Explorando a elasticidade de nuvens IaaS para reconfigurar dinamicamente aplicações n-camadas

Devair Dener Darolt, Felipe Rodrigo de Souza, Guilherme Piegas Koslovski [1]

Resumo: Com o surgimento das nuvens computacionais dinâmicas, aspectos de gerenciamento relacionados com provisionamento sob demanda, escalável e elástico, tornaram-se realidade para provedores e usuários. Nesse cenário, infraestruturas virtuais são provisionadas para hospedar serviços com elevado número de acessos, explorando a elasticidade oferecida pelos provedores para adaptar os recursos computacionais e de comunicação à carga de trabalho submetida, mantendo a qualidade do serviço hospedado. Usualmente, serviços hospedados na nuvem são decompostos em diversas camadas, sendo cada camada individualmente configurável. Uma reconfiguração pode ser iniciada a partir de um pico de processamento, latência elevada na resposta ao usuário final, entre outros indicadores, variando de acordo com as necessidades da aplicação hospedada. Nesse cenário, o presente trabalho propõe um mecanismo para reconfigurar infraestruturas virtuais elásticas, melhorando a relação entre tempo de resposta de uma aplicação n-camadas e o custo de provisionamento do serviço. O algoritmo utiliza o tempo de processamento das requisições submetidas à aplicação como limiar de elasticidade, aumentando o diminuindo o número de máquinas virtuais alocadas. A implementação e análise da solução foi realizada com o simulador de nuvens computacionais CloudSim. A análise experimental indicou uma diminuição no custo de provisionamento combinado com um menor tempo de resposta quando a aplicação é hospedada em uma infraestrutura elástica.

***Palavras-chave*:** Elasticidade, CloudSim, Nuvens Computacionais, IaaS, Servidores Web

Abstract: With the advances introduced by cloud computing, the dynamic and elastic provisioning of virtual resources have became reality for providers and users. In this scenario, virtual infrastructures are allocated to host services capable of supporting a large number of requests, exploiting the elasticity offered by providers to adapt computing and communications resources to the application workload, and simultaneously reducing provisioning costs. Usually, cloud services are decomposed in layers, which are individually configured. In this scenario, a reconfiguration can be started from a workload peak, delay in communication, low response time, among others, being defined by the hosted application goal. The present work extends CloudSim, a cloud computing simulator, implementing an algorithm to explore the elasticity of resources. The algorithm uses requests time processing as a performance indicator. Experimental analysis indicated a low provisioning cost combined with a shorter response time when the application is hosted on an elastic infrastructure.

***Keywords*:** *Elasticity, CloudSim, Cloud Computing, IaaS, Multitiered Applications*

1 Introdução

O paradigma de Computação em Nuvem revolucionou o provisionamento de serviços na Internet ao permitir um acesso ubíquo, conveniente e sob demanda a um conjunto configurável de recursos computacionais compartilhados, que podem ser rapidamente provisionados e liberados com mínimo esforço gerencial [15]. Recentemente, as nuvens computacionais estão consolidadas em várias comunidades acadêmicas, governamentais e industriais devido às facilidades de gerenciamento que foram introduzidas, como agilidade na alocação de recursos, escalabilidade e elasticidade [12]. Sobretudo, as nuvens computacionais foram amplamente difundidas pelo baixo investimento necessário para disponibilizar serviços *on-line*.

[1]Programa de Pós-Graduação em Computação Aplicada (PPGCA), Departamento de Ciência da Computação (DCC), Universidade do Estado de Santa Catarina (UDESC), Joinville (SC) - Brasil
devairdarolt@gmail.com, feliperodrigodesouza@gmail.com, guilherme.koslovski@udesc.br

http://dx.doi.org/10.5335/rbca.v8i2.5414

Dentre os serviços ofertados por provedores de nuvem, o presente trabalho foca em Infraestrutura como Serviço (*Infrastructure as a Service – IaaS*). Este modelo trata do fornecimento dos recursos (processamento, capacidade de armazenamento, entre outros) em sua forma fundamental, através da abstração em máquinas virtuais (MVs) [15]. Em uma infraestrutura virtual (IV), qualquer aplicação pode ser hospedada, sendo que os requisitos de processamento, armazenamento e comunicação são especificados pelos usuários solicitantes. Uma IV pode hospedar sistemas distribuídos complexos, modelados em n-camadas, servindo aplicações SaaS (*Software as a Service*) ou PaaS (*Platform as a Service*). Usualmente, as IVs que hospedam aplicações n-camadas distribuem a carga de trabalho entre subsistemas constituintes, sendo decompostos em balanceadores de carga, servidores web e servidores de banco de dados ou arquivos. Assim, quando uma requisição é submetida às aplicações com esse tipo de arquitetura, os balanceadores de carga identificam quais servidores web estão aptos a receber as requisições, considerando métricas como carga de trabalho atual (número de requisições sendo atendidas), taxa de utilização da CPU, tráfego de rede, utilização da memória, entre outras.

A qualidade do serviço hospedado (QoS – *Quality of Service*) e a qualidade percebida pelo usuário final (QoE – *Quality of Experience*) são métricas resultantes da configuração da IV que hospeda a aplicação [24]. Melhorar a eficiência dos serviços hospedados (QoS ou QoE) é um requisito crucial para empresas e organizações que pretendem atingir um grande público. Seguindo essa motivação, as aplicações n-camadas hospedadas em IVs podem explorar a elasticidade para manter ou melhorar a qualidade do serviço oferecido [6]. Em IaaS, MVs podem ser destruídas quando poucas requisições são enviadas à aplicação, diminuindo o custo necessário para manter a IV, ou agregadas ao serviço quando houver um pico na utilização da aplicação. É latente a necessidade de estudo sobre modelos que permitam o ajuste dinâmico dos recursos computacionais reservados para atender picos de demanda em aplicações hospedadas. Sobretudo, os mecanismos de reconfiguração elástica devem considerar a perspectiva do custo operacional necessário para manutenção do serviço [27]. *O presente trabalho explora a elasticidade dos recursos que compõem uma infraestrutura virtual para otimizar o desempenho de aplicações n-camadas hospedadas. O mecanismo desenvolvido considera a definição de limiares para indicação da qualidade da aplicação hospedada, representando a perspectiva do usuário do serviço. Os limiares são agnósticos à aplicação, sendo definidos pelo usuário contratante durante o estabelecimento do acordo de nível de serviço. Quando os limiares são ultrapassados, MVs são acrescentadas ou removidas à IV. Ainda, um parâmetro é indicado para definir o percentual de elasticidade buscada, evitando um aumento desnecessário no custo de provisionamento.*

Realizar procedimentos de análise, implementação e validação de modelos em cenários de produção induz custos computacionais, financeiros e gerenciais. Embora gerenciadores de nuvens computacionais (*e.g.*, OpenStack, OpenNebula, Eucalyptus) disponibilizem mecanismos para provisionamento elástico, a extensão para implementação do mecanismo e a calibragem inicial dos modelos de elasticidade requerem a composição de protótipos temporários ou a adaptação de ambientes de produção (cenários reais compartilhados por múltiplos usuários). Ainda, protótipos permitem uma experimentação em escala reduzida do modelo. Dessa forma, para analisar a solução proposta, diversos cenários são estudados com o simulador de nuvens computacionais CloudSim [5], em especial a versão com suporte à rede [9]. Em suma, a opção por simulações remete ao baixo investimento necessário para simular configurações de *datacenters* reais, sem a necessidade de contratação de serviços em provedores ou o desenvolvimento de protótipos. A análise experimental compreendeu três cenários envolvendo aplicações n-camadas, variando o número de requisições submetidas ao sistema e o percentual máximo de elasticidade acordado entre usuários e provedores. Os resultados indicaram a diminuição no custo de provisionamento combinado com um menor tempo de resposta quando a aplicação é hospedada em uma infraestrutura elástica. O custo de provisionamento considerado representa o número de recursos computacionais reservados durante um determinado período de tempo. Ainda, os resultados apontam que a definição de um limite máximo para redimensionamento elástico é salutar considerando a relação custo e tempo de resposta.

