Elasticidade em cloud computing: conceito, estado da arte e novos desafios

Rodrigo da Rosa Righi[1]

Resumo: A elasticidade é, sem dúvida, uma das características mais marcantes da computação em nuvem, sendo um diferencial desse tipo de sistema distribuído em relação a outros, como grades computacionais e *peer-to-peer*. Com base nos paradigmas de computação sobre demanda e pague-pelo-que-use, é possível dinamicamente aumentar ou diminuir instâncias de máquinas virtuais e/ou nós de computação, bem como aplicar reconfigurações de percentagem de CPU, memória e largura de banda de rede relativos a um serviço em nuvem. Além dos evidentes benefícios de custo e desempenho para o usuário, o provedor da nuvem tem a vantagem de oferecer um melhor uso dos recursos aos seus usuários. Nesse contexto, este artigo apresenta o estado da arte na área de elasticidade em nuvem, enfatizando desde a abordagem padrão que usa transações web até iniciativas para a computação de alto desempenho. Ainda, o texto discute sobre métricas para ativação da elasticidade, o seu nível de atuação (SaaS, PaaS ou IaaS), bem como a interface de uso (sem intervenção do usuário, linha de comando, ferramenta gráfica ou diretivas de programação). Para fins de experimentação, um estudo de caso do emprego da elasticidade em aplicações de alto desempenho sobre o *middleware* OpenNebula é apresentado e discutido. Por fim, o artigo aponta os desafios na área e oportunidades de pesquisa, tanto no cunho das nuvens privadas quanto no das públicas.

***Palavras-chave*:** Computação em nuvem. Elasticidade. Gerência de recursos. Virtualização.

Abstract: *Elasticity is undoubtedly one of the most striking features of cloud computing, appearing as a differential of such kind of distributed system in comparison with others like grid computing and peer-to-peer. Concerning both the paradigms of on-demmand computing and pay-as-you-go, elasticity offers the ability to dynamically increase or decrease the number of instances or computing nodes, as well as to reconfigure CPU, memory and network bandwidth data for a cloud service. Besides the obvious benefits of cost and performance for the user, the cloud provider also has the advantage of providing a better use of resources for the subscribers. In this context, this paper presents the state-of-art in the area of cloud elasticity, emphasizing since the standard approach which uses Web transactions to research initiatives on high performance computing. Furthermore, the article discusses about metrics for elasticity activation, its level of activity (SaaS, PaaS or IaaS), as well as the user interface (without user intervention, command line, graphical tool or application programming interface). For purposes of experimentation, we present a case study considering the elasticity topic applied to high performance applications. Finally, the article discusses the challenges and research opportunities in the area, both in the deployment of private and public clouds.*

***Keywords*:** *Cloud computing. Elasticity. Resource provisioning. Virtualization.*

1 Introdução

Até meados da década de 1970, os computadores existentes eram apenas de grande porte (Mainframes) mantidos em ambientes controlados e acessados à distância. Ao final dos anos 70 e início dos anos 80 surgiram os microcomputadores, cuja adoção foi impulsionada pelo advento das redes ethernet e da transição da internet para uso civil. Ao final da década de 80, estavam colocados os elementos para o desenvolvimento de sistemas cliente/servidor. Diferentemente de um terminal com poder de processamento nulo ou exíguo da década de 70,

[1]Prof. do Programa Interdisciplinar de Pós-Graduação em Computação Aplicada - Universidade do Vale do Rio dos Sinos - São Leopoldo - RS
Contato: rrrighi@unisinos.br

http://dx.doi.org/10.5335/rbca.2013.3084

o cliente em questão pode ser definido como um computador ou uma aplicação que executa certa quantidade de processamento de forma autônoma. Além disso, ele tem a capacidade de enviar requisições para um ou mais servidores para execução de processamentos ou solicitação de dados [1]. A evolução da arquitetura cliente/servidor, no início dos anos 2000, foi marcada pela adoção de arcabouços para programação com objetivos distribuídos, uso de Web Services e a consolidação e o surgimento de arquiteturas como clusters e grades computacionais. Em especial, as grades destacam-se por formarem organizações virtuais que usam o poder de instituições ao redor do mundo para a colaboração entre pesquisadores e cálculo de aplicações científicas [2, 3].

Uma análise de características comuns nas arquiteturas cliente/servidor e grades computacionais permite a identificação dos seguintes pontos: (i) número de pessoas necessárias para tornar o sistema distribuído operacional; (ii) perícia requerida para resolver problemas técnicos de hardware e instalação de software; (iii) servidores e equipamentos de rede podem ter preços elevados, e uma configuração pertinente num determinado momento pode ficar defasada em outros logo adiante; (iv) a falha em um servidor pode ocasionar a indisponibilidade do sistema; (v) o parque de servidores normalmente existe em número limitado, tornando fixa a carga máxima suportada pelo serviço ou pela aplicação. Tais pontos são mitigados com o fortalecimento da computação em nuvem, ou *cloud computing*, na última década [4]. A nuvem em si não é um conceito novo, mas sim o ressurgimento do paradigma da computação como utilitário (*utility computing*) [5]. Em linhas gerais, isso se deve aos avanços das técnicas de virtualização e do modelo de negócio pague-pelo-que-use inerente à computação em nuvem. Um dos conceitos de nuvem mais respeitados na comunidade científica é o do *National Institute of Standards and Technology* (NIST), que assim a define[2]:

Infraestrutura com capacidades de aprovisionamento de recursos de forma rápida e elástica, em alguns casos automática, para aumentar e diminur o número de recursos. Para o usuário, tais capacidades muitas vezes parecem ser ilimitadas, podendo ser realizadas em qualquer quantidade e a qualquer momento.

A definição do NIST enfatiza uma das características mais marcantes da nuvem, que é a capacidade de elasticidade de recursos sob demanda, de acordo com o comportamento do serviço. Em especial, ela ataca o ponto (v) citado previamente. Por exemplo, novas máquinas físicas e/ou virtuais podem ser adicionadas para suprir picos de requisições durante um período do dia ou irregularidades de processamento ou E/S em aplicações de alto desempenho. A Figura 1 ilustra o comportamento de um serviço com uma alocação de carga estática e outra dinâmica, ou elástica. A alocação estática tenta provisionar recursos sempre para o pior caso (momentos de pico), correndo o risco de essa previsão ser subestimada, como mostra a Figura 1 (a) em dois pontos. Ainda, há a questão do desperdício de recursos, dado que momentos de pico não caracterizam a carga média no sistema. Por outro lado, a alocação elástica de recursos é dependente do comportamento do serviço (ou aplicação), das métricas de monitoramento e do nível de serviço (SLA) contratado. Isso traz benefícios tanto para o usuário quanto para o administrador da nuvem, pois o primeiro consegue manter o desempenho de seu serviço em execução em níveis aceitáveis, pagando um preço adequado para isso, enquanto o segundo consegue um melhor uso dos recursos da infraestrutura, uma vez que aqueles liberados por um usuário podem ser repassados para outro sob demanda.

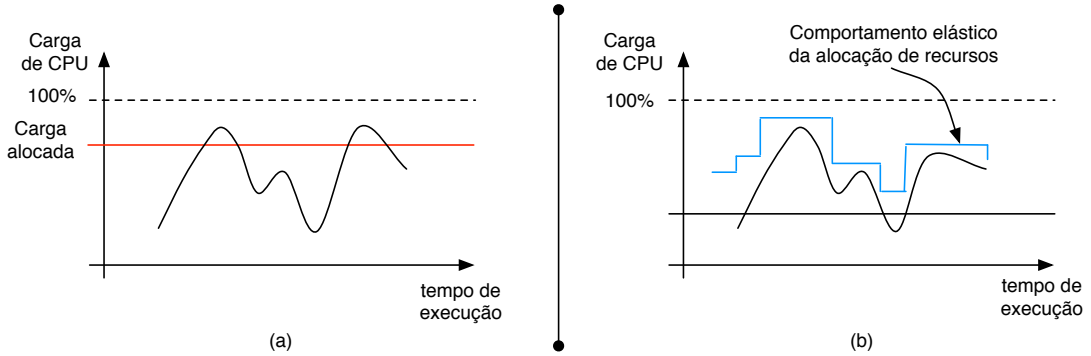

Figura 1: Tratamento da computação em nuvem: (a) carga do serviço ultrapassa a alocada, comprometendo desempenho e funcionalidade; (b) tratamento elástico da carga de acordo com o comportamento do serviço

O conceito de elasticidade vai ao encontro da computação verde. Assim como é possível aumentar o número

[2] http://csrc.nist.gov/publications/drafts/800-145/Draft-SP-800-145_cloud-definition.pdf

de recursos, eles também podem ser reduzidos de modo que a interrupção ou migração de máquinas virtuais (MVs) possa gerar o desligamento da energia de equipamentos físicos. Ainda com essa potencialidade, estudos revelam que os principais usos da nuvem estão relacionados com economia financeira na compra de recursos e balanceamento dinâmico de carga [4]. Ao explorar o balanceamento de carga, o estado da arte mostra que a abordagem mais comum para a elasticidade é a reativa baseada na replicação de MVs [6, 7, 8]. Assim, quando um determinado índice de observação (*threshold*) de uma métrica ou combinação delas for atingido, dar-se-ão as ações de elasticidade para corrigir a carga. A Figura 2 apresenta esse cenário, que é muito visto em sistemas web hospedados em nuvem. O balanceador de carga possui um endereço URL e é o ponto de entrada para um serviço que pode ser acessado, por exemplo, via requisições HTTP diretas (Get e Post), Web Services ou REST. Nesse caso, ele é responsável por gerir o número de réplicas de acordo com métricas, como tempo de resposta das requisições, cargas nas instâncias, interações pela rede etc.

