplicas de instâncias em zonas de disponibilidade distintas visando aumentar a disponibilidade da aplicação.

Agregados de Hosts permitem o particionamento dentro de uma Zona de Disponibilidade. Recursos são agrupados com base em metadados e o particionamento só é visível ao administrador da nuvem. Agregados de Hosts podem ser usados em diferentes cenários, tais como: (i) para permitir políticas complexas de escalonamento baseado em categorização de recursos; (ii) nuvem com múltiplos hypervisors; (iii) definição de grupos lógicos dentro de uma infraestrutura; e (iv) recursos heterogêneos.

Células são similares a regiões. Elas também são usadas para segregar uma nuvem, mas provêm uma coordenação centralizada entre nuvens (células). Nuvens implementadas com Células provêm um controle centralizado (API Cell) para tratar requisições de usuários. O escalonamento de instâncias é realizado em duas etapas: primeiro determinando qual célula deve tratar a requisição (escalonamento de célula), e segundo, determinando qual máquina deve hospedar a instância (escalonamento do nova). Usuários não podem selecionar a célula que desejam usar para hospedar uma instância.

Os mecanismos de regiões e células permitem a criação de InterClouds com Federação Voluntária Centralizadas. A ferramenta ICE, apresentada neste trabalho, utiliza o mecanismo de regiões para criação de InterClouds.

4. Trabalhos relacionados

Apesar da sua arquitetura baseada em componentes, o OpenStack é conhecido pela sua alta complexidade de instalação. A instalação manual do OpenStack pode ser bastante complicada e demorada. Os diversos detalhes necessários para configuração dos componentes requerem bastante atenção do administrador, a fim de evitar erros durante a instalação. Isso é ainda mais crítico quando se deseja testar diferentes configurações ou alterações na implementação dos componentes.

Para diminuir o alto custo de instalação de uma nuvem OpenStack real, ou mesmo para fins de teste, existem diversos projetos que automatizam a instalação dos componentes. Diversos destes projetos também permitem a criação de um ambiente AIO (All-In-One), onde todos os componentes são instalados em uma mesma máquina física ou virtual para fins de teste. Alguns destes projetos são: DevStack [OpenStack Foundation 2016a], Fuel [Fuel Community Project 2016], Packstack [Packstack 2016] e OpenStack-Ansible [OpenStack Foundation 2016b].

O DevStack [OpenStack Foundation 2016a] é uma ferramenta baseada em scripts bash para automatizar a criação de ambientes de desenvolvimento usando o OpenStack. O DevStack permite a criação de um ambiente AIO usando máquinas físicas, virtuais ou containers. Também é possível usar o DevStack para criar ambientes compostos por

diversos nós. O DevStack utiliza para instalação o código do OpenStack disponibilizado através do repositório principal do projeto. Portanto, ele é bastante utilizado para realizar testes de novas versões do OpenStack. O DevStack permite a configuração do ambiente a ser criada através de um arquivo de configuração (local.conf), onde são definidos os componentes que devem ser instalados, assim como parâmetros básicos de configuração.

O Fuel [Fuel Community Project 2016] é uma ferramenta open source usada para provisionar e gerenciar ambientes OpenStack. Ele pode ser usado para provisionar um ambiente OpenStack inteiro em uma máquina virtual para fins de teste, ou em hardware real para produção, incluindo o provisionamento de ambientes com alta disponibilidade formados por múltiplos nós. Para facilitar a instalação e gerência, a ferramenta provê uma interface gráfica Web, ao invés do uso de linha de comando como o DevStack. A arquitetura do Fuel utiliza as ferramentas Cobbler e Puppet para provisionamento e configuração dos componentes.

O Packstack [Packstack 2016] é uma ferramenta similar ao DevStack voltada a distribuições RedHat, CentOS ou Fedora. Ele utiliza módulos Puppet e o protocolo ssh para instalação do OpenStack em diversas máquinas, ou em uma única máquina para fins de teste. O Packstack é utilizado via linha de comando e permite uma configuração interativa dos diversos componentes, ou totalmente automatizada através de um arquivo com configurações (answer file).

O OpenStack-Ansible [OpenStack Foundation 2016b] permite o provisionamento de ambientes de produção ou teste (AIO) usando a ferramenta de automação Ansible. O ambiente é provisionado a partir do código existente no repositório do OpenStack. O OpenStack-Ansible utiliza LXC (Linux Containers) para instalação dos diversos componentes do OpenStack visando o isolamento e facilidade de manutenção.