O restante deste artigo é organizado da seguinte forma: a Seção 2 descreve as diferentes formas de fornecimento de elasticidade, detalhando a elasticidade em aplicações n-camadas. O mecanismo utilizado para a tomada de decisão é descrito na Seção 3. A implementação do mecanismo no simulador CloudSim é descrita na Seção 4. A Seção 5 apresenta os resultados obtidos com as simulações, discutindo perspectivas de trabalhos futuros. Por sua vez a Seção 6 apresenta os trabalhos relacionados, enquanto a Seção 7 conclui o trabalho.

2 A Elasticidade de Recursos em Nuvens Computacionais

Umas das motivações para migração de aplicações para a nuvem, é a possibilidade de provisionamento elástico. A elasticidade é um recurso de gerenciamento ofertado por provedores de nuvens computacionais, permitindo que usuários possam aumentar ou diminuir, de maneira rápida, os recursos computacionais virtuais, em tempo real [6]. Ou seja, uma aplicação hospedada inicialmente com n máquinas virtuais compondo sua IV, pode ser dinamicamente reconfigurada para atender picos de trabalho ou novas demandas, alterando a configuração para $n \pm x$, onde x representam as MVs elásticas (criadas ou destruídas). Assim, uma aplicação elástica é aquela que automaticamente adapta-se ao contexto de execução.

Provedores de nuvem e usuários possuem diferentes visões sobre a elasticidade em nuvens computacionais. Para os provedores, fica a responsabilidade de gerenciar os recursos e realizar alterações nas infraestruturas virtuais provisionadas. Na visão dos usuários, o substrato computacional é totalmente abstraído, sendo acessado por interfaces que facilitam a utilização, permitindo, consequentemente, que o usuário foque no plano de negócio do serviço hospedado. Inclusive, os usuários possuem a percepção de que os recursos computacionais são infinitos [4].

Usualmente, provedores permitem a configuração de elasticidade através de APIs (*Application Programming Interface*), linguagens (*e.g.*, CloudFormation [2]), ou através de mecanismos de reconfiguração dinâmica, como por exemplo Google Compute Engine Autoscaler [3] e Amazon Auto Scaling [4]. No primeiro método (APIs e linguagens), a elasticidade da infraestrutura é chamada de elasticidade manual e fica a cargo dos contratantes do serviço, enquanto no segundo a escalabilidade é realizada através da análise do comportamento computacional da IV. A análise do redimensionamento pode ser realizada com modelos analíticos ou baseados em heurísticas [8]. Os métodos analíticos inferem o comportamento do sistema, porém possuem uma complexidade de desenvolvimento e gerenciamento elevada, principalmente em sistemas com alta variação da carga de trabalho.

Em contrapartida, as heurísticas usualmente diminuem o tempo necessário para a reconfiguração elástica, entretanto, sendo menos eficientes em relação a otimização quando comparadas aos modelos analíticos. Em sua maioria [8], controlam a elasticidade de uma IV através do monitoramento da capacidade da CPU, tráfego de rede, utilização de memória e taxas de entrada e saída. Para as métricas de monitoramento de CPU, normalmente são definidos limiares superiores e inferiores considerando a carga de processamento [17]. Alguns métodos consideram os requisitos da aplicação, distribuindo o sistema através de balanceadores de cargas [22]. Em outras, métricas representam o objetivo de nível de serviço e o acordo de nível de serviço (SLA). O primeiro visa definir os requisitos da aplicação, como disponibilidade e desempenho, enquanto o segundo trata de características mensuráveis como, vazão, tempo de resposta e outras características referente a qualidade.

2.1 Estratégias para Provisionamento de Elasticidade

Em nuvens computacionais, as estratégias de elasticidade podem ser classificadas em replicação (elasticidade horizontal), redimensionamento (elasticidade vertical) e migração. A elasticidade horizontal busca adicionar e remover MVs na infraestrutura virtual do usuário, dando maior escalabilidade ao sistema. Nesse método, imagens de MVs são criadas contendo uma réplica do serviço hospedado. O aumento do número de réplicas ativas diminui o risco de indisponibilidade, pois na ocorrência de uma falta em um ou mais equipamentos, as demais MVs permanecem respondendo as requisições dos usuários. Por sua vez, o método de elasticidade vertical compreende o redimensionamento das capacidades dos recursos provisionados (*e.g.*, CPU, disco, rede e memória). Já a migração é a técnica mais simples de provisionamento de elasticidade e consiste em mover MVs entre hospedeiros físicos distintos [8].

A migração de uma MV segue o mesmo princípio da criação, ou seja, o hospedeiro destinatário deve possuir recursos computacionais suficientes para hospedar a MV migrante, bem como as demais MVs já provisionadas no equipamento. Essa técnica pode trazer vantagens para o usuário final, como por exemplo a minimização da distância entre as MVs [26]. Ao minimizar a distância entre MVs, normalmente a latência de comunicação é diminuída, melhorando o desempenho das aplicações hospedadas [13] [21]. A Figura 1 mostra como o processo de migração pode ser utilizado para redistribuir ou centralizar as MVs em nós computacionais com capacidade

[2] AWS CloudFormation: https://aws.amazon.com/cloudformation/
[3] Google Compute Engine Autoscaler: https://cloud.google.com/compute/docs/autoscaler/v1beta2/.
[4] Amazon Auto Scaling: https://aws.amazon.com/autoscaling/.

compatível. Em caso de centralização (c), os demais nós podem ser desativados, o que consequentemente permite a diminuição de custos administrativos (*e.g.*, reserva de recursos, economia de energia). O contrário também pode ser realizado: as máquinas virtuais podem ser descentralizadas objetivando o aumento da confiabilidade de uma IV [7] [3]. Alguns provedores de nuvens públicas, como a Amazon EC2 [5], Microsoft Windows Azure [6] e Google Cloud Platform [7] exploraram principalmente a elasticidade horizontal. *Seguindo a tendência dos principais provedores de nuvens públicas, o presente estudo tem seu foco voltado para uma combinação de elasticidade horizontal com migração de serviços.*

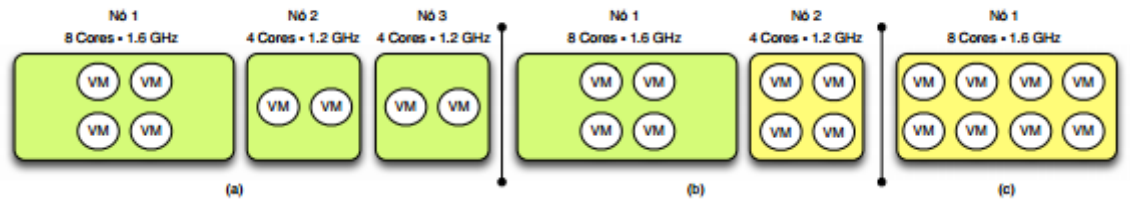


Figura 1: Exemplo de provisionamento de elasticidade com migração de MVs (extraído de [6]). (a) Configuração inicial; (b) Migração de 2 MVs do nó 3 para o nó 2; (c) Migração de todas as MVs para o nó 1.