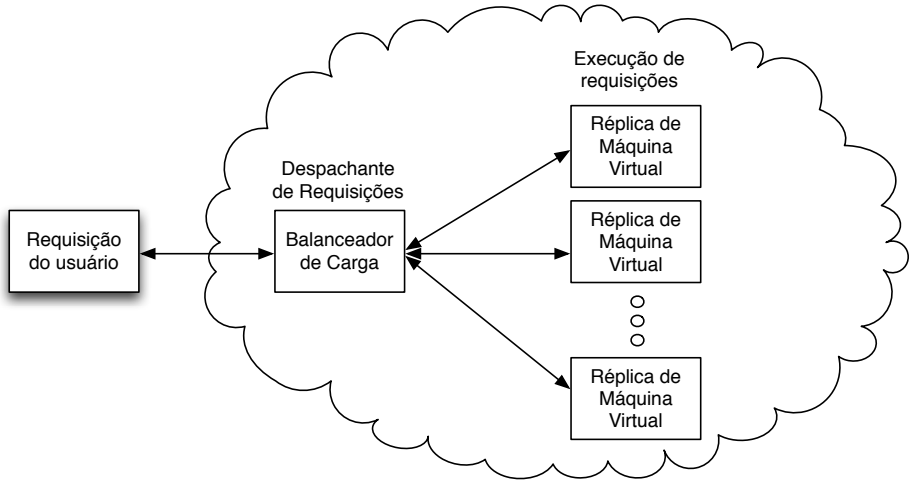

Figura 2: Abordagem tradicional da elasticidade com um balanceador de carga. Pode ser vista em sistemas como Amazon AWS [9], Nimbus [10] e Windows Azure [11]

Nesse contexto, o presente artigo apresenta em detalhes o tema elasticidade em ambientes de nuvem computacional. Ele é abordado quanto ao nível de atuação na nuvem, métricas para a sua avaliação, interface de interação e configuração verificando-se se é gerida por meio de mecanismos reativos (com uso de limites mínimo e máximo para aumentar ou diminuir recursos) ou proativos (modelos matemáticos de predição de comportamento). Além da replicação, o texto cobre outras técnicas para gerir a elasticidade, como redimensionamento de recursos e migração de máquinas virtuais. Os tópicos descritos acima são analisados tanto para sistemas consolidados disponíveis na Web quanto para iniciativas acadêmicas de pesquisa. Por fim, o artigo apresenta uma discussão sobre oportunidades de trabalho na área, enfatizando, principalmente, lacunas para que a elasticidade ganhe espaço além do cenário típico mostrado na Figura 2.

Este artigo está organizado em seis seções. A seção 2 discute a elasticidade perante diferentes abordagens de avaliação. A seção 3 descreve iniciativas de empresas disponíveis na web assim como trabalhos prevenientes da academia. A seção 4 mostra um estudo de caso aplicado à computação de alto desempenho. A seção 5 faz uma discussão sobre o tema elasticidade, elencando lacunas e oportunidades de trabalho. Por fim, a conclusão do artigo é apresentada na seção 6, que ressalta os principais desafios da elasticidade na atual computação em nuvem.

2 Analisando a elasticidade sob diferentes aspectos

Seguidamente, o termo "elasticidade" é confundido com "escalabilidade" [12]. A elasticidade diz respeito à capacidade, proativa ou reativa, de aumentar ou diminuir os recursos de um serviço em tempo de execução. A noção de tempo na elasticidade é crucial, envolvendo tanto o atraso para a percepção da necessidade de reconfiguração quanto a duração desse procedimento. Já a escalabilidade define a habilidade de um sistema de lidar com uma quantidade maior de carga à medida que novos recursos são adicionados, mantendo um nível de desempenho uniforme ou aproximado. Diferentemente da elasticidade, o conceito de escalabilidade é livre da noção de tempo.

Colocadas as definições, as subseções que seguem descrevem a elasticidade perante diferentes critérios de análise.

2.1 Modalidade

A modalidade da ação da elasticidade decide que tratamento será dado aos novos recursos. Vários autores concordam quanto à divisão de elasticidade horizontal e vertical para essa classificação [13, 14, 15, 16]. Na horizontal é possível aumentar ou diminuir o número de instâncias (MVs), assim como é possível sua migração para novos nós de processamento. Já na vertical, há o redimensionamento de atributos de CPU, disco, rede ou memória, além da alternativa de alocar ou desalocar nós de computação.

O uso de máquinas multicore leva a que a abordagem horizontal seja bastante explorada, como no caso dos provedores Amazon e Windows Azure. Por exemplo, é possível adotar uma estratégia de mapeamento de uma MV por núcleo de processamento (core) existente na infraestrutura. Quando se tem o uso completo de um nó, a necessidade de recursos pode implicar a abordagem vertical com a alocação de um novo nó. Segundo Younge et al. [17], essa estratégia é pertinente para a economia de energia elétrica. Tais autores mostram em gráficos que a alocação de uma MV no mesmo nó de computação é menos custosa em termos de potência (Watts) do que alocar um novo nó e ali lançar uma MV.

2.2 Política de alocação de recursos

Galante e Bona [18] apresentam esse item em duas abordagens: (i) manual e (ii) automática. O item ii ainda pode ser desmembrado nas classificações (a) reativa e (b) proativa. A classificação manual exige uma ação do usuário/programador. Ela pode representar o emprego de uma interface de programação (API) oferecida pelo *middleware* de nuvem para que o programador escreva o seu próprio gerenciador ou o uso de ferramentas, sejam elas em linha de comando ou gráficas. Quanto às abordagens automáticas, a reativa é marcada pelo emprego do mecanismo regra-condição-ação [19, 20, 21]. Nesse caso, é comum o emprego de *thresholds* na escrita das regras para gerência das métricas. Um *threshold* superior é usado para a alocação, enquanto outro, inferior, é útil para a consolidação de recursos. Apesar da denominação automática, normalmente a técnica reativa passa por uma pré-configuração pelo usuário, que deve redigir as regras, condições e ações [9, 11]. Além de requerer perícia na ferramenta de nuvem, cada novo serviço necessita de um novo esforço para a redação do trio acima.

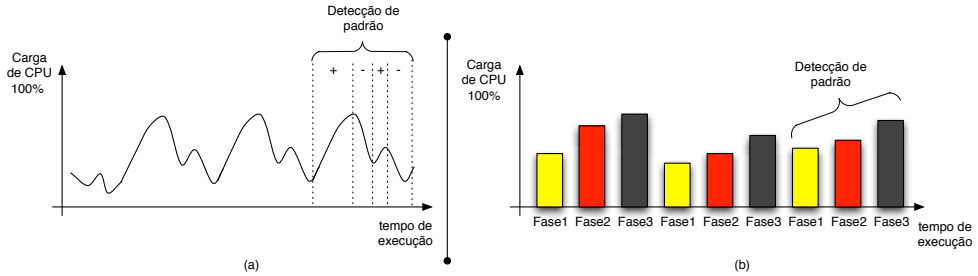

Figura 3: Política de alocação automática e proativa: (a) monitoramento contínuo do serviço ou da aplicação para a detecção dinâmica de padrões; (b) discretização do tempo de cada fase e uso da técnica de profile

A abordagem reativa é tradicional em sistemas de nuvem disponíveis na web, como Amazon, Windows Azure e Nimbus. Já a proativa usa técnicas de predição para antecipar o comportamento de carga do sistema e, assim, decidir pelas ações de elasticidade. Tecnicamente, é comum que implementações proativas usem *Fast Fourier Transform* (FFT), Wavelets, séries temporais e/ou perfis (*profiles*) [22, 15, 23]. A Figura 3 ilustra o comportamento de uma aplicação que apresenta um padrão de comportamento quanto ao uso de CPU. Logicamente, um gerenciador da elasticidade proativo pode antecipar a reconfiguração de recursos e, por exemplo, reduzir o tempo de conclusão da aplicação. Um dos maiores desafios na implementação de um gerente desse tipo é a detecção dinâmica da janela de observação [22]. A Figura 3 (b) mostra o emprego da técnica de *profile*, que é comum na computação de alto desempenho para a detecção de possíveis gargalos. Em especial, essa parte da figura mostra o comportamento de uma aplicação iterativa composta por três fases. Nesse caso, o gerenciador da elasticidade pode alocar recursos no decorrer da fase1 para a fase3 e desalocá-los na finalização de uma iteração.

