Provisionamento Automático de Recursos em Nuvem IaaS: eficiência e limitações de abordagens reativas

Fábio Morais[1], **Raquel Lopes**[1], **Francisco Brasileiro**[1]

[1]Universidade Federal de Campina Grande (UFCG)
Laboratório de Sistemas Distribuídos, Campina Grande – PB – Brasil

fabio@lsd.ufcg.edu.br
{raquel,fubica}@dsc.ufcg.edu.br

Abstract. *Public cloud providers such as Amazon AWS and Rackspace offer programmable reactive auto-scaling services. This service addresses long running applications with time varying demands and works as follows: the client configures thresholds related to some application metric and when these thresholds are reached, actions are triggered in order to add or release resources from this application. For instance, we can configure the action of adding 2 new VMs whenever the average CPU utilization of the VMs running the application is greater than 70%. In this paper we perform an in-depth analysis of this type os reactive auto-scaling service, identifying its efficiency and limitations.*

Resumo. *Provedores públicos de Computação na Nuvem como Amazon AWS e Rackspace oferecem serviços programáveis de provisionamento automático e reativo de recursos. Esse serviços tratam de aplicações, de longa duração com demandas que variam no tempo, da seguinte forma: o cliente configura limiares relacionados a alguma métrica da aplicação e quando esses limiares são atingidos, ações são disparadas para adicionar ou liberar recursos da aplicação. Por exemplo, é possível configurar a ação de adicionar 2 novas VMs sempre que a utilização média de CPU das VMs executando a aplicação for maior que 70%. Nesse artigo realizamos uma análise aprofundada sobre esse tipo de serviço de provisionamento automático e reativo, identificando eficiências e limitações.*

1. Introdução

Provedores de Computação na Nuvem que oferecem infraestrutura como serviço (IaaS) são ambientes adequados para executar aplicações horizontalmente escaláveis. Normalmente, esse tipo de aplicação possui carga de trabalho variável no tempo e seu desempenho pode ser ajustado alterando o número de nós computacionais alocados para executá-la. As duas principais características de IaaS que incentivam a execução destas aplicações nestes ambientes são: (i) elasticidade da infraestrutura de execução; e (ii) modelo de tarifação em que se paga conforme a utilização do serviço (*pay-as-you-go*), permitindo que os provedores de aplicações paguem apenas pelos recursos adquiridos. Na prática, esses recursos são oferecidos no formato de máquinas virtuais (VMs) adquiridas pelos provedores das aplicações que são os usuários dos provedores IaaS.

O provisionamento automático de uma aplicação horizontalmente escalável consiste no ato de decidir periodicamente a quantidade de VMs necessárias para executar a aplicação no próximo intervalo de tempo de curto prazo (na ordem de minutos). Isso é realizado com o intuito de alcançar a qualidade de serviço (QoS) desejada para a aplicação,

ao mesmo tempo que os custos de infraestrutura são minimizados. Apesar da flexibilidade fornecida por ambientes de IaaS, o provisionamento automático ideal de uma aplicação desse tipo não é uma tarefa trivial. Primeiramente, é preciso antecipar a capacidade mínima requerida pela aplicação em um futuro próximo. Em seguida, decidir a quantidade de VMs necessária para atender a demanda da aplicação. Finalmente, essas decisões precisam ser continuamente realizadas para lidar com a variabilidade da carga de trabalho da aplicação ao longo do tempo.

Considerando um cenário em que o *provisionamento automático de recursos é oferecido como um serviço de IaaS*, a solução de provisionamento deve ser responsável por gerenciar a infraestrutura de execução da aplicação visando que esta atinja a QoS desejada usando a menor quantidade de recursos possível (*trade-off* custo/desempenho). Um outro aspecto importante deste serviço trata de questões de privacidade; espera-se que esse serviço utilize informações não-específicas da aplicação que podem ser coletadas no nível da infraestrutura de execução (como utilização de CPU, Memória, etc). Em geral, informações obtidas da aplicação (e.g. taxa de chegada de requisições, tipo de requisições, tamanhos de filas, etc) são consideradas sensíveis, não sendo facilmente compartilhadas.