2.2 Elasticidade em Serviços n-Camadas

Obter eficiência em serviços hospedados na nuvem não é uma tarefa trivial. Corporações e instituições de pesquisa têm investigado diferentes tipos de arquiteturas e tecnologias para composição de aplicações distribuídas, objetivando a disponibilização de um serviço com qualidade para os usuários finais. Algumas arquiteturas de aplicações web são divididas em diversas camadas lógicas (ou subsistemas), caracterizando a arquitetura de n-camadas. Usualmente, são encontradas aplicações com arquiteturas compostas de até 3 camadas, sendo elas i) camada de apresentação; ii) camada lógica; e iii) camada de persistência. Essa arquitetura facilita a divisão da carga de trabalho em grupos especializados, facilitando assim o desenvolvimento, gerenciamento e manutenção da aplicação, pois quando uma das camadas passa por alterações de tecnologia (por exemplo, atualização da linguagem utilizada para desenvolvimento das interfaces, ou alteração na arquitetura do banco de dados) as demais podem permanecer inalteradas.

A camada de apresentação é direcionada especificamente para atender as requisições de usuários. Em geral, os sistemas web possuem nessa camada as páginas contendo formulários e interfaces que facilitam a interação dos usuários com a camada lógica. Por sua vez, a camada lógica, independente das demais camadas, é destinada ao conjunto de rotinas que gerenciam as regras do negócio (por exemplo, vendas, contabilidade, controle de estoque, entre outros). Essa camada realiza a transição dos dados da camada de apresentação para a camada de persistência. Já a camada de persistência tem como objetivo o tratamento dos dados brutos, sendo responsável pelo gerenciamento e armazenamento desses dados de forma que possam ser recuperados por diversos sistemas independentes [14]. Nessa camada é comum encontrarmos bancos de dados e servidores de arquivos. Dessa forma, quando um sistema web, hospedado em uma nuvem elástica, recebe poucas requisições de usuário, é suficiente que somente um servidor por camada esteja ativo. Entretanto, caso diversos usuários passem a acessar o sistema simultaneamente, a arquitetura compreenderá novos recursos computacionais, de modo a atender as novas requisições.

Embora a organização em n-camadas apresente aspectos positivos, encontrar a quantidade ótima de servidores necessários para o fornecimento da aplicação é uma tarefa complexa. Existem pesquisas que buscam obter de forma analítica a quantidade de servidores necessários analisando a viabilidade em relação ao SLA (*Service Level Agreement*) [11] [25]. Isso requer um amplo entendimento e planejamento da aplicação e seus algoritmos. Quando o planejamento é otimista (estima-se que a utilização será menor que a calculada), a capacidade planejada pode ter uma sobrecarga comprometendo o funcionamento da aplicação, violando o SLA. Por outro lado, se o pla-

[5] Amazon EC2: https://aws.amazon.com/pt/ec2.
[6] Microsoft Windows Azure: https://azure.microsoft.com.
[7] Google Cloud Platform: https://cloud.google.com.

nejamento for pessimista (estima-se que a utilização pode ser mais alta que a calculada), recursos desnecessários podem ser reservados, gerando um alto custo de provisionamento.

Para representar as etapas de comunicação e computação realizadas por uma aplicação n-camadas, decompomos a aplicação em balanceadores de cargas, servidores web e servidores de bancos de dados. Especificamente, a Figura 2 demonstra o modelo que representa os estágios de processamento e comunicação necessários para atender uma requisição: (i) inicialmente, o balanceador de carga recebe e envia a requisição para um servidor web, guiando sua decisão por um algoritmo de balanceamento e distribuição de carga; (ii) o servidor web executa um estágio de processamento e (iii) realiza uma requisição ao banco de dados; (iv) o banco de dados executa um estágio de processamento e (v) envia uma quantidade de dados para o servidor web de origem; (vi) os dados recebidos do banco de dados são processados pelo servidor web, que posteriormente (vii) processa e responde a solicitação.

Em aplicações n-camadas, a elasticidade pode ser explorada em múltiplos pontos da IV. Tradicionalmente, o número de servidores web pode ser aumentado ou diminuído de acordo com a carga submetida ao sistema. A ocorrência de uma nova MV acrescida à IV é informada ao balanceador de carga, que consequentemente direciona as requisições de usuários. Ainda, alguns gerenciadores de nuvens computacionais (*e.g.*, OpenStack, Amazon EC2, RackSpace) permitem a reconfiguração de uma MV provisionada, alterando o tipo de sua instância ou reconfigurando parâmetros individuais (*e.g.*, RAM, CPUs virtuais e armazenamento). Um raciocínio similar é realizado com os dispositivos de armazenamento (legenda *Banco de dados* na Figura 2), independente do gerenciador e tecnologia utilizados (*e.g.*, banco de dados relacionais, noSQL), o subsistema de armazenamento (centralizado ou distribuído) por ser aumentado ou diminuído, guiado pelo volume de dados armazenado e número de acessos concorrentes.

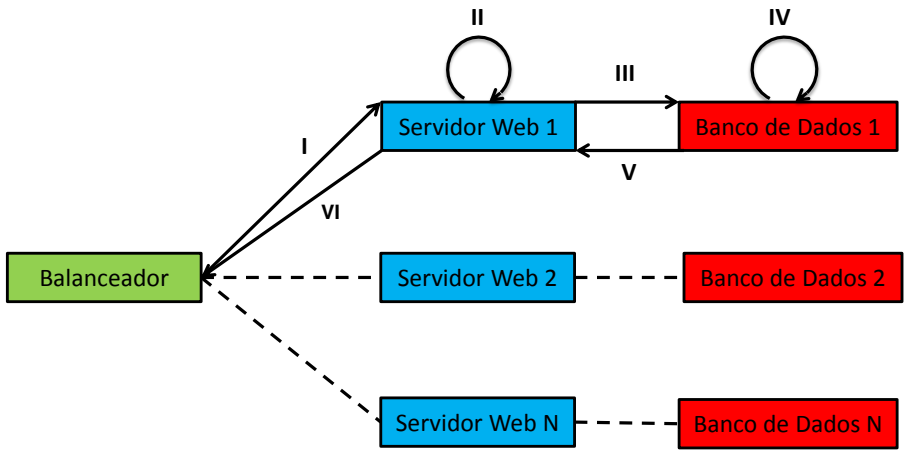


Figura 2: Caminho percorrido por uma requisição submetida à aplicação n-camadas modelada.

3 Mecanismo para Provisionamento de Aplicações n-Camadas Elásticas

O mecanismo para provisionamento e reconfiguração de serviços elásticos proposto explora a combinação de migração e replicação de máquinas virtuais (elasticidade horizontal, conforme discutido na Seção 2.1) para reconfiguração de uma IV, guiado por limiares informados pelo usuário contratante, durante o estabelecimento do acordo de nível de serviço. Desta forma, a solução proposta é agnóstica ao serviço hospedado, desde que este siga uma modelo n-camadas, conforme exemplificado pela Figura 2. Ainda, a escolha do algoritmo empregado pelo balanceador de carga (*e.g*, *round-robin*, max-min) é uma informação relacionada com a aplicação, sendo especificada pelo usuário. A definição de limiares permite a configuração de um serviço reativo, ou seja, quando limiares são atingidos, o mecanismo atua acrescentando ou removendo máquinas virtuais, ou migrando tarefas computacionais.