Todas as ferramentas citadas possuem aproximadamente as mesmas funcionalidades. Elas permitem a criação automatizada de ambientes AIO para teste e ambientes de produção (com exceção do DevStack) usando diversas máquinas. A principal limitação encontrada em todas as ferramentas, e tratada na ferramenta ICE apresentada neste trabalho, é o suporte a criação de ambientes de teste usando múltiplas regiões e a possibilidade de emular diversos nós de computação. A próxima seção apresenta a ferramenta ICE e como ela permite a emulação de múltiplas regiões OpenStack, suportando diversos nós de computação em cada região.

5. ICE - InterCloud Emulator

A ferramenta desenvolvida durante esse trabalho tem como objetivo provisionar um ambiente InterCloud de forma automatizada e sem uma configuração complexa. Além disso, também tem o objetivo de validar o ambiente desenvolvido com o auxílio do Rally.

A ferramenta apresenta as seguintes funcionalidades e características:

- Provisionamento de uma InterCloud OpenStack para testes baseada em multi-regiões.
- Possibilidade de realizar *benchmark* sobre o ambiente InterCloud afim de verificar seu comportamento diante de determinada carga de usuários.
- Isolamento do ambiente InterCloud do usuário, bem como entre regiões. Não havendo necessidade de acessar diretamente as máquinas virtuais criadas.

5.1. Arquitetura

Para este trabalho optou-se por uma arquitetura modular, robusta e bastante isolada em termos de componentes. O objetivo desejado é que cada região seja executada em uma máquina virtual, assim como um controlador para os testes de *benchmark*. Também deseja-se que os ambientes de cada nuvem consigam comunicar-se entre si, e o usuário preocupe-se apenas em configurar corretamente os dados de entrada e obtenha uma saída clara ao final da execução da ferramenta. A Figura 1 apresenta de forma simplificada o projeto da arquitetura.

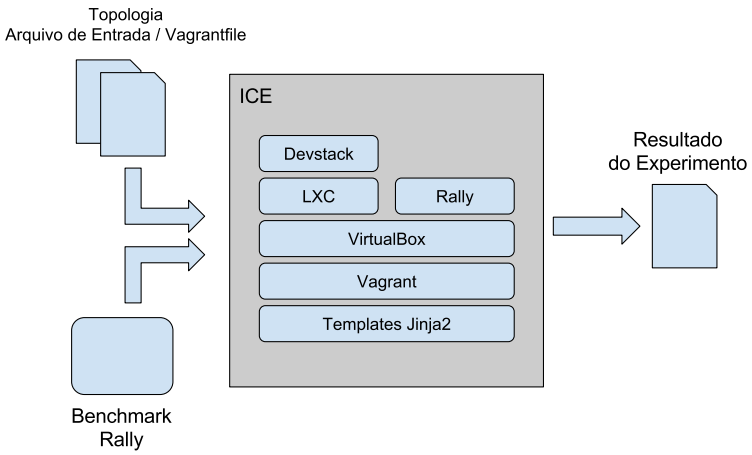

Figura 1. Arquitetura da ferramenta

Dentro de cada máquina virtual responsável por agrupar uma região, temos contêineres do tipo LXC. Eles foram usados para garantir o isolamento e poder simular uma configuração multi-nodo sem necessitar de uma grande quantidade de recursos. Cada região contém um nodo de controle e uma quantidade enesimal de recursos computacionais– chamados aqui de *compute nodes*– quantidade essa que é definida pelo usuário no arquivo de descrição na seção 5.2.

O nodo de controle é responsável por centralizar os serviços de autenticação e autorização, gerenciamento de imagem, e geração de novos nodos de computação. Os *compute nodes* por sua vez são diretamente responsáveis por executar o processamento requisitado pelos usuários.

Durante a implementação, utilizamos diversas ferramentas para auxiliar o desenvolvimento. Dentre elas estão Jinja2, Vagrant, VirtualBox, Linux Container, Rally e DevStack.

Como forma de provisionar e gerenciar a infraestrutura de cada nuvem, utilizamos o Vagrant junto com o VirtualBox. O Vagrant [Hashimoto 2013] é uma poderosa ferramenta para gerar e configurar ambientes virtualizados completos em desenvolvimento e em produção, além de oferecer diversas formas de virtualização e ser facilmente estendida através de *plugins*.