2.3 Objetivos e métricas de atuação

Um objetivo ou uma métrica define uma variável de estudo, ou coleção delas, que servirá para lançar as ações de elasticidade. A abordagem mais comum é analisar a carga da CPU de cada um dos nós ou a carga de uso de cada uma das instâncias. Com base em dados coletados, faz-se uma média aritmética para determinar a métrica da elasticidade. Na sequência, é possível estabelecer *threholds* máximo e mínimo, nos quais a elasticidade vertical e/ou horizontal é dada se a média obtida está fora dos limites. Essa abordagem é seguida por vários trabalhos que não são uníssonos na definição do *threshold* superior: (i) 70% [6]; (ii) 75% [7]; (iii) 80% [24, 25]; (iv) 90% [20, 26, 27]. Tomando como métrica o uso de CPU, outros itens também podem ser levados em conta, como o relógio do processador, o tempo de execução de tarefas (*makespan*), o número de instruções e o redimensionamento de prioridades de uso de CPU. Por exemplo, Raveendran et al. [21] trabalham com um tempo máximo para a conclusão de cada uma das fases de um programa iterativo. No início de uma nova iteração, busca-se verificar se o tempo da anterior não excedeu um *threshold*, e, em caso positivo, uma nova máquina virtual é criada.

Outras medidas para monitoramento verificam o custo financeiro da infraestrutura alocada [8, 15], controle da energia elétrica [22, 28, 17], número de requisições em fila [29], ou ainda, a memória usada pelo serviço [24, 9, 30, 31]. Quanto ao custo, a elasticidade executada com o aumento de recursos acarreta um maior gasto para o consumidor. Então, pesquisas nessa linha procuram um melhor mapeamento de recursos dada uma certa quantidade de dinheiro (*budget*) e, normalmente, fazem uso da migração de MVs para balanceamento de carga. Isso porque esse movimento nunca ocasiona no aumento do custo, e sim pode gerar sua consolidação e redução. Quanto ao controle de energia elétrica, Younge et al. [17] mostram experimentos que corroboram a tese que defendem, segundo a qual, é mais efetivo em termos de energia o mapeamento de MVs em um nó com várias cores em vez da alocação de vários nós. Eles mostram que, além da redução da potência consumida (Watts), o desempenho multicore é equiparável com aquele obtido em múltiplos nós. Já quanto à memória, existem aplicações intensivas em relação ao emprego de operações de E/S, principalmente aquelas que operam métodos numéricos e simulações, que precisam manipular com uma grande quantidade de memória RAM. Muitas vezes, o conjunto alocado de MVs ou nós não oferecem os recursos requeridos, sendo necessária a inclusão de novos. Nessa linha, sistemas que oferecem a abstração de memória distribuída compartilhada (DSM) podem ser usados em nuvem por oferecerem um único espaço de endereçamento de memória e facilitarem a interação entre os fluxos de execução.

Outra técnica mais formal para o controle da elasticidade é o estabelecimento de objetivos de nível de serviço, ou SLO [7, 32]. Ele é útil para definir os objetivos de disponibilidade e desempenho para um aplicativo. SLO é a parte chave para estabelecimento de um acordo de nível de serviço (SLA) entre o usuário e o provedor de nuvem. É comum uma confusão entre os termos SLA e SLO para nuvem [33]. O primeiro trata de todo o acordo, especificando qual o serviço a ser provido, temporizações, custos, desempenho e responsabilidades. O segundo, por sua vez, analisa características mensuráveis de um SLA, como disponibilidade, vazão, frequência, tempo de resposta e qualidade. Um SLO pode ser composto por uma combinação de requisitos para essas características. O exemplo "90% das requisições devem ser atendidas com sucesso e cada qual deve demorar no máximo 20 segundos" engloba ambos, vazão e tempo de resposta, para definição do SLO. A elasticidade ocorre caso o SLO seja desrespeitado pelo provedor num determinado momento.

2.4 Estratégias

É possível classificar as estratégias de elasticidade em três grupos: (i) replicação; (ii) migração; e (iii) redimensionamento. A replicação é a mais comum, e seu funcionamento foi mostrado na Figura 2. O passo inicial para a execução da replicação é a criação de uma imagem ou *template*, que será instanciado para criar uma máquina virtual. A replicação de MVs é pertinente para tolerância a falhas e balanceamento de carga. Normalmente, as instâncias ou réplicas não estabelecem comunicação entre si, mas sim com um administrador que atua como um despachante de requisições. Ainda, é comum que as réplicas tenham acesso a um banco de dados compartilhado. Replicação é, normalmente, vista em sites de comércio eletrônico e de busca pela internet [9, 34]. Há um centralizador com endereço IP conhecido que recebe requisições, as repassa para as réplicas e gerencia a sua escala de acordo com a demanda. Replicação também é uma estratégia corriqueira em aplicações de alto desempenho *Single Program Multiple Data* (SPMD) que seguem o modelo mestre-escravo [12, 35, 21]. Por exemplo, uma nova MV para um escravo é lançada em decorrência do aumento de computação detectado pelo mestre.

A migração pode ser vista como a técnica mais trivial para a expressão da elasticidade [7, 28, 22, 15, 31, 23].

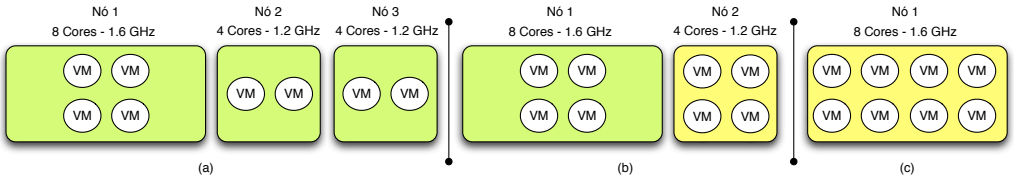

Figura 4: Elasticidade através de migração de máquinas virtuais: (a) configuração inicial; (b) 2 migrações e consolidação de 1 nó físico; (c) 4 migrações e consolidação de 2 nós físicos

Máquinas virtuais possuem a característica de isolamento, e *hypervisors* como XEN[3] e KVM[4] suportam a sua transferência entre máquinas físicas com um nível de desempenho aceitável. Ambos possuem a capacidade conhecida como *live migration*, que reduz o tempo de inatividade (*down time*) do serviço por meio da cópia otimizada de páginas de memória. Em nível de aplicação, nenhuma modificação é necessária, e o próprio *hypervisor* responsabiliza-se pela reorganização das conexões de rede.

Além de balanceamento de carga, a migração é essencial para implementar mecanismos de controle de energia elétrica. Isso porque máquinas virtuais podem ser reunidas num único recurso físico e os demais podem ser desligados. A Figura 4 ilustra essa situação. Apesar de o nó 1 possuir 8 núcleos e um relógio mais potente, o desempenho do cenário (c) não necessariamente é melhor que os demais. Deve-se levar em conta a arquitetura de interconexão entre os núcleos dentro do processador e questões como contenção de acesso à memória principal. No que se refere ao redimensionamento, ele engloba tanto alterações de configuração de recursos quanto a adaptação de aplicações. No primeiro caso, é possível alterar a percentagem de CPU destinada a uma MV ou, ainda, redimensionar dinamicamente a largura de banda suportada por uma rede privada virtual (VPN). No segundo, por exemplo, a aplicação reage à inclusão de novos recursos e pode criar novas *threads* ou novos processos para usufruir da reconfiguração da infraestrutura. Essa abordagem é diferente da replicação, na qual adaptações no serviço não são realizadas.

2.5 Interface de uso

As três estratégias de elasticidade discutidas na subseção 2.3 podem ser viabilizadas por meio do uso de uma ou da combinação das seguintes interfaces: (i) linha de comando; (ii) interface gráfica (GUI); (iii) biblioteca de programação que segue uma das APIs suportadas pelo *middleware* de nuvem. A linha de comando é uma abordagem bastante explorada em nuvens privadas e normalmente associada com o sistema operacional GNU-Linux. Apesar de essa interface proporcionar alta flexibilidade, ela requer um nível de perícia avançado por parte dos usuários. Essa dificuldade é minimizada com o apoio de uma ferramenta gráfica, que normalmente é dada mediante um navegador web.