Embora exista a ilusão de poder computacional infinito, provedores de nuvem determinam uma capacidade máxima permitida para adição de novos recursos. Por exemplo, o provedor Amazon EC2 permite que seus usuários adicionem no máximo 20 instâncias por demanda, enquanto Google Computing Engine limita o número máximo

de operações submetidas à interface de gerenciamento. No mecanismo proposto, o crescimento da IV elástica é limitado por um valor máximo, estudado na análise experimental com diferentes valores, demonstrando a relação obtida entre custo de alocação e desempenho da aplicação hospedada.

A heurística do mecanismo busca a diminuição do tempo médio de processamento das requisições submetidas ao serviço, simultaneamente evitando o aumento do custo de provisionamento. Durante o provisionamento de uma IV, instâncias de MVs podem receber servidores de diferentes tipos (banco de dados ou serviços web) em quantidades variadas. Teoricamente, uma única MV pode suportar completamente a aplicação, reduzindo o custo de provisionamento ao máximo em detrimento do desempenho da aplicação hospedada. A distribuição inicial das MVs para os componentes da aplicação é baseada em uma distribuição proporcional de recursos em arquiteturas n-camadas. Assim, definimos que a quantidade de servidores web (representada por w) é obtida a partir da equação $w = (N - lb)\frac{c}{c+1}$, sendo: N o número máximo de MVs que podem ser provisionadas para o usuário; lb a quantidade de balanceadores de carga; e c a quantidade máxima de servidores que podem ser conectados em um único banco de dados. Por fim, com o número de servidores web definido, a quantidade de bancos de dados necessária é definida por $db = \frac{w}{c}$. Essa abordagem visa uma distribuição eficiente dos recursos computacionais solicitados pelo usuário para hospedar a aplicação distribuída, de modo que cada MV hospede ao menos um serviço que constitui a aplicação, respeitando a quantidade máxima de conexões permitidas por banco de dados e balanceadores de cargas.

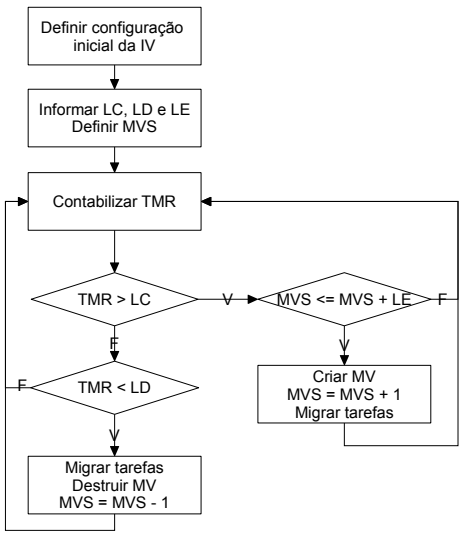


Figura 3: Fluxograma do mecanismo para adaptação elástica das IVs.

A configuração inicial de uma IV pode ser alterada com a migração ou replicação de serviços. A medida que a carga de trabalho aumenta (por exemplo, o número de acessos cresce), novas MVs são alocadas para atender as requisições, e posteriormente é iniciada a migração das tarefas para as novas MVs que estejam com uma menor carga de trabalho. Ainda, se há várias MVs alocadas e estas possuem poucas tarefas em execução, o mecanismo decide pela migração das tarefas restantes, para posteriormente eliminar as MVs, diminuindo o custo de provisionamento final. A Figura 3 exemplifica o mecanismo de adaptação. Inicialmente, a configuração da infraestrutura virtual é definida, identificando os valores iniciais de w, lb e db. O algoritmo é guiado por três limiares para tomada de decisão sobre a criação e destruição de MVs: limiar de criação (LC), limiar de destruição (LD) e limiar de elasticidade (LE). LC e LD representam a variação máxima tolerada pela aplicação hospedada, enquanto LE indica o aumento máximo de recursos previamente autorizado (limitado por provedores ou usuários durante o estabelecimento do acordo de nível de serviço). LC e LD são confrontados com o tempo médio de resposta (TMR) dos serviços hospedados. Quando o valor obtido é maior que o percentual máximo de variação identificado pelo limiar superior, novas MVs são acrescentadas ao ambiente, respeitando, entretanto, o limiar de elasticidade. Uma análise similar ocorre para a destruição de MVs. Assim como proposto em trabalhos anteriores [17], os limiares podem ser ajustados para representar os objetivos das aplicações hospedadas.

4 Implementação do Mecanismo no Simulador CloudSim

O mecanismo descrito na Seção 3 foi implementado como um módulo do simulador CloudSim [5], especificamente da versão com suporte à modelagem de recursos de comunicação [9]. Para modelar uma aplicação distribuída e organizada em n-camadas, estendemos a classe *AppCloudlet*, criando a *WebServerAppMultitier*, que recebe uma lista de MVs na qual são criadas as *cloudlets* (nomenclatura do simulador para representar tarefas). Após a criação dos componentes da aplicação, são submetidas requisições para o sistema. Para representar um cenário estabilizado, algumas requisições são submetidas antes do algoritmo de elasticidade entrar em execução. Assim, antes da simulação ser iniciada, todos os estágios de comunicação do serviço hospedado já estão organizadas na forma de máquinas de estado (conforme ilustrado na Figura 2). Os algoritmos descritos anteriormente foram implementados no $broker$ do simulador. No CloudSim, quando há mais de uma MV em um mesmo hospedeiro físico, o *framework* realiza a execução em modo paralelo, permitindo que uma MV que esteja executando em uma vCPU não tenha seu desempenho comprometido por MVs executando em outras vCPUs. Já o escalonador de *cloudlets* executa no modo de tempo compartilhado: quando existe mais de uma *cloudlet* executando em uma mesma MV, a execução é enfileirada de modo que uma necessite aguardar o término da anterior.

Os Algoritmos 1 e 2 apresentam pseudocódigos do mecanismo proposto (fluxograma da Figura 3). O primeiro descreve a seleção de *cloudlets* (tarefas que compõem a aplicação n-camadas hospedada) para migração. Para evitar o reprocessamento de respostas já iniciadas, optamos por não migrar serviços com comunicação em andamento, ou seja, qualquer serviço aguardando ou enviando uma mensagem síncrona não será selecionado como potencial candidato para migração. Ainda, *cloudlets* finalizando seu processamento não são selecionadas, pois o tempo de migração é usualmente superior ao tempo de processamento remanescente. O Algoritmo 2 evidência que a seleção das *cloudlets* e das MVs candidatas à migração ocorre em momentos distintos. Enquanto o Algoritmo 1 seleciona as *cloudlets*, os métodos $overloadedVM()$ e $underloadedVM()$ selecionam as MVs considerando a carga de processamento (comparando o percentual de uso de CPU). A variação máxima tolerada pela aplicação hospedada é representada por $deltaAverageTime$ no Algoritmo 2, enquanto o crescimento da IV elástica é limitado por um valor máximo, representado pelo parâmetro $elasticity$.