O Vagrant possui duas bases fundamentais: boxes [HashiCorp 2016b] – imagens reduzidas e empacotadas dos sistemas operacionais; e o *Vagrantfile* [HashiCorp 2016a] – arquivo que descreve a criação das VMs, junto com os softwares e configurações que serão instalados nas mesmas. Ao executar a ferramenta, o *Vagrantfile* é interpretado para geração do ambiente InterCloud.

A possibilidade de utilizar uma box implicou em uma otimização da construção do ambiente de nuvem, uma vez que temos por objetivo gastar o mínimo de tempo possível com ações repetitivas. Também por esse motivo, decidimos utilizar o Vagrant no desenvolvimento desse trabalho. Para utilizar essa estratégia, antes do primeiro provisionamento, a ferramenta desenvolvida cria uma máquina virtual com todas as pré-configurações necessárias, e ela será utilizada como imagem para todas as demais máquinas virtuais. Ela então é compactada em uma box e adicionada à lista de boxes do Vagrant.

O Linux Container trabalha com a tecnologia de virtualização a nível de sistema operacional. Neste tipo de virtualização, são criados compartimentos a fim de permitir que cada VM possa ter recursos limitados de memória, CPU e rede [Canonical 2015]. Isto oferece melhor flexibilidade e melhoria de desempenho na criação de ambientes virtuais. Por esta razão, foi utilizada a tecnologia de contêiner no projeto. Para prover a conectividade o LXC cria uma bridge que interliga os contêineres com a rede externa caracterizando um isolamento de quadros permitindo aos contêineres obter interfaces de rede próprias.

A instalação do OpenStack foi feita de forma automatizada através da ferramenta DevStack – versão Kilo – que já foi discutida anteriormente na seção 4. O serviço de identificação do OpenStack – Keystone– foi centralizado na região primária com o objetivo de simular a conectividade de múltiplas regiões.

Para adaptar o *Vagrantfile* à demanda do usuário, conforme o arquivo de entrada, foi utilizada a biblioteca Jinja2 de Python para modelar *templates*. Esses *templates* geram a quantidade de regiões existentes no arquivo de entrada, com suas respectivas quantidades de computes-nodes, e também a máquina virtual responsável pelos testes de *benchmark*. Durante a construção da nuvem, a criação dos computes-nodes ocorrem de forma paralelizada, enquanto a próxima região é gerada. Esse paralelismo é uma das causas pelos bons resultados, que serão apresentados na seção 6.

O Rally é uma ferramenta de *benchmark* que automatiza e unifica a montagem, a validação e a avaliação de um ambiente OpenStack multi-nodo. Ela permite simular uma grande quantidade de usuários e solicitações sendo realizadas, gerando uma carga significativa sobre os ambientes da nuvem.

5.2. Descrição da topologia

Logo nos primeiros testes com o Vagrant, percebeu-se que o *Vagrantfile* seria limitado e em certos casos insuficiente para descrever todas as funcionalidades inicialmente discutidas para a InterCloud. Por esse motivo, para permitir uma liberdade maior de características em cada nuvem, optou-se por utilizar um arquivo JSON para descrever o ambiente a ser testado. A sintaxe do formato JSON também é largamente utilizada em projetos diversos, é de fácil compreensão e validação, utiliza o formato chave-valor, listas de elementos, e elementos aninhados. Esses foram os principais fatores que influencia-

ram fortemente na sua escolha para esse trabalho. Na tabela 1 serão explicados os campos utilizados no arquivo de entrada, enquanto a Figura 2 apresenta a estrutura do arquivo de entrada utilizado.

Tabela 1. Campos do arquivo de entrada

Informação/chave	**Descrição/Valor**
regions	Lista contendo todos os elementos que representam cada uma das nuvens.
name	Nome para ser utilizado em cada ambiente.
controller_ip	IP que será usado em cada nodo de controle de cada região.
hostIp	IP padrão que será usado em cada máquina virtual.
bridge	Nome da interface de rede Bridge que será usada no Vagrantfile.
numberOfComputeNodes	Quantidade de nodos de computação que serão criados.

```
{
    "regions": [
        {
            "name":"RegionOne",
            "controller_ip": "192.100.100.10",
            "virtualBoxSpecs": {
                "hostIp": "100.0.0.11",
                "bridge": "br1"
            },
            "numberOfComputeNodes": 2
        },
      {
            "name":"RegionTwo",
            "controller_ip": "192.100.200.10",
            "virtualBoxSpecs": {
                "hostIp": "100.0.0.12",
                "bridge": "br1"
            },
            "numberOfComputeNodes": 2
        }
    ]
}
```

Figura 2. Exemplo de arquivo JSON de entrada

A Topologia física do experimento foi configurada usando-se uma interface de bridge no servidor físico, com conectividade através de rotas para a rede local e para Internet, através do uso de NAT – *Network Address Translation*. Através desta bridge são conectadas as máquinas virtuais com suas respectivas interfaces lógicas, representando regiões com conectividades distintas.