A Figura 5 apresenta o gerente gráfico Sunstone disponibilizado pelo OpenNebula. Por fim, o uso de API é pertinente para a construção de programas que gerenciam os recursos na nuvem. Por exemplo, é possível escrever um programa que lance máquinas virtuais na nuvem e que, na sequência, as monitore de modo a tomar ações de elasticidade para corrigir o desempenho. Normalmente, cada uma das três interfaces é passível de uso sob os modos usuário (com ações específicas sobre os seus serviços e componentes autorizados para si) e administrador (sem limitações quanto à gerência da nuvem).

O uso de API é pertinente para desonerar a tarefa de monitoramento do usuário. Em nível de programação, é possível escrever um gerente que realiza uma verificação periódica de recursos físicos e virtuais. Caso o SLA seja desrespeitado ou níveis de *threshold* sejam atingidos, ações de elasticidade podem ser tomadas de acordo com as estratégias já mencionadas. Essa abordagem de monitoramento e elasticidade reativa é vista em nuvens públicas como a Amazon AWS e Windows Azure, bem como em iniciativas de pesquisa como Cloud Operating System [7], Elastack [20] e Sandpiper [31]. Quanto a interfaces, a mais conhecida é aquela oferecida pela Amazon, denominada de Amazon Web Services (AWS)[5]. Utilizando-se de chamadas à Web Services, um usuário pode criar, lançar e destruir instâncias de servidores de acordo com a sua necessidade, pagando por hora sobre a quantidade

[3] http://www.xen.org/
[4] http://www.linux-kvm.org/
[5] http://aws.amazon.com/

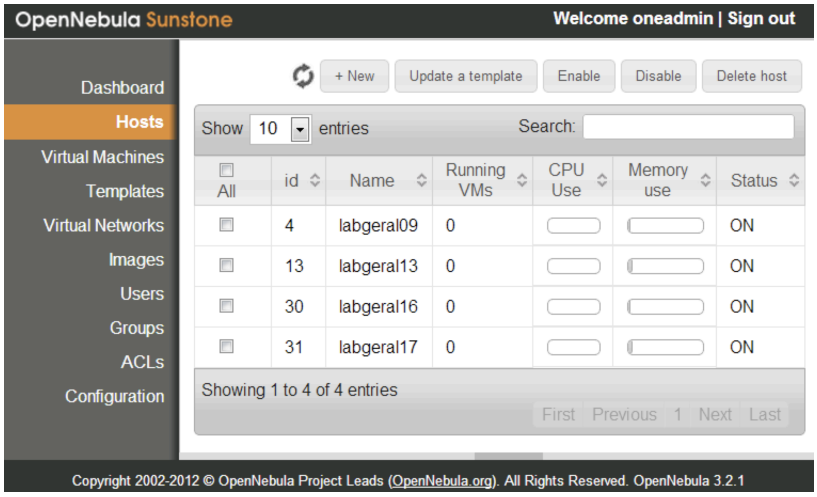

Figura 5: Interface gráfica Sunstone oferecida pelo *middleware* de nuvem OpenNebula

de instâncias. Apesar de não ser um padrão, AWS é a interface mais usada para explorar os recursos de nuvem. Nesse sentido, os demais *middlewares*, em sua maioria, oferecem a sua própria API e compatibilidade com AWS. A seguir, são apresentados alguns *middlewares* de nuvem e as APIs que suportam: (i) OpenNebula [36]: AWS, OCCI (Open cloud Computing Interface), Java e vCloud; (ii) RackSpace [11]: RESTful, C#/.NET, Python, PHP e Java; (iii) Nimbus [10]: WSRF e AWS; (iv) Eucalyptus [37]: AWS e Java; (v) OpenStack [38]: Java e AWS; (vi) CloudStack [39]: Java e AWS. Nuvens privadas mantêm compatibilidade com a interface AWS para a formação de nuvens híbridas. Dessa forma, é possível transferir MVs para a nuvem pública quando a organização privada não obtiver recursos suficientes para manter o SLA acordado.

2.6 Aplicação da elasticidade em diferentes tipos de recursos

A computação em nuvem e a elasticidade estão ligadas com a gerência dinâmica de recursos computacionais, em especial o uso de CPU. O cenário atual da virtualização de CPU pode ser decomposto em três categorias principais [40]: (i) virtualização completa; (ii) virtualização em nível de sistema operacional e paravirtualização; e (iii) virtualização assistida por hardware. Enquanto o aprovisionamento de recursos de CPU é largamente explorado na literatura, ganha força nos últimos anos um conceito chamado nuvem de comunicação (*Cloud Networking*) [41], que introduziu a entrega de canais virtuais de comunicação como um serviço, de forma similar à abordagem utilizada pela nuvem tradicional. Esse conceito vem transformando a internet em um amplo conjunto de recursos virtuais por meio dos quais os serviços de computação, armazenamento e comunicação podem ser reservados e aprovisionados dinamicamente para diferentes usuários e aplicações. Sua aplicabilidade tem sido estudada em camadas distintas do modelo de referência OSI.

Entre as principais tecnologias para a virtualização de rede, é possivel citar: (i) redes locais (LANs) virtuais; (ii) redes privadas virtuais (VPNs); e (iii) *overlays*. Por exemplo, diante de um cenário com diversos usuários, cada qual com a sua rede local virtual para acessar o mesmo servidor, dependendo do perfil do usuário e de questões envolvendo custo, a estratégia de redimensionamento para a elasticidade pode alocar mais largura de banda em certos momentos para o tráfego de dados de uma rede virtual específica.

3 Implementações da elasticidade: iniciativas disponíveis na web e provenientes de atividade de pesquisa

O tema computação em nuvem é largamente abordado tanto por provedores com intuito comercial ou de código aberto quanto por trabalhos acadêmicos. Os provedores de nuvem destacam-se por apresentarem uma solução robusta pronta para uso, enquanto os artigos acadêmicos mostram arcabouços, *plugins* e/ou novos algoritmos para o problema da elasticidade em nuvem. Em especial, a Amazon tem um papel chave na área de computação

em nuvem, na qual vem atuando desde 2006. Como dito anteriormente (subseção 2.5), ela influencia o mercado de modo que outros sistemas ofereçam compatibilidade com a sua interface AWS. Em se tratando de nuvens privadas, por outro lado, o sistema Eucalyptus foi um dos precursores da adoção da virtualização em ambientes corporativos. As subseções que seguem descrevem em detalhes as iniciativas de elasticidade nos dois ramos apresentados.

3.1 Iniciativas disponíveis na web

A Tabela 1 apresenta alguns sistemas marcantes que estão disponíveis para download na web. Os sistemas nessa modalidade destacam-se por oferecerem o tratamento da elasticidade de forma manual para o usuário [39, 42, 43, 37, 36, 38] ou por meio de pré-configuração de mecanismos com elasticidade reativa [9, 10, 11, 34]. Na abordagem manual, salienta-se o uso de *middlewares* tradicionais para a construção de nuvens privadas. O usuário pode fazer uma aplicação para monitorar os serviços que rodam sobre MVs e disparar por conta própria as ações de elasticidade. Sistemas como Amazon AWS, Nimbus e Windows Azure oferecem subsistemas parametrizáveis para monitoramento de serviços e gerenciamento da elasticidade. Os usuários devem completar as regras e os *thresholds* superior e inferior de uma métrica a ser monitorada, bem como as condições e as ações para a reconfiguração.

A Amazon AWS disponibiliza uma ferramenta gráfica e uma API onde o usuário escolhe uma métrica para ser monitorada e uma ação que será lançada quando tal métrica atingir um determinado limite. O monitoramento é feito pela ferramenta CloudWatch[6], e a elasticidade de fato é gerida de forma reativa por outra denominada de AutoScaling[7]. O Amazon CloudWatch habilita o monitoramento de recursos AWS em tempo real, incluindo instâncias EC2 e o balanceador de carga. Métricas como utilização de CPU, latência de rede, contador de requisições são suportadas por ele. O usuário ainda pode especificar a sua própria métrica, como uso de memória, volume de transações em banco de dados e taxa de erros.