Algorithm 1: $findCloudletsToMigrate$: pseudocódigo para encontrar as *cloudlets* que podem ser migradas.

```
1  cloudlets: serviços que podem ser migrados;
2  for mv ∈ IV do
3      for cl ∈ cloudlets(mv) do
4          if cl is sending or waiting for a packet then
5              continue;
6          if cl is finished then
7              continue;
8          if cl is almost finished then
9              continue;
10         add cl to cloudlets;
11 return cloudlets;
```

5 Análise Experimental

Para estudar a aplicabilidade da solução proposta, uma análise experimental foi conduzida com o simulador de nuvens computacionais CloudSim. Os experimentos foram realizados em um computador AMD Phenom II com 4 GB de RAM e Linux Ubuntu 12.04.02. O simulador foi executado com o Java versão 1.7.0_09. Seguindo trabalhos anteriores [21], cada hospedeiro físico foi modelado de forma padronizada com 16 GB de RAM e 16 CPUs, interconectados em uma topologia *fat-tree* [1] com largura de banda de 1 Gbps entre os *switches* de borda e agregação, e 10 Gbps entre os *switches* de agregação e de núcleo. A topologia *fat-tree* foi selecionada devido a ampla adoção em provedores de nuvens computacionais [24]. A Figura 4 apresenta a simplificação de uma

Algorithm 2: $elasticProvisioning$: Pseudocódigo para determinar a criação ou destruição de MVs a partir do tempo de processamento das requisições.

```
1  elasticity: percentual máximo de MVs elásticas;
2  runningVMs: MVs ativas;
3  deltaAverageTime = updateRuntimeAverage();
4  cloudletsToMigrate = findCloudletsToMigrate();
5  if cloudletsToMigrate is ∅ then
6      return;
7  if (deltaAverageTime ≥ l_create) and (runningVMs ≤ runningVMs + elasticity) then
8      src = overloadedVM();
9      dst = createVMs();
10     migrateCloudlets(cloudletsToMigrate, src, dst);
11     runningVMs = updateRunningVMs(dst);
12 else if (deltaAverageTime ≤ l_destroy) then
13     src = vmToDestroy();
14     dst = underloadedVM();
15     migrateCloudlets(cloudletsToMigrate, src, dst);
16     destroyVM(src);
```

topologia *fat-tree*. No presente trabalho, a topologia foi definida com 8 *pods*.

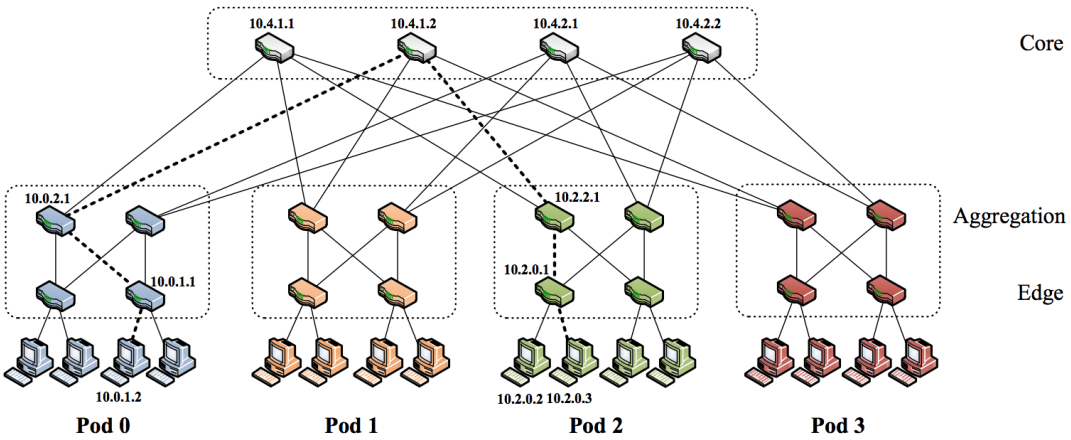


Figura 4: Simplificação de uma topologia *fat-tree* composta por 4 *pods*. O exemplo descreve a organização dos *switches* de borda (*edge*), agregação (*aggregation*) e núcleo (*core*) [1].

As configurações das máquinas virtuais reservadas para hospedar a aplicação com arquitetura n-camadas seguiram uma distribuição uniforme entre configurações pré-determinadas. Em suma, a configuração de RAM foi selecionada entre 1, 2 e 4 GB, enquanto as CPUs virtuais foram selecionadas entre 1, 2, 4 e 8 núcleos alocados por MV. Quanto aos enlaces de comunicação, a capacidade requisitada foi selecionada entre 50%, 25% e 12,5% da menor capacidade ponto-a-ponto (a saber, 1 Gbps) da topologia física. A alocação inicial da IV foi realizada seguindo o algoritmo de Alocação Orientada por Distância (AOD) [21], que busca a diminuição da distância entre os recursos provisionados.

Para cada cenário, a IV foi inicialmente composta por 5 máquinas virtuais seguindo a distribuição de serviços discutida na Seção 3 (resultando em 1 balanceador de carga, 3 servidores web e um banco de dados). Optamos pelo algoritmo *round-robin* como padrão para distribuição das requisições recebidas pelo balanceador de carga. O uso de carga sintética contempla uma variada gama de situações de configuração de MVs, analisando a aplicabilidade da solução proposta com diferentes configurações de aplicações n-camadas hospedadas na IV.

Os parâmetros de entrada dos cenários de teste representam serviços web n-camadas usualmente hospedados em provedores de nuvem. Três cenários de testes, com cargas sintéticas, foram analisados variando o número máximo de recursos elásticos: i) sem elasticidade (identificado nos gráficos pela legenda 0%); ii) 100% de elasticidade; e iii) e o terceiro com 200% de elasticidade permitida. Cada cenário representa um possível acordo de nível de serviço estabelecido entre o usuário contratante e o provedor de serviço. O percentual de elasticidade (0, 100 ou 200) indica o crescimento máximo acordado do número de máquinas virtuais (parâmetro $elasticity$ do Algoritmo 2). A escolha dos percentuais máximos de elasticidade foi baseada na configuração padrão do provedor de nuvem pública Amazon EC2. Neste provedor, cada usuário pode solicitar no máximo 20 instâncias de MVs, ou seja, um crescimento máximo de 200% sobre a configuração inicial indicada (5 MVs).

Inicialmente, 100 requisições foram submetidas para colocar o sistema em regime estável. Uma requisição representa um acesso ao serviço hospedado na nuvem computacional, sendo que seu processamento segue o percurso apresentado na Figura 2. O tempo de processamento de uma *cloudlet* foi selecionado de forma uniforme entre 0,1 a 4 segundos (o tempo de processamento dos estágios representados na Figura 2). Já o tempo de comunicação entre os recursos é guiado pelo volume de tráfego atual.

Buscando a simulação de picos de execução, foram adicionadas aleatoriamente entre 100 e 500 requisições. Para o cenário sem elasticidade, as MVs não podem ser destruídas e novas MVs não podem ser criadas. Entretanto, nos cenários com 100% e 200% de provisionamento elástico, as MVs são removidas (quando ociosas) ou dinamicamente criadas até um limite de 100% (ou 200%) da quantidade inicialmente informada pelo usuário, aplicando os Algoritmos 1 e 2. A definição dos limiares indicativos de saturação e ociosidade de um sistema não é uma tarefa trivial. A quantificação da qualidade observada pelo usuário final requer um mecanismo de monitoração na camada do usuário, sendo dependente da aplicação em questão. Baseado em observações experimentais, os limiares l_{create} e $l_{destroy}$ foram definidos como 20% e 50%, respectivamente, para a aplicação analisada. Ou seja, os limiares representam o melhor caso de configuração possível para a carga submetida.

A análise experimental quantificou duas métricas: o custo total de provisionamento e o tempo médio de processamento das requisições. A primeira representa o investimento necessário para o provisionamento completo da IV enquanto a segunda representa a visão dos usuários do serviço hospedado. O custo de provisionamento foi definido como o somatório da capacidade de processamento reservado durante o período de provisionamento. Para realizar uma análise comparativa, o custo de provisionamento sem elasticidade foi considerado como linha de base, sendo comparado com as demais configurações. Em cada cenário foram realizadas 100 execuções, e as barras de erros nos gráficos representam o intervalo de confiança de 95%.