Em cada região são necessárias as *bridges* do LXC para a demanda de conectividade dos contêineres com a rede externa. A topologia física do experimento é detalhada conforme a Figura 3.

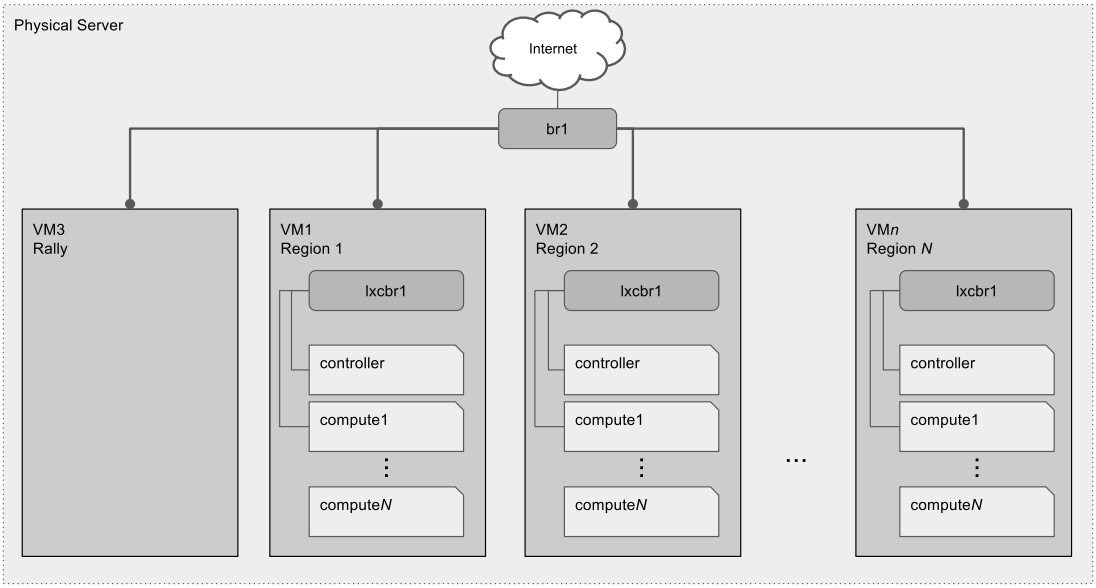

Figura 3. Topologia física do experimento

5.3. Descrição de experimento

A ferramenta desenvolvida pode ser utilizada através da linha de comando em um ambiente linux. A Tabela 2 explica em detalhes os parâmetros utilizados na ferramenta.

Tabela 2. Campos utilizados

Parâmetro	Descrição de cada parâmetro
-i	Arquivo de entrada descrevendo a infraestrutura das clouds
-o	Arquivo de saída - Relatório em HTML e JSON gerado pelo Rally
-p	Provisionamento - Executa a geração do ambiente
-r	Provisionamento do Rally
-b	Benchmark - Dá início aos testes de *benchmark* configurados no Rally e devolve os resultados no local indicado pelo parâmetro -o

O comando abaixo é um exemplo de provisionamento de uma InterCloud.

ICE -i inputfile -o outputfile -p -b benchmarkfile

6. Avaliação

O ambiente para o experimento é formado por um servidor PowerEdge R730 Dell com 2 processadores Intel Xeon E5-2650v3 2.30GHz (cada um com: 10 cores físicos + 10 cores virtuais) e 224 GB de memória RAM. Para estabilizar o tempo do experimento e impedir interferências externas de rede, foi criado um servidor de cache de onde eram realizados os downloads de pacotes e de imagens durante a construção da nuvem.

Na avaliação do trabalho foram realizados vários testes usando uma ou mais regiões, e cada região contendo um ou mais *compute nodes*. Durante os testes, obti-

vemos um tempo médio de 12 minutos para a pré-configuração. Os dados da Figura 4 levam em consideração que essa etapa já foi realizada.