Tabela 1: Tratamento da elasticidade por sistemas disponíveis na Web e iniciativas de pesquisa da academia

Sistemas	Nível	Modalidade	Política	Objetivo / Métrica(s)	Método	Interface/Observações
OpenNebula [36]	IaaS	Horizontal e vertical	Manual	Def. pelo usuário	Replicação e migração	Linha de comando, interface gráfica, API XML-RPC
Eucalyptus [37]	IaaS	Horizontal e vertical	Manual	Def. pelo usuário	Replicação e migração	Linha de comando, ferramenta gráfica e API XML-RPC e EC2
Amazon AWS [9]	IaaS	Horizontal	Automática, reativa, pré-configuração	CPU, rede, memória e disco	Replicação	Amazon AutoScale e Cloud Watch
Microsoft Azure [11]	IaaS, PaaS	Horizontal	Automática, reativa, pré-configuração	CPU e rede	Replicação e migração	Ferramenta Gráfica e API .NET
Nimbus [10]	IaaS	Horizontal	Automática, reativa, pré-configuração	CPU	Replicação	Ferramenta gráfica ou plugin Phanton
OpenStack [38]	IaaS	Horizontal	Manual	Def. pelo usuário	Replicação	Linha de comando e API
CloudStack [39]	IaaS	Horizontal e vertical	Manual	Def. pelo usuário	Replicação e migração	Linha de comando, interface gráfica e API
RightScale [34]	IaaS	Horizontal	Automática, reativa, pré-configuração	CPU, número de trabalhos em fila	Replicação	Arcabouço e interface gráfica
Heroku [43]	PaaS	Horizontal	Manual	CPU e rede	Replicação e redimensionamento	Linguagem de programação usada no SalesForce
GoGrid [42]	IaaS	Horizontal	Manual	CPU	Replicação	RESTful e interface gráfica

O Windows Azure oferece uma biblioteca (*AutoScaling Application Block*) em que o usuário deve adicionar a sua aplicação para prover a elasticidade. O usuário pode usar contadores de desempenho e escrever regras de elasticidade. O Windows Azure organiza a submissão de um serviço em dois arquivos: (i) configuração, que relata as imagens e os parâmetros de inicialização; (ii) mapeamento (*deploy*), que informa a quantidade de instâncias para cada serviço. Assim, a biblioteca permite chamar métodos para que o segundo arquivo seja dinamicamente recriado para a reconfiguração dos recursos.

O AzureWatch[8] é outra maneira de controlar o número de instâncias. Ele é um arcabouço adaptado para o Windows Azure, que trabalha com informações de uso de CPU, dados de MVs, histórico, tempo de requisições, e

[6] http://aws.amazon.com/cloudwatch
[7] http://aws.amazon.com/autoscaling
[8] http://www.paraleap.com/azurewatch

também é dirigido por regras explícitas de atuação. Além das duas opções em nível de programação, o Windows Azure oferece outra por meio de interface gráfica, na qual é possível mudar o número de instâncias e o tamanho máximo do banco de dados associado ao serviço. O sistema Nimbus, por sua vez, trabalha como um escalonador de trabalhos em *batch* para ambientes de nuvem, podendo atuar sobre o Amazon EC2. Nimbus utiliza Phantom[9] para monitorar o comportamento dos recursos e para disparar, de forma reativa, ações de escalabilidade. A configuração de Phantom permite completar os seguintes parâmetros: (i) número máximo e mínimo de MVs; (ii) número de MVs que serão lançadas quando encontrado um limite superior; (iii) número de MVs que serão consolidadas quando atingido um limite inferior.

3.2 Iniciativas provenientes de pesquisa acadêmica

Esse grupo de trabalhos busca sanar lacunas e/ou aprimorar abordagens para o tratamento da elasticidade, tendo como referência tanto as iniciativas disponíveis na internet quanto o estado da arte de gerentes (*hypervisors*) de ambientes virtuais. Com base na Tabela 2, é possível montar uma análise detalhada das iniciativas da academia para a gerência da elasticidade por nichos. Entre eles, é possível destacar: (i) foco em computação paralela e de alto desempenho [20, 30, 7, 28, 8, 12, 35, 21]; (ii) análise do custo financeiro da elasticidade [8, 15]; (iii) computação em nuvem de forma federada [44, 7, 29, 25]; (iv) modelo preditivo para detecção de padrões de comportamento de serviços em nuvem [8, 22, 15, 32, 31, 23]; (v) foco na economia de energia e na superlotação de máquinas virtuais sobre físicas [28, 22]; (vi) computação em tempo real e tratamento de *deadlines* na nuvem [8, 22, 21, 45]; (vii) elasticidade em *middlewares* orientados a mensagens (MOM) [45]; (viii) cargas de trabalho e aplicações web voltadas para negócios ou área transacional [6, 26, 12, 27, 31, 46].

ElasticMPI propõe a elasticidade em aplicações *Message Passing Interface* (MPI) com a parada e o relançamento dessas aplicações no momento da reconfiguração de recursos [21]. O sistema assume que o usuário sabe de antemão o tempo esperado para cada uma das etapas de seu programa. Desse modo, o sistema de monitoramento pode detectar que a configuração corrente não conseguirá cumprir o *deadline* esperado e novos recursos são adicionados. Em adição, a abordagem de ElasticMPI faz uma alteração no código fonte da aplicação, de modo a inserir diretivas de monitoramento. Imai et al. propuseram um sistema operacional para nuvem chamado de COS [7], o qual oferece um arcabouço genérico baseado em uma linguagem de programação orientada a atores chamada de SALSA. Quando todas as MVs estão sobrecarregadas, é possível que ocorram migrações tanto em nível de MV (passar de um nó para outro) quanto em nível dos próprios atores (passar de uma MV para outra). O trabalho desses autores necessita de aplicações que sejam compostas estritamente de componentes migráveis escritos em SALSA.

Ming, Li e Humphrey [8] apresentam o conceito de autoescalabilidade, responsável por alterar o número de instâncias baseado em informações da carga de trabalho sem a intervenção do usuário. Uma vez que o programa possui *deadlines* para execução de suas fases, a proposta trabalha com recursos de MVs e nós para cumprir o prazo. Martin et al. [12] apresentam um cenário típico de requisições sobre um serviço em nuvem que atua com um balanceador de carga. A elasticidade altera a quantidade de MVs trabalhadoras de acordo com a demanda no serviço. O sistema é baseado em Web Services, onde cada trabalhador executa Apache Tomcat 6. Em adição, o balanceador de carga realiza o escalonamento usando a política Round-Robin. Elastack é um sistema que executa sobre OpenStack para suprir a carência de elasticidade desse último [20]. Para a detecção de mudanças no ambiente, Elastack usa o *middleware* Serpentine [47], que executa em cada uma das instâncias trabalhadoras, as quais comunicam-se com um balanceador de carga. No momento que termina a carga a ser processada, O Serpentine avisa o balanceador de carga que aquela instância pode ser consolidada.

4 Observando a potencialidade da elasticidade para computação de alto desempenho

Com o intuito de testar a elasticidade em ambiente de computação em nuvem, foi desenvolvido um protótipo simples chamado de OpenElastic, que foi construído com a API Java XML-RPC de OpenNebula, operando, então, sobre nuvens privadas, e segue a estratégia de elasticidade reativa com uso de *thresholds*. Ele é responsável por lançar uma aplicação paralela na nuvem, por monitorá-la e por ativar eventuais ações de elasticidade. A aplicação é iterativa e segue o modelo de programação paralela mestre-escravo, podendo ser vista nas subdivisões (a) e (b)

[9] http://www.nimbusproject.org/doc/phantom/latest

Tabela 2: Tratamento da elasticidade por sistemas disponíveis na web e iniciativas de pesquisa da academia

Sistemas	Nível	Modalidade	Política	Objetivo / Métrica(s)	Método	Interface/Observações
Just in Time (JIT) Clouds [44]	IaaS	Horizontal e vertical	Manual	Def. pelo usuário	Replicação	API particular
PRESS [32]	IaaS	Horizontal	Automática, reativa	Objetivo de nível de serviço (SLO)	Redimensionamento	Arcabouço de gerenciamento para o XEN
ElasticMPI [21]	PaaS	Vertical	Automática, reativa	CPU	Replicação e redistribuição de dados	Sistema que altera a aplicação
Work Queue [35]	PaaS	Horizontal	Manual	CPU	Redimensionamento	Arcabouço sobre Windows Azure e Amazon AWS
Cloud Operating System (COS) [7]	PaaS	Horizontal	Automática, reativa	CPU e rede	Replicação e migração de objetos	Gerenciamento em nível de sistema operacional
Ming, Li e Humphrey [8]	IaaS	Horizontal	Automática, proativa	Custo financeiro e tempo de término de jobs	Replicação	Sistema sobre o Windows Azure
Martin et al. [12]	SaaS	Horizontal e vertical	Automática, baseado no monitoramento dos agentes	CPU	Replicação	Funciona sobre Amazon EC2
Lightweight Scaling [30]	IaaS	Horizontal e vertical	Automática, reativa	CPU e memória	Replicação e redimensionamento	Arcabouço de gerência de recursos
Kingsfiher [15]	IaaS	Horizontal	Automática, proativo e reativo	Custo financeiro	Replicaçcão e migração	Sistema executa sobre OpenNebula
Moreno e Xu [22]	IaaS	Horizontal e vertical	Automática, proativa	Energia elétrica	Replicação e migração	Arcabouço de gerência de recursos
Scattered [23]	IaaS	Horizontal	Automática, proativa	CPU e rede	Migração	Arcabouço de gerência de recursos
Sandpiper [31]	IaaS	Horizontal e vertical	Automática, reativa e proativa	CPU, rede e memória	Redimensionamento e migraçcão	Arcabouço de gerência de recursos sobre o hipervisor Xen
Elastack [20]	IaaS	Horizontal	Automática, reativa	CPU	Replicação	Arcabouço para gerência de serviços para OpenStack
Knauth e Fetzer [28]	IaaS	Horizontal	Automática, reativa	Energia elétrica	Migraçcão	Arcabouço para gerência de serviços
Elastic Queue Service [45]	IaaS	Horizontal e vertical	Automática, reativa	Número de requisições	Replicação	Arcabouço para gerência de serviços obre o AWS
Elastic Site [29]	IaaS	Vertical	Automática, reativa, preconfiguração	Número de requisições	Replicação e redimensionamento	Gerenciador de recursos sobre o Nimbus
Dawoud et al. [6]	IaaS	Vertical	Automática, reativa, préconfiguração	Objetivo de nível de serviço (SLO)	Redimensionamento	Gerenciador de recursos sobre o XEN
Suleiman [27]	IaaS	Vertical	Automática, reativa	CPU	Replicação	Utiliza o Amazon AWS e Cloud Watch
Zhang et al. [46]	IaaS	Horizontal	Automática, reativa, preconfiguração	CPU	Replicação	Baseado em componentes weblets
Kaleidoscope [24]	IaaS	Horizontal	Automática, reativa	CPU e memória	Replicação (clonagem)	Microelasticidade com clonagem de MVs