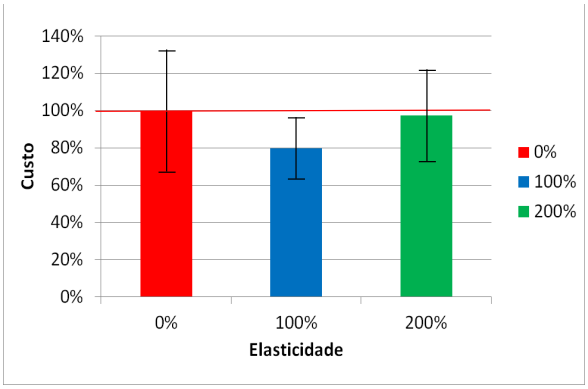
5.1 Custo Total de Provisionamento

A Figura 5 apresenta o custo total de provisionamento para os cenários avaliados. Para o cenário sem elasticidade (0%), o custo referente ao tempo de processamento realizado pelas 5 MVs inicialmente provisionadas para atender as requisições é utilizado como base para comparações. Segundo os resultados apresentados na Figura 5, as configurações elásticas obtiveram custo menor ou similar em todos os cenários avaliados. Especificamente, a configuração com 100% de elasticidade máxima obteve o menor custo total de provisionamento. Essa situação reflete um cenário real vivenciado em nuvens computacionais: o aumento dos recursos computacionais alocados nem sempre resulta um uma melhora no serviço disponibilizado. Nesse caso, acrescentar mais MVs à IV representa uma maior distribuição dos processos. Ou seja, a ocorrência de gargalos de comunicação nos servidores ou equipamentos comunicantes pode depreciar o serviço.

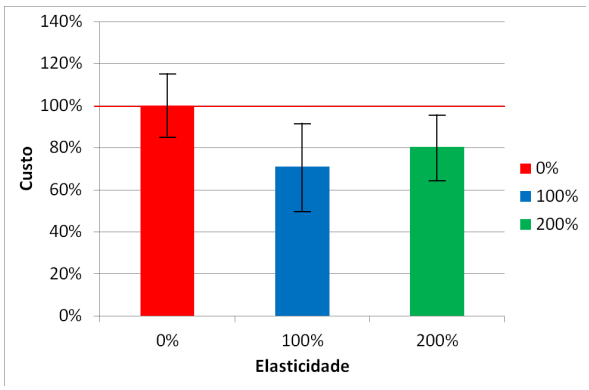
Quanto ao percentual de redução de custos, a configuração com 200% de elasticidade reduziu entre aproximadamente 5% e 20% nos cenários analisados, enquanto uma configuração com 100% de elasticidade máxima obteve uma redução de custos com valores entre 18% e 42%. Especificamente, no cenário com maior número de requisições submetidas (Figura 5(e)) o uso do algoritmo elástico de reduziu aproximadamente 1,7 vezes o custo total de provisionamento.

5.2 Tempo Médio de Processamento de Requisições

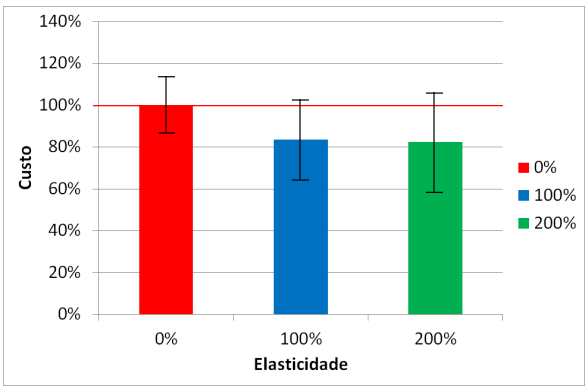
O tempo médio de processamento das requisições submetidas é apresentado na Figura 6 para os cenários avaliados. Para 100 requisições (Figura 6(a)) a aceleração obtida no tempo médio de processamento com uso de



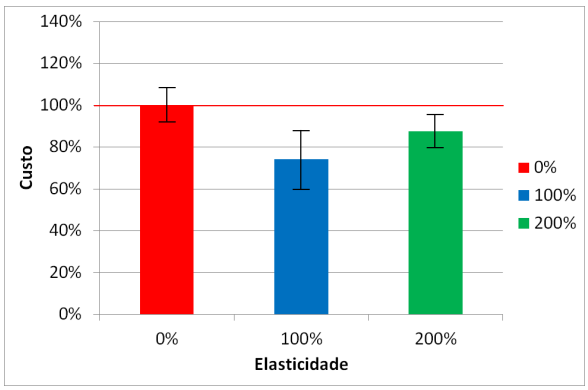
(a) Cenário com 100 requisições.



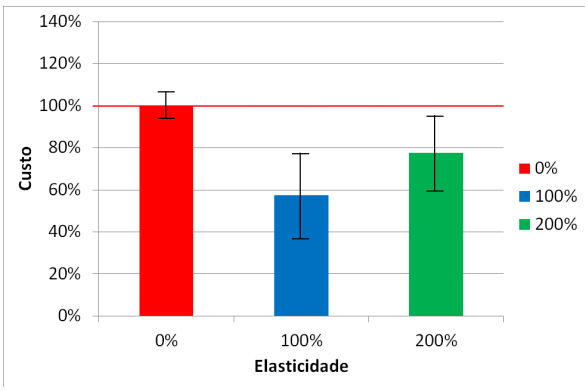
(b) Cenário com 200 requisições.



(c) Cenário com 300 requisições.



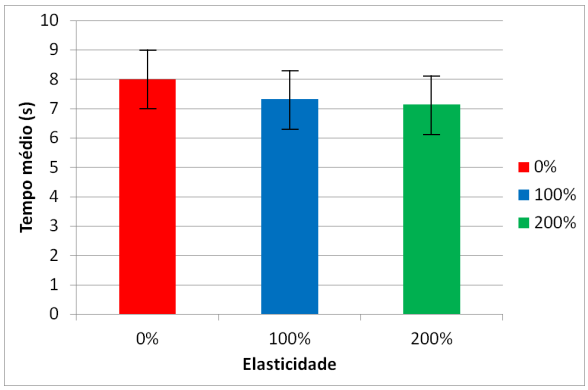
(d) Cenário com 400 requisições.



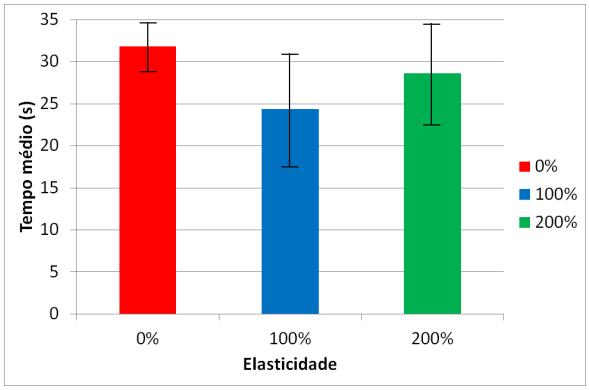
(e) Cenário com 500 requisições.

Figura 5: Custo total de provisionamento das MVs alocadas para o usuário.

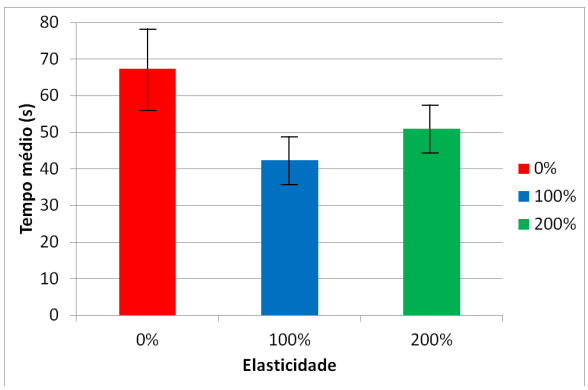
provisionamento elástico é baixa. Entretanto, para os demais cenários, a configuração com 100% de elasticidade máxima acelerou o tempo médio de processamento em 1,3, 1,6, 1,77 e 5 vezes para 200, 300, 400 e 500 requisições, respectivamente. Comparando o provisionamento estático (0%) com os modelos elásticos é evidente a redução do tempo de processamento oriundo da aplicação do algoritmo de elasticidade.