O serviço de provisionamento segue um dos seguintes modos de operação: reativo ou proativo. O serviço proativo tenta antecipar a carga da aplicação para tomar decisões sobre a capacidade adequada. Já a técnica reativa, que é o foco deste artigo, consiste em uma reação programável às mudanças percebidas na aplicação e/ou na sua infraestrutura. Essencialmente, essa abordagem utiliza um conjunto de regras de provisionamento para decidir quando e em qual quantidade a aplicação deve ser provisionada [Lorido-Botrán et al. 2014]. O provisionamento reativo utiliza apenas informações sobre o estado atual da aplicação e seu ambiente para decidir sobre o provisionamento da aplicação.

Espera-se que os sistemas reativos não sejam eficientes ao prover aplicações com cargas de trabalho de intensa variabilidade no tempo [Lorido-Botrán et al. 2014]. Isso decorre da natureza reativa e pontual da solução e do fato da configuração das regras serem consideravelmente sensíveis a mudanças e tendências da carga de trabalho da aplicação, que geram a necessidade de ajustes frequentes mesmo na presença de um especialista na aplicação atuando no processo de configuração [Lorido-Botrán et al. 2014]. Desta forma, prováveis equívocos na configuração das regras podem provocar situações de sub-provisionamento que levam à degradação da QoS da aplicação e possíveis violações de SLO (*Service Level Objective*), ou de super-provisionamento, com o desperdício de recursos adquiridos do provedor de IaaS. Outro ponto negativo é que em geral as soluções reativas usam métricas intrusivas, específicas da aplicação, que não podem ser consideradas por um serviço de provisionamento genérico e automático em IaaS.

Apesar de criticada em termos de desempenho, a abordagem reativa ainda é muito explorada devido sua simplicidade e natureza intuitiva [Lorido-Botrán et al. 2014, da Rosa Righi et al. 2017]. Os principais provedores do mercado de Computação na Nuvem e IaaS, como Amazon Web Services (AWS) [Amazon 2016], Rackspace [Rackspace 2016] e Microsoft Azure [Azure 2016], oferecem serviços de provisionamento automático e reativo de recursos[1]. Esta abordagem também é amplamente explorada na

[1]A Google [Google 2016] oferece um serviço de provisionamento automático que também baseia-se na técnica reativa, mas utiliza um modelo de provisionamento mais sofisticado, não conhecido pelo público, para decidir a quantidade de VMs que devem ser provisionadas a cada intervalo de tempo.

literatura através do uso de diferentes conjuntos de métricas de desempenho, configurações de limiares e ações de provisionamento [Lim et al. 2009, Calheiros et al. 2012, Fito et al. 2010, Calcavecchia et al. 2012, Marshall et al. 2010, Bonvin et al. 2011, Ghanbari et al. 2011, Seung et al. 2011].

Neste artigo, avaliamos de forma profunda o desempenho de soluções reativas que utilizam apenas métricas de uso da infraestrutura e, portanto, podem ser implementadas como um serviço de provisionamento automático. O nosso objetivo é avaliar o desempenho de soluções reativas em termos de custo de execução e nível de QoS da aplicação provisionada, além de analisar o desempenho destas soluções em relação à aplicabilidade da técnica reativa no cenário de provisionamento automático como um serviço.

O restante deste artigo está estruturado da seguinte forma. A seguir a problemática do provisionamento automático como um serviço é descrito e detalhado (Seção 2). Na Seção 3 é realizado um levantamento sobre possíveis abordagens reativas do mercado e da literatura que podem ser empregadas como soluções de provisionamento automático em IaaS. Uma análise de desempenho de abordagens de provisionamento reativo segundo diferentes aspectos é realizada na Seção 4. Por fim, discute-se sobre os resultados obtidos (Seção 5) e conclui-se o trabalho com direcionamentos para futuros trabalhos (Seção 6).

2. Provisionamento Automático como um Serviço

O cenário de provisionamento automático como um serviço é composto essencialmente por quatro atores: (i) o provedor de IaaS; (ii) o provedor da aplicação a ser executada no ambiente de IaaS; (iii) o usuário final da aplicação provisionada que faz uso do serviço prestado pela mesma; e (iv) o provedor do serviço de provisionamento automático de recursos. A relação entre esses atores está representada na Figura 1.