da Figura 6. O OpenElastic atua fora da nuvem e conecta-se ao servidor OpenNebula via SSH. O servidor e os nós de computação da nuvem compartilham um diretório para troca de dados via *Network File System* (NFS). Essa abordagem de diretório foi escolhida para que o OpenElastic avise o mestre de novos recursos e que receba dele a autorização para consolidá-los quando necessário. Nesse sentido, o diretório é usado para as seguintes ações: (i) OpenElastic avisa o mestre que novas MVs estão disponíveis; (ii) OpenElastic pede permissão para consolidar MVs e um nó de computação; (iii) processo mestre concede a permissão para consolidação. A ação 2 é pertinente para que não seja terminada a execução de um processo no meio de uma etapa de processamento.

O OpenElastic realiza o monitoramento periódico da aplicação num intervalo de 30 segundos. Para ativar a elasticidade, faz uso da técnica de envelhecimento (*Aging*), na qual a última observação tem um peso maior [48]. Por exemplo, as medidas de carga C={78, 89, 83, 79, 84, 90} resultariam numa carga total de CT=$\frac{1}{2}.78 + \frac{1}{4}.89 + \frac{1}{8}.83 + \frac{1}{16}.79 + \frac{1}{32}.84 + \frac{1}{64}.90$=83.59. Com um *threshold* de 80%, serão disparadas ações de elasticidade. Além desse *threshold*, o OpenElastic adota outro para a desalocação de 20%. Diferente de Amazon AWS e Windows Azure, esse protótipo não necessita de uma pré-configuração, abstraindo questões técnicas e o uso da elasticidade para o usuário. Caso opte, ele pode informar um SLA via padrão WS-Agreement para denotar os números máximo e mínimo de MVs para a execução de sua aplicação.

O grão de trabalho de OpenElastic é sempre um nó com n MVs, cada qual com um processo escravo. O

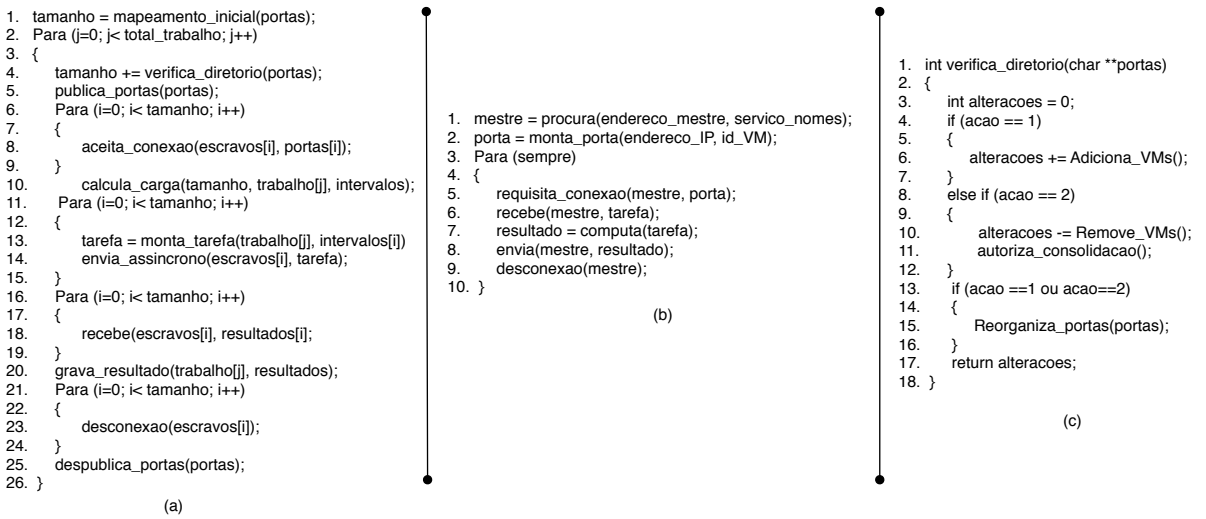

Figura 6: Pseudocódigo de uma aplicação que executa sob gerência de OpenElastic: (a) aplicação do mestre; (b) aplicação escravo; (c) desenvolvimento da função que controla a elasticidade no processo mestre

valor de n é igual ao número de núcleos de processamento (cores) que o nó possui. Assim, num momento de sobrecarga, o OpenElastic aloca um nó e máquinas virtuais e informa o endereço IP de cada uma via diretório compartilhado. A aplicação paralela calcula a soma de vetores, onde cada processo escravo recebe uma parte da entrada. Assim, o OpenElastic suporta a elasticidade para aplicações que seguem o paralelismo de dados. O processo mestre trabalha com um milhão de pares de vetores de entrada e realiza a soma de cada dupla com a ajuda dos escravos. O tamanho e os elementos em ponto flutuante de cada vetor foram obtidos de forma aleatória.

Os testes foram feitos com uma configuração inicial que continha dois nós, cada qual com 2 máquinas virtuais. A abordagem sem elasticidade levou 22 minutos e 34 segundos, enquanto que a elástica demorou 19 minutos e 23 segundos. Essa última tinha a capacidade de alocar mais dois nós de computação, caso necessário. Foram feitas 30 execuções que resultaram num desvio padrão de 22 segundos. O tempo médio para um lançamento completo de uma MV foi de 1 minuto e 14 segundos. Por fim, a abordagem do OpenElastic que usa um gerente externo à aplicação foi pertinente para que ela não fique bloqueada na alocação ou desalocação de uma MV. Como é possível observar na Figura 6, a aplicação foi modelada segundo as diretivas da biblioteca de Sockets do sistema operacional GNU-Linux. Essa interface pode ser adaptada para MPI 2, que também suporta conexões (*accept* e *connect*) entre os processos.

5 Discussão, desafios e oportunidades de trabalho

Uma análise do estado da arte no tópico elasticidade na computação em nuvem permite concluir que a estratégia mais comum é a de replicação, de forma reativa ou manual, e em nível IaaS. Nesse sentido, futuros trabalhos podem explorar a elasticidade em nível PaaS, onde são oferecidas bibliotecas e compiladores acoplados à instância de máquina virtual. A principal vantagem de operar nesse nível é desonerar o usuário de possíveis modificações na sua aplicação para torná-la elástica. É possível empregar compiladores especializados e/ou traduções fonte-para-fonte em nível PaaS de forma abstrata para o usuário.

Outro subtema pertinente dentro da elasticidade é o tratamento de falsos-positivos e falsos-negativos na criação e destruição de MVs. O primeiro engloba uma reconfiguração que é feita sem necessidade, e o segundo enfatiza a negligência de uma, enquanto ela é realmente útil. Iniciativas de pesquisa esperam um número x de ocorrências consecutivas além dos limites do *threshold* para lançar o escalonamento [20, 35, 21]. Por exemplo, com x igual a 3 e *threshold* máximo de 80%, essa técnica tem problemas quando são observadas as seguintes medidas de carga C={75, 98, 93, 79, 89, 96, 78}. Nesse caso, não ocorre a elasticidade, mas se percebe que em mais de 50% do tempo o sistema operou além de seu limite. Nesse sentido, técnicas como a do envelhecimento (*Aging*) [48] podem ser úteis para o tratamento de picos.