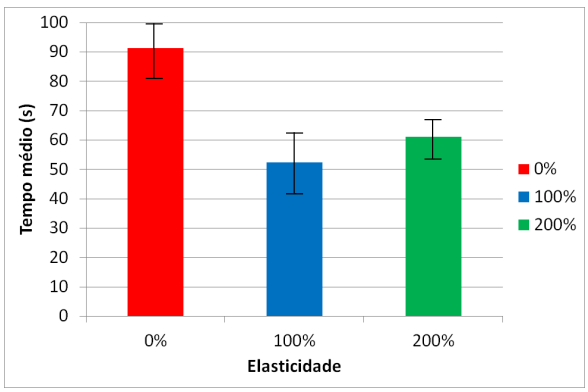
(a) Cenário com 100 requisições.



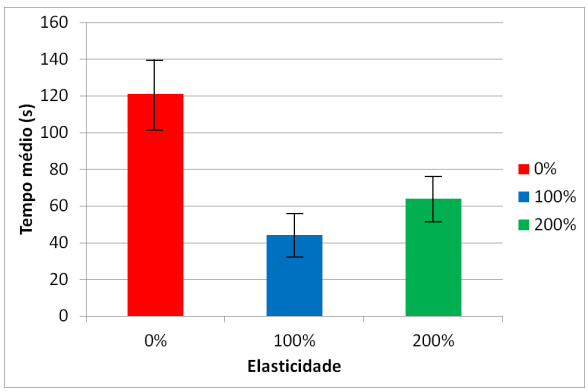
(b) Cenário com 200 requisições.



(c) Cenário com 300 requisições.



(d) Cenário com 400 requisições.



(e) Cenário com 500 requisições.

Figura 6: Tempo médio de processamento das requisição submetidas.

5.3 Discussão e Considerações

Analisando conjuntamente as métricas, é possível visualizar que um provisionamento com 100% de elasticidade máxima é superior aos cenários sem elasticidade e com elasticidade máxima de 200%, tanto em relação ao custo total (recursos provisionados durante o período de execução), quanto em relação ao tempo médio de resposta das requisições. Ou seja, nem sempre um maior investimento em recursos computacionais (configurações com 200% de elasticidade máxima) resultará na melhora do serviço ofertado.

A configuração de ambientes elásticos não é uma tarefa trivial. Existem situações em que as métricas de uso

de recursos disponibilizadas pelos provedores não são suficientes para representar a visão do usuário do sistema hospedado. Por exemplo, a aplicação do algoritmo proposto neste trabalho em um ambiente de produção requer o monitoramento constante do serviço hospedado. Atualmente, provedores de nuvem disponibilizam métricas referentes ao consumo das máquinas virtuais e de rede, deixando a monitoração da aplicação sob responsabilidade do usuário contratante. A instrumentação de uma aplicação n-camadas para experimentação em um cenário de produção é apontada como perspectiva de continuação. Por fim, o provisionamento elástico de recursos de comunicação (*e.g.*, reserva de largura de banda, reconfiguração de caminhos) é uma linha futura promissora, já que a topologia de rede pode ser um fator limitado para o desempenho da aplicação hospedada.

6 Trabalhos Relacionados

A literatura especializada compreende trabalhos atrelados ao provisionamento elástico de recursos e a simulação de nuvens computacionais.

Trabalhos relacionados com a elasticidade de infraestruturas virtuais vem sendo desenvolvidos com o objetivo de minimizar os custos para hospedagem de aplicações em nuvem tanto para clientes quanto para provedores, mantendo em paralelo, a qualidade do serviço ofertado [8]. Esses trabalhos focam no monitoramento de métricas como carga de processamento, volume de dados transferidos e capacidade da memória, delimitadas por limiares para prover a elasticidade [17] [18] [19] [16]. Ainda, a predição do uso futuro de recursos virtuais tem sido utilizada para tomada de decisão em ambientes elásticos [10] [20]. Complementarmente, provedores de nuvens computacionais definem APIs e modelos para configuração de elasticidade, como por exemplo AWS CloudFormation, Google Compute Engine Autoscaler e Amazon Auto Scaling.

Em suma, os trabalhos revisados objetivam o redimensionamento dos recursos computacionais guiado por métricas tradicionais, referentes ao consumo computacional (utilização de CPU, volume de dados transferido, entre outros). Embora eficientes, essas métricas não representam a visão do usuário utilizador do serviço hospedado. Assim, o presente trabalho complementa o estudo introduzindo um algoritmo elástico guiado pelo tempo de processamento da aplicação, ou seja, o tempo de resposta aguardado pelo usuário solicitante. Essa métrica representa uma visão combinada do uso dos recursos computacionais. A elasticidade não é guiada apenas por valores absolutos de consumo de recursos, mas pelo seu impacto no tempo de processamento da aplicação.

Quanto a simulação, as pesquisam buscam a experimentação de novas tecnologias voltadas para tais ambientes e a identificação de novas oportunidades de pesquisa. Dentre os simuladores de nuvens computacionais identificados, o CloudSim [5] é um simulador gratuito, que facilita o estudo de algoritmos e metodologias através da extensão de classes desenvolvidas em Java. Por apresentar tais oportunidades, o *framework* vem sendo incrementado ao longo dos anos [2] , permitindo simulações cada vez mais elaboradas e representativas. Em [9] foi introduzida uma representação inicial da topologia de rede de um provedor de nuvem. Essa implementação foi adaptada para representar a topologia *fat-tree* discutida na análise experimental. Recentemente, o CloudSimSDN [23] foi proposto para simular no CloudSim ambientes com redes definidas por software, apresentando um novo modelo de representação da infraestrutura, bem como incluindo ferramentas para testes e controle de políticas de alocação baseadas em fluxo. Uma possível combinação da reconfiguração introduzida por CloudSimSDN com o algoritmo de elasticidade proposto nesse trabalho é indicada como perspectiva de continuidade do estudo para investigar o provisionamento elástico dos recursos de comunicação em uma nuvens computacionais.

7 Conclusão

O conceito de nuvens computacionais foi amplamente difundido na comunidade acadêmica e comercial. Dentre os fatores motivadores, destaca-se a elasticidade dos recursos computacionais. Melhorar a eficiência dos serviços disponibilizados é um requisito crucial para diversas empresas e organizações que pretendem atingir um grande público. Através da elasticidade, MVs podem ser destruídas quando poucas requisições são enviadas à aplicação, ou agregadas ao serviço quando houver picos de utilização da aplicação. Entretanto, dificilmente os mecanismos para provisionamento de elasticidade consideram a visão dos usuários solicitantes do serviço.

Nesse contexto, este trabalho propôs um mecanismo baseado no tempo médio de resposta das requisições recebidas por uma aplicação distribuída e organizada em camadas. Quando o tempo médio de resposta ultrapassa

um limiar representativo da qualidade percebida pelo usuário, o mecanismo acrescenta máquinas virtuais à infraestrutura virtual, posteriormente migrando tarefas críticas (com elevado tempo de processamento). De forma similar, quando um limiar de ociosidade é identificado, tarefas são consolidadas em máquinas virtuais, permitindo a desativação de instâncias ocioasas.