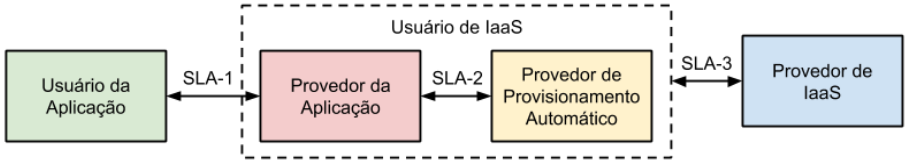

Figura 1. Visão geral da relação entre atores do serviço de provisionamento automático de recursos em ambientes de IaaS.

Tipicamente, o provedor de uma aplicação horizontalmente escalável adquire VMs de um provedor de IaaS para executar de forma dedicada a sua aplicação. Assim, o provedor da aplicação é usuário do serviço de IaaS, criando-se uma relação de serviço regida por um contrato de nível de serviço (SLA), identificado por SLA-3 na Figura 1. O provedor de IaaS garante (i) a aquisição e criação de VMs por parte de seu usuário e (ii) a disponibilidade de acesso a essas VMs, sob o risco de pagar penalidades. O serviço é tarifado segundo valores previamente definidos para cada intervalo de tempo de uso das VMs adquiridas e para os tipos de VMs adquiridas. Em um modelo público de IaaS, por exemplo, os provedores cobram um preço fixo por VM usada em um período menor ou igual ao ciclo de tarifação praticado pelo provedor (normalmente com duração de 1 hora).

A aplicação que executa no ambiente de IaaS é acessada remotamente pelos seus usuários e também deve haver um contrato que rege esse "serviço". Nesse caso o SLA-1 presente na Figura 1 indica o contrato estabelecido entre o provedor da aplicação e

os usuários da aplicação. Esses SLAs são responsáveis por garantir que os usuários da aplicação tenham acesso a um serviço com o nível de qualidade de serviço desejado.

O cenário deste estudo considera um quarto ator, que consiste em um serviço de provisionamento automático que é responsável por gerenciar os recursos alocados para executar a aplicação. Esse serviço atua como um procurador do responsável da aplicação junto ao provedor de IaaS, tendo o papel de adquirir e liberar recursos (tipicamente VMs) quando necessário, tornando-se também um usuário do serviço de IaaS.

O serviço de provisionamento automático realiza periodicamente o planejamento da capacidade da infra-estrutura (quantidade de VMs) para acomodar as flutuações da carga de trabalho da aplicação[2]. Para tal, esse serviço realiza as ações de provisionamento necessárias para cumprir o planejamento realizado. Essas flutuações de carga são vistas pelo serviço de provisionamento como variações na utilização dos diferentes tipos de recursos (CPU, memória, etc) executando a aplicação. Diferentes tipos de instância de VM podem ser selecionados para executar a aplicação durante sua execução [Morais et al. 2016], mas assume-se nesse estudo que apenas um tipo de instância é usado.

Quando a estratégia de provisionamento falha, decidindo por uma capacidade menor do que a necessária, o resultado é a degradação da QoS da aplicação e, possivelmente, perdas econômicas para o provedor da aplicação devido ao descumprimento do SLA-1. Isto acontece uma vez que os recursos ficam sobre-utilizados. O contrato entre o provedor da aplicação e do serviço de provisionamento automático (SLA-2 na Figura 1) deve considerar limiares adequados de uso dos recursos que levam à QoS desejada da aplicação. Pois quando esses limiares são atingidos, quanto maior for a utilização de um recurso menor será a QoS da aplicação, já que haverá mais competição pelo uso do recurso. Esses limiares definidos no SLA-2, quando descumpridos, conduzem a uma situação de saturação e queda da QoS da aplicação mencionada anteriormente.

Os SLAs entre os provedores das aplicações e o serviço de provisionamento (SLA-2) definem SLOs por meio de limiares de utilização dos recursos em uso para executar a aplicação. Por exemplo, um SLO pode definir que a utilização de CPU das VMs executando a aplicação não deve ultrapassar 80%. Assim, violações dos SLOs definidos no SLA-2 podem gerar degradação da QoS da aplicação, que é possivelmente percebida pelo usuário final. Se a utilização de recursos for mantida abaixo, mas o mais próximo possível do limiar estabelecido no SLO, os SLAs da aplicação (SLA-1) são satisfeitos. Além disso, quanto mais próximo dos limiares estabelecidos no SLA-2 estiverem as utilizações dos recursos que executam a aplicação, menor será o custo de execução da aplicação. Assim, o objetivo do serviço de provisionamento é manter os recursos alocados com utilizações mais próximas possíveis porém menores que o estabelecido nos SLOs do SLA-2.