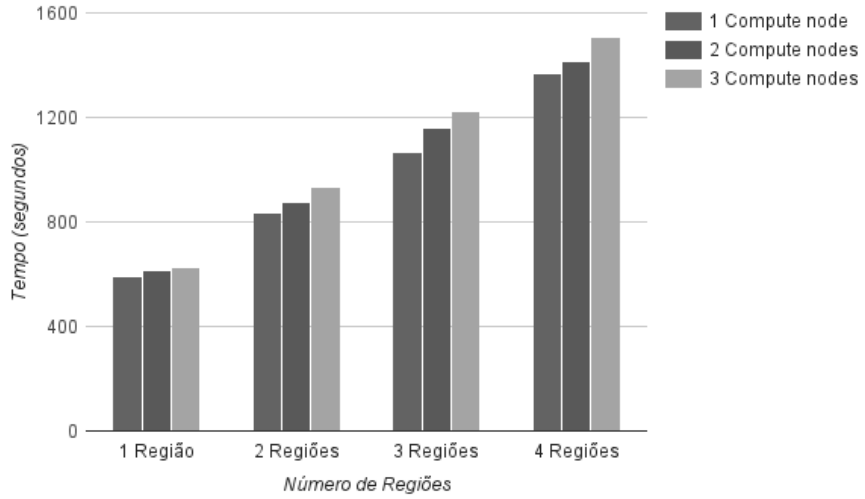

Figura 4. Gráfico com os tempos de cada teste

Como podemos ver pela Figura 4, o acréscimo de regiões causa um aumento linear no tempo total para a construção da nuvem – em média 273 segundos para cada região adicionada. De forma similar, o incremento de *compute nodes* por região implica em um aumento também linear no tempo total para a construção da nuvem – em média 54 segundos por *compute node* adicionado.

O tempo padrão para download e instalação de uma instância do DevStack – *All-in-One*–, sem a ferramenta, também utilizando o servidor de cache, é de aproximadamente 17 minutos. De posse desses dados, pode-se observar que a partir de uma InterCloud com duas regiões, obtém-se um ganho de tempo na construção da nuvem.

7. Conclusões e trabalhos futuros

Após todas as etapas de desenvolvimento desse trabalho, verificou-se que a ferramenta ICE foi capaz de atender os objetivos inicialmente planejados: criação rápida de ambientes de teste e de desenvolvimento em nuvem, assim como a avaliação do ambiente através de *benchmarks*. Ela ainda encontra-se em estágio inicial de desenvolvimento, e com certeza poderá ser melhorada em trabalhos futuros.

Dentre os muitos recursos discutidos durante o desenvolvimento do trabalho que poderiam ser incorporados ao mesmo, pode-se citar: implementação da nuvem utilizando células; promover InterCloud em nuvens baseadas em diferentes plataformas; avaliar cenários de exceção assim como a sua escalabilidade com mais regiões; recursos para simular desempenho de *links* de internet entre *clouds* – latência e vazão; simulação de localização geográfica em cada região – latitude e longitude; tornar a construção do ambiente ainda mais rápida através da paralelização de parte da ferramenta.

Referências

Apache Software Foundation (2016). Apache jclouds. https://jclouds.apache.org.

Canonical (2015). Linux container, https://linuxcontainers.org/.

Cases, U. (2010). Functional requirements for inter-cloud computing. In *Global Inter-Cloud Technology Forum, GICTF White Paper*.

Fifield, T., Fleming, D., Gentle, A., Hochstein, L., Proulx, J., Toews, E., and Topjian, J. (2014). *OpenStack Operations Guide*. "O'Reilly Media, Inc.".

Fuel Community Project (2016). Fuel for openstack. https://www.fuel-infra.org.

Grozev, N. and Buyya, R. (2014). Inter-cloud architectures and application brokering: taxonomy and survey. *Software: Practice and Experience*, 44(3):369–390.

HashiCorp (2016a). Vagrant boxes. https://www.vagrantup.com/docs/vagrantfile/.

HashiCorp (2016b). Vagrantfile. https://www.vagrantup.com/docs/getting-started/boxes.html.

Hashimoto, M. (2013). *Vagrant: Up and Running*. O'Reilly.

Mell, P. and Grance, T. (2011). The nist definition of cloud computing.

OpenStack Foundation (2016a). Devstack - an openstack community production. http://docs.openstack.org/developer/devstack/.

OpenStack Foundation (2016b). Openstack ansible. https://wiki.openstack.org/wiki/OpenStackAnsible.

OpenStack Foundation (2016c). Openstack open source cloud computing software. http://openstack.org.

Packstack (2016). Packstack. https://github.com/openstack/packstack.