Para análise do mecanismo, o *framework* CloudSim foi estendido, acrescentando a heurística proposta e modelando cenários representativos de provedores de nuvem e aplicações web n-camadas. A análise experimental indicou a diminuição do custo de provisionamento combinado com um menor tempo de resposta quando a aplicação é hospedada em uma infraestrutura elástica. Em relação a trabalhos futuros, as heurísticas podem levar em consideração a utilização dos recursos de comunicação, bem como a origem das requisições recebidas.

Agradecimentos

Os autores gostariam de agradecer a UDESC pelo auxílio financeiro e ao Laboratório de Processamento Paralelo e Distribuído (LabP2D) pela disponibilização dos recursos computacionais para a realização do trabalho.

Referências

[1] Mohammad Al-Fares, Alexander Loukissas, and Amin Vahdat. A scalable, commodity data center network architecture. *SIGCOMM Comput. Commun. Rev.*, 38(4):63–74, August 2008.

[2] Cássio P. Alkmin and Daniel Cordeiro. SimMyCloud, simulando o gerenciamento de recursos virtualizados em plataformas de computação em nuvem. In *Salão de Ferramentas - SBRC 2014*, Maio 2014.

[3] Peter Bodík, Ishai Menache, Mosharaf Chowdhury, Pradeepkumar Mani, David A Maltz, and Ion Stoica. Surviving Failures in Bandwidth-Constrained Datacenters. In *Proc. ACM SIGCOMM*, pages 431–442, 2012.

[4] Rajkumar Buyya, Chee Shin Yeo, and Srikumar Venugopal. Market-oriented cloud computing: Vision, hype, and reality for delivering it services as computing utilities. In *Proc. of the 10th IEEE Int. Conf. on High Performance Computing and Communications*, HPCC '08, pages 5–13, Washington, DC, USA, 2008. IEEE Computer Society.

[5] Rodrigo N. Calheiros, Rajiv Ranjan, Anton Beloglazov, Cesar A. F. De Rose, and Rajkumar Buyya. Cloudsim: A toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms. *Softw. Pract. Exper.*, 41(1):23–50, January 2011.

[6] Rodrigo da Rosa Righi. Elasticidade em cloud computing: conceito, estado da arte e novos desafios. *Revista Brasileira de Computação Aplicada (RBCA)*, 5(2):2–17, 2013.

[7] G.A. De S Cavalcanti, R.R. Obelheiro, and G. Koslovski. Optimal resource allocation for survivable virtual infrastructures. In *10th International Conference on the Design of Reliable Communication Networks*, 2014.

[8] G. Galante and L.C.E. de Bona. A survey on cloud computing elasticity. In *Utility and Cloud Computing (UCC), 2012 IEEE Fifth International Conference on*, pages 263–270, Nov 2012.

[9] S.K. Garg and R. Buyya. Networkcloudsim: Modelling parallel applications in cloud simulations. In *Utility and Cloud Computing (UCC), 2011 Fourth IEEE International Conference on*, pages 105–113, Dec 2011.

[10] Zhenhuan Gong, Xiaohui Gu, and J. Wilkes. Press: Predictive elastic resource scaling for cloud systems. In *Network and Service Management (CNSM), 2010 International Conference on*, pages 9–16, Oct 2010.

[11] Rui Han, Moustafa M. Ghanem, Li Guo, Yike Guo, and Michelle Osmond. Enabling cost-aware and adaptive elasticity of multi-tier cloud applications. *Future Gener. Comput. Syst.*, 32:82–98, March 2014.

[12] Brendan Jennings and Rolf Stadler. Resource management in clouds: Survey and research challenges. *Journal of Network and Systems Management*, pages 1–53, 2014.

[13] G. Koslovski, S. Soudan, P. Goncalves, and P. Vicat-Blanc. Locating virtual infrastructures: Users and inp perspectives. In *Integrated Network Management (IM), 2011 IFIP/IEEE International Symposium on*, pages 153–160, May 2011.

[14] Claudia Ferreira Gonzalez Llana and Márcio Francisco Campos. Identificando atividades para apoiar o desenvolvimento de sistemas para a web. *Anais do Curso de Ciência da Computação. Volume III*, 22230:5, 2003.

[15] Peter M. Mell and Timothy Grance. Sp 800-145. the nist definition of cloud computing. Technical report, NIST, Gaithersburg, MD, United States, 2011.

[16] Fabio Morais, Francisco Brasileiro, Raquel Lopes, Ricardo Araujo, Augusto Macedo, Wade Satterfield, and Leandro Rosa. Um arcabouço para provisionamento automática de recursos em provedores de iaas independente do tipo de aplicação. In *Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos - SBRC*, Brasília, Brasil, May 2013.

[17] R. J. ; PFITSCHER, M. A. PILLON, and R. R. OBELHEIRO. Diagnóstico do provisionamento de recursos para máquinas virtuais em nuvens iaas. In *31o Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos - SBRC*, Brasília, Brasil, May 2013.

[18] Ricardo J. Pfitscher, Mauricio A. Pillon, and Rafael R. Obelheiro. Customer-oriented diagnosis of memory provisioning for iaas clouds. *SIGOPS Oper. Syst. Rev.*, 48(1):2–10, May 2014.

[19] R. Righi, V. Rodrigues, C. Andre daCosta, G. Galante, L. Bona, and T. Ferreto. Autoelastic: Automatic resource elasticity for high performance applications in the cloud. *Cloud Computing, IEEE Transactions on*, PP(99):1–1, 2015.

[20] N. Roy, A. Dubey, and A. Gokhale. Efficient autoscaling in the cloud using predictive models for workload forecasting. In *Cloud Computing (CLOUD), 2011 IEEE International Conference on*, pages 500–507, July 2011.

[21] Denivy B Ruck, Ramon Oliveira, and Guilherme P Koslovski. Comparação de algoritmos para alocação de Infraestruturas Virtuais. *Revista Brasileira de Computação Aplicada*, Oct 2014.

[22] Rhodney Simões and Carlos Kamienski. Gerenciamento de elasticidade em nuvens privadas e híbridas. In *XII Workshop de Computação em Clouds e Aplicações - WCGA*, Florianópolis, Brasil, May 2014.

[23] Jungmin Son, A.V. Dastjerdi, R.N. Calheiros, Xiaohui Ji, Young Yoon, and R. Buyya. Cloudsimsdn: Modeling and simulation of software-defined cloud data centers. In *Cluster, Cloud and Grid Computing (CCGrid), 2015 15th IEEE/ACM International Symposium on*, pages 475–484, May 2015.

[24] William Stallings. *Foundations of Modern Networking: SDN, NFV, QoE, IoT, and Cloud*. Addison-Wesley Professional, 1st edition, 2015.

[25] Tram Truong Huu, Guilherme Koslovski, Fabienne Anhalt, Johan Montagnat, and Pascale Vicat-Blanc Primet. Joint elastic cloud and virtual network framework for application performance-cost optimization. *Journal of Grid Computing*, 9(1):27–47, 2011.

[26] William Voorsluys, James Broberg, Srikumar Venugopal, and Rajkumar Buyya. Cost of virtual machine live migration in clouds: A performance evaluation. In *Proceedings of the 1st International Conference on Cloud Computing*, CloudCom '09, pages 254–265, Berlin, Heidelberg, 2009. Springer-Verlag.

[27] L. Wu, S. K. Garg, and R. Buyya. Sla-based resource allocation for software as a service provider (saas) in cloud computing environments. In *Cluster, Cloud and Grid Computing (CCGrid), 2011 11th IEEE/ACM International Symposium on*, pages 195–204, May 2011.