Outros pontos fracos, no tratamento da elasticidade, que podem ser revertidos em oportunidades de trabalho são mostrados na lista a seguir: (i) necessidade de alteração do código fonte da aplicação [35, 21]; (ii) uso de componentes proprietários, não disponíveis em sistemas operacionais como GNU Linux, Windows e MacOS [20, 7, 21]; (iii) necessidade de ter acesso a dados da aplicação ou do serviço antes de sua execução, tais como o tempo esperado de execução de cada componente ou previsão do tempo de chegada entre requisições [28, 49, 50, 21]; (iv) reconfiguração de recursos com parada total do serviço e posterior relançamento [21]; (v) comunicação na nuvem reduzida a interações entre réplica e gerenciador (balanceador) a uma taxa constante [23]; (vi) segurança na desalocação de MVs. O item (v) enfatiza que, normalmente, é limitada a comunicação dentro da nuvem, onde instâncias simplesmente interagem com um gerente. Entretanto, o paralelismo de tarefas e o mapeamento de aplicações paralelas modeladas em fases (*Bulk Synchronous Parallel*) ou em pipeline são um desafio para a elasticidade em nuvem, visto que conexões n-para-n devem ser mantidas e atualizadas a cada reconfiguração. Já o item (vi) demonstra a preocupação quanto à exposição de dados confidenciais no momento da desalocação de uma MV, visto que ela pode ser reusada por outro usuário. Essa consideração enfatiza a importância de procedimentos eficientes de automação para gerenciar a infraestrutura da nuvem.

No que se refere à expressão da elasticidade em nível de aplicação, merece destaque o trabalho de Dustdar et al. [51]. Eles apresentam, em alto nível, uma série de primitivas de programação pertinentes para tornar elástico um serviço ou aplicação em nuvem. Os autores as classificam dentro dos grupos monitoramento, regras e estratégias. A iniciativa de Dustdar et al. [51] tem relevância por ser uma das pioneiras na área de APIs específicas para a elasticidade.

Como discutido anteriormente (ver seção 2.5), a interface mais difundida para uso na nuvem ainda é a AWS da Amazon. Por meio dela ou de uma interface gráfica, o usuário configura detalhes técnicos relativos a *thresholds*, segurança e métricas. Claramente, essa questão pode ser um impeditivo para aqueles usuários (físicos, médicos, químicos, engenheiros etc) que possuem aplicações e não apresentam perícia na configuração de um sistema no estilo da Amazon. Recentemente, essa empresa lançou uma nova maneira de rapidamente executar aplicações em nuvem. Segundo a Amazon, a iniciativa chamada AWS Elastic Beanstalk[10] trata automaticamente de detalhes quanto ao aprovisionamento de recursos, balanceamento de carga e elasticidade após o upload da aplicação. Entretanto, ainda se faz necessária a configuração da ferramenta Amazon Auto Scaling para gerir as ações automáticas de Beanstalk.

6 Conclusão

A elasticidade é desejável tanto no âmbito comercial quanto no acadêmico. Em especial, para o primeiro, ela representa a possibilidade de pequenas empresas crescerem com um custo inicial também pequeno. Se o crescimento for menor que aquele esperado, pelo menos não há a necessidade de pagar pela compra de uma infraestrutura física. Agora, o contrário pode fazer uso da elasticidade para uma expansão ou atualização do parque de recursos necessários para atender a demanda requisitada pelo serviço. A elasticidade também é essencial para reduzir uma métrica muito conhecida da área de negócios: tempo para o mercado ou *time-to-market*. Novas ideias necessitam ser testadas e validadas, e o rápido aprovisionamento de recursos permite que ambas as tarefas sejam feitas com uma boa relação custo-benefício. Quanto ao âmbito acadêmico, a elasticidade merece destaque na era do Big Data, onde flutuações de carga tanto de comunicação (E/S) quanto de processamento estão presentes em aplicações de alto desempenho e mineração de dados. A alocação em nível IaaS de clusters com máquinas multicore é uma realidade em provedores como Amazon e Windows Azure, e a adoção de redes de sistema como Infiniband e 10 Gigabit Ethernet é uma crescente em ambientes de nuvem.

A seção 5 antecipou alguns desafios e oportunidades de pesquisa na área da elasticidade em nuvem. O artigo termina com a descrição de novos desafios que deverão ser enfrentados por analistas de TI e desenvolvedores de *middlewares*. São eles:

- Uso de uma interface padrão *de facto* para a computação em nuvem e expressão da elasticidade. Atualmente, o uso de Web Services do AWS desponta claramente como a interface mais usada, mas ainda não é reconhecida como um padrão aberto para nuvem.

[10] http://aws.amazon.com/elasticbeanstalk/

- A exploração da elasticidade em aplicações paralelas que seguem os modelos mestre-escravo e divisão-e-conquista é frequente em artigos que combinam alto desempenho e nuvem. Entretanto, há o desafio de construir aplicações elásticas escritas segundo os modelos de computação em fases síncronas e pipeline.
- Nuvens públicas são conhecidas pela elasticidade reativa e necessidade de uma etapa de pré-configração por parte do usuário. Um desafio para os provedores é retirar a necessidade de intervenção do usário e oferecer mecanismos genéricos que seguem a elasticidade proativa. Logicamente, o aumento automático de recursos acarreta um maior custo financeiro, e o usuário deve expressar algum controle sobre esse item.
- Como mencionado anteriormente, o uso de *thresholds* é comum para determinar ações de elasticidade. Os trabalhos com elasticidade reativa analisados operam com valores fixos para os limites inferior e superior. Com o crescimento do poder computacional observado hoje em dia, aumenta, também, o uso de aplicações irregulares cujo desempenho depende da qualidade dos recursos, da entrada de dados e da quantidade de fases. Um valor otimizado de *threshold* no início da aplicação pode ser ineficaz em outros momentos.
- Hoje em dia, serviços oferecidos em nível SaaS, como o Google Docs e Gmail, são muito difundidos e abstraem totalmente do usuário a infraestrutura da nuvem e detalhes técnicos. Espera-se que o desenvolvimento da elasticidade também aconteça no sentido da crescente abstração para o usuário nos níveis PaaS e IaaS.

Agradecimentos

Este trabalho é parcialmente financiado pelos seguintes órgãos de pesquisa brasileiros: CNPq (Projeto Ciência sem Fronteiras n^o 238206/2012-2) e Fapergs (Projeto CloudMan n^o 149412-5).

Referências

[1] RICCI, L.; CARLINI, E. Distributed virtual environments: From client server to cloud and p2p architectures. In: . c2012. p. 8–17.

[2] LUO, S.; PENG, X.; FAN, S.; ZHANG, P. Study on computing grid distributed middleware and its application. In: . c2009. v. 3. p. 441–445.

[3] DA ROSA RIGHI, R.; PILLA, L. L.; CARISSIMI, A.; NAVAUX, P. A.; HEISS, H.-U. *Parallel Processing Letters*, v. 20, n. 2, p. 123–144, June 2010.

[4] LECZNAR, M.; PATIG, S. Cloud computing providers: Characteristics and recommendations. In: BABIN, G.; STANOEVSKA-SLABEVA, K.; KROPF, P. (Eds.) *E-Technologies: Transformation in a Connected World*. Springer Berlin Heidelberg, 2011. v. 78 of *Lecture Notes in Business Information Processing*, p. 32–45.

[5] PARKHILL, D. *The challenge of the computer utility*. Number p. 246 in The Challenge of the Computer Utility. Addison-Wesley Pub. Co., 1966.

[6] DAWOUD, W.; TAKOUNA, I.; MEINEL, C. Elastic vm for cloud resources provisioning optimization. In: ABRAHAM, A.; LLORET MAURI, J.; BUFORD, J.; SUZUKI, J.; THAMPI, S. (Eds.) *Advances in Computing and Communications*. Springer Berlin Heidelberg, 2011. v. 190 of *Communications in Computer and Information Science*, p. 431–445.

[7] IMAI, S.; CHESTNA, T.; VARELA, C. A. Elastic scalable cloud computing using application-level migration. In: . UCC '12. Washington, DC, USA: IEEE Computer Society, c2012. p. 91–98.

[8] MAO, M.; LI, J.; HUMPHREY, M. Cloud auto-scaling with deadline and budget constraints. In: . c2010. p. 41 –48.

[9] CHIU, D.; AGRAWAL, G. Evaluating caching and storage options on the amazon web services cloud. In: . c2010. p. 17 –24.

[10] MARSHALL, P.; KEAHEY, K.; FREEMAN, T. Elastic site: Using clouds to elastically extend site resources. In: . c2010. p. 43 –52.

[11] ROLOFF, E.; BIRCK, F.; DIENER, M.; CARISSIMI, A.; NAVAUX, P. Evaluating high performance computing on the windows azure platform. In: . c2012. p. 803 –810.

[12] MARTIN, P.; BROWN, A.; POWLEY, W.; VAZQUEZ-POLETTI, J. L. Autonomic management of elastic services in the cloud. In: . ISCC '11. Washington, DC, USA: IEEE Computer Society, c2011. p. 135–140.

[13] ALI-ELDIN, A.; TORDSSON, J.; ELMROTH, E. An adaptive hybrid elasticity controller for cloud infrastructures. In: . c2012. p. 204 –212.

[14] FRISCHBIER, S.; PETROV, I. Aspects of data-intensive cloud computing. In: SACHS, K.; PETROV, I.; GUERRERO, P. (Eds.) *From Active Data Management to Event-Based Systems and More*. Springer Berlin Heidelberg, 2010. v. 6462 of *Lecture Notes in Computer Science*, p. 57–77.

[15] SHARMA, U.; SHENOY, P.; SAHU, S.; SHAIKH, A. A cost-aware elasticity provisioning system for the cloud. In: . ICDCS '11. Washington, DC, USA: IEEE Computer Society, c2011. p. 559–570.

[16] SULEIMAN, B.; SAKR, S.; JEFFERY, R.; LIU, A. On understanding the economics and elasticity challenges of deploying business applications on public cloud infrastructure. *Journal of Internet Services and Applications*, v. 3, p. 173–193, 2012.

[17] YOUNGE, A.; VON LASZEWSKI, G.; WANG, L.; LOPEZ-ALARCON, S.; CARITHERS, W. Efficient resource management for cloud computing environments. In: . c2010. p. 357–364.

[18] GALANTE, G.; BONA, L. C. E. D. A survey on cloud computing elasticity. In: . UCC '12. Washington, DC, USA: IEEE Computer Society, c2012. p. 263–270.

[19] BAUER, E.; ADAMS, R. *Capacity and elasticity*. John Wiley and Sons, Inc., 2012. p. 296–310.

[20] BEERNAERT, L.; MATOS, M.; VILACA, R.; OLIVEIRA, R. Automatic elasticity in openstack. In: . SDMCMM '12. New York, NY, USA: ACM, c2012. p. 2:1–2:6.

[21] RAVEENDRAN, A.; BICER, T.; AGRAWAL, G. A framework for elastic execution of existing mpi programs. In: . IPDPSW '11. Washington, DC, USA: IEEE Computer Society, c2011. p. 940–947.

[22] MORENO, I.; XU, J. Customer-aware resource overallocation to improve energy efficiency in realtime cloud computing data centers. In: . c2011. p. 1 –8.

[23] ZHANG, X.; SHAE, Z.-Y.; ZHENG, S.; JAMJOOM, H. Virtual machine migration in an over-committed cloud. In: . c2012. p. 196 –203.

[24] BRYANT, R.; TUMANOV, A.; IRZAK, O.; SCANNELL, A.; JOSHI, K.; HILTUNEN, M.; LAGAR-CAVILLA, A.; DE LARA, E. Kaleidoscope: cloud micro-elasticity via vm state coloring. In: . EuroSys '11. New York, NY, USA: ACM, c2011. p. 273–286.

[25] MIHAILESCU, M.; TEO, Y. M. The impact of user rationality in federated clouds. *Cluster Computing and the Grid, IEEE International Symposium on*, Los Alamitos, CA, USA, v. 0, p. 620–627, 2012.

[26] DAWOUD, W.; TAKOUNA, I.; MEINEL, C. Elastic virtual machine for fine-grained cloud resource provisioning. In: KRISHNA, P.; BABU, M.; ARIWA, E. (Eds.) *Global Trends in Computing and Communication Systems*. Springer Berlin Heidelberg, 2012. v. 269 of *Communications in Computer and Information Science*, p. 11–25.

[27] SULEIMAN, B. Elasticity economics of cloud-based applications. In: . SCC '12. Washington, DC, USA: IEEE Computer Society, c2012. p. 694–695.

[28] KNAUTH, T.; FETZER, C. Scaling non-elastic applications using virtual machines. In: . c2011. p. 468 –475.

[29] MARSHALL, P.; KEAHEY, K.; FREEMAN, T. Elastic site: Using clouds to elastically extend site resources. In: . CCGRID '10. Washington, DC, USA: IEEE Computer Society, c2010. p. 43–52.

[30] HAN, R.; GUO, L.; GHANEM, M. M.; GUO, Y. Lightweight resource scaling for cloud applications. *Cluster Computing and the Grid, IEEE International Symposium on*, Los Alamitos, CA, USA, v. 0, p. 644–651, 2012.

[31] WOOD, T.; SHENOY, P.; VENKATARAMANI, A.; YOUSIF, M. Black-box and gray-box strategies for virtual machine migration. In: . NSDI'07. Berkeley, CA, USA: USENIX Association, c2007. p. 17–17.

[32] SHEN, Z.; SUBBIAH, S.; GU, X.; WILKES, J. Cloudscale: elastic resource scaling for multi-tenant cloud systems. In: . SOCC '11. New York, NY, USA: ACM, c2011. p. 5:1–5:14.

[33] LI, X.; DU, J. Adaptive and attribute-based trust model for service level agreement guarantee in cloud computing. *Information Security, IET*, v. 7, n. 1, p. 39–50, 2013.

[34] SURHONE, L.; TENNOE, M.; HENSSONOW, S. *Rightscale*. VDM Publishing, 2010.

[35] RAJAN, D.; CANINO, A.; IZAGUIRRE, J. A.; THAIN, D. Converting a high performance application to an elastic cloud application. In: . CLOUDCOM '11. Washington, DC, USA: IEEE Computer Society, c2011. p. 383–390.

[36] MILOJICIC, D.; LLORENTE, I. M.; MONTERO, R. S. Opennebula: A cloud management tool. *Internet Computing, IEEE*, v. 15, n. 2, p. 11 –14, march-april 2011.

[37] LONEA, A.; POPESCU, D.; PROSTEAN, O. A survey of management interfaces for eucalyptus cloud. In: . c2012. p. 261 –266.

[38] WEN, X.; GU, G.; LI, Q.; GAO, Y.; ZHANG, X. Comparison of open-source cloud management platforms: Openstack and opennebula. In: . c2012. p. 2457 –2461.

[39] CAI, B.; XU, F.; YE, F.; ZHOU, W. Research and application of migrating legacy systems to the private cloud platform with cloudstack. In: . c2012. p. 400 –404.

[40] DE OLIVEIRA, R.; DA SILVA CARISSIMI, A.; TOSCANI, S. *Sistemas operacionais - vol. 11: Série livros didáticos informática ufrgs*. Livros didáticos. Bookman, 2010.

[41] BANIKAZEMI, M.; OLSHEFSKI, D.; SHAIKH, A.; TRACEY, J.; WANG, G. Meridian: an sdn platform for cloud network services. *Communications Magazine, IEEE*, v. 51, n. 2, p. 120–127, 2013.

[42] FU, L.; GONDI, C. Cloud computing hosting. In: . c2010. v. 3. p. 194–198.

[43] FUJII, T.; KIMURA, M. Analysis results on productivity variation in force.com applications. In: . c2011. p. 314 –317.

[44] COSTA, R.; BRASILEIRO, F.; DE SOUZA FILHO, G. L.; SOUSA, D. M. Just in Time Clouds: Enabling Highly-Elastic Public Clouds over Low Scale Amortized Resources. Technical report, Universidade Federal de Campina Grande, 2010.

[45] TRAN, N.-L.; SKHIRI, S.; ZIMáNYI, E. Eqs: An elastic and scalable message queue for the cloud. In: . CLOUDCOM '11. Washington, DC, USA: IEEE Computer Society, c2011. p. 391–398.

[46] ZHANG, X.; JEONG, S.; KUNJITHAPATHAM, A.; GIBBS, S. Towards an elastic application model for augmenting computing capabilities of mobile platforms. In: CAI, Y.; MAGEDANZ, T.; LI, M.; XIA, J.; GIANNELLI, C. (Eds.) *Mobile Wireless Middleware, Operating Systems, and Applications*. Springer Berlin Heidelberg, 2010. v. 48 of *Lecture Notes of the Institute for Computer Sciences, Social Informatics and Telecommunications Engineering*, p. 161–174.

[47] MATOS, M.; CORREIA, JR., A.; PEREIRA, J.; OLIVEIRA, R. Serpentine: adaptive middleware for complex heterogeneous distributed systems. In: . SAC '08. New York, NY, USA: ACM, c2008. p. 2219–2223.

[48] TANENBAUM, A. *Computer networks*. 4th. ed. Upper Saddle River, New Jersey: Prentice Hall PTR, 2003.

[49] KUMAR, K.; FENG, J.; NIMMAGADDA, Y.; LU, Y.-H. Resource allocation for real-time tasks using cloud computing. In: . c2011. p. 1 –7.

[50] MICHON, E.; GOSSA, J.; GENAUD, S. Free elasticity and free cpu power for scientific workloads on iaas clouds. In: . c2012. p. 85 –92.

[51] DUSTDAR, S.; GUO, Y.; HAN, R.; SATZGER, B.; TRUONG, H.-L. Programming directives for elastic computing. *Internet Computing, IEEE*, v. 16, n. 6, p. 72 –77, nov.-dec. 2012.