É importante ressaltar que o serviço de provisionamento considerado deve ser capaz de prover recursos automaticamente para diferentes aplicações com o mínimo grau de intrusividade. Por isso, ele usa informações não específicas da aplicação. Além disso, o serviço deve buscar garantir a QoS desejada para a aplicação ao mesmo tempo que reduz o custo de execução da aplicação quando há variação em sua carga de trabalho.

[2]Para que este serviço de provisionamento automático seja possível, considera-se um sistema de monitoramento que coleta periodicamente a utilização dos recursos das VMs ativas que executam a aplicação.

3. Trabalhos Relacionados

O provisionamento reativo consiste na principal técnica de provisionamento utilizada pelo mercado de Computação na Nuvem e está majoritariamente presente nas soluções comerciais de IaaS [Amazon 2016, Rackspace 2016, Azure 2016]. A técnica reativa também é amplamente abordada pela literatura através do provisionamento automático baseado em informações específicas da aplicação (por exemplo, tempos de resposta, taxa de chegada de requisições, tipos de requisições, etc) [Calheiros et al. 2012, Fito et al. 2010, Calcavecchia et al. 2012, Marshall et al. 2010, Bonvin et al. 2011, Seung et al. 2011], que possuem uma relação direta com a QoS da aplicação. Acreditamos que um serviço de provisionamento automático precisa lidar apenas com métricas não-intrusivas obtidas, no nível da infraestrutura virtual, o que inviabiliza o uso dessas soluções para este fim.

As soluções reativas não-intrusivas que conhecemos [Ghanbari et al. 2011, Lim et al. 2009] assumem que a técnica simplesmente baseada em limiares inicialmente definidos não é suficiente para assegurar os objetivos de provisionamento, apesar de não haver nenhuma avaliação mais consistente deste fato. Por esse motivo, essas soluções combinam o uso de regras de provisionamento com outras técnicas de tomada de decisão e atualização de limiares. Ghanbari et al. combinam a técnica de regras reativas a um processo de voto, em que as VMs provisionadas devem concordar majoritariamente sobre as ações de provisionamento a serem realizadas. A solução de Lim et al. utiliza limiares proporcionais que são dinamicamente modificados para reduzir o impacto do provisionamento sob a utilização de recursos da infraestrutura e atenuar as limitações do uso de regras estaticamente definidas. Com objetivo semelhante, o trabalho de Netto et al. propõe o ajuste dinâmico da quantidade de VMs provisionadas por ação de provisionamento realizada [Netto et al. 2014].

Todavia, até onde se sabe não existe um estudo aprofundado sobre o desempenho de abordagens reativas de provisionamento automático a partir de métricas não-intrusivas, apesar da difusão dessa técnica no mercado de Nuvem e IaaS. Nesse sentido, este trabalho tem como objetivo analisar criteriosamente a eficiência e as limitações do uso dessa abordagem em um cenário de provisionamento automático como um serviço de IaaS.

4. Análise de Desempenho de Abordagens Reativas

Na prática, a abordagem reativa é a principal técnica de provisionamento atualmente em uso no mercado de Computação na Nuvem. Essa abordagem atua reagindo a mudanças na carga de trabalho da aplicação, que podem ser percebidas através das métricas monitoradas da aplicação ou da infraestrutura. As regras de provisionamento definem ações a serem tomadas em reação a mudanças. Em um cenário de provisionamento não-intrusivo, essas regras fazem uso dos limiares (*threshold-based*) de utilização da infraestrutura de execução (por exemplo, percentual de utilização de CPU) definidos no SLA-2 (Seção 2).

A configuração de cada regra é composta de duas partes essenciais. A primeira parte define a condição para disparo de uma ação de provisionamento. Esta parte é composta por um conjunto de triplas $\langle$métrica, limiar, operador condicional$\rangle$ que definem as condições para o disparo de ações de provisionamento. Por exemplo, uma ação é disparada se a métrica de utilização média de CPU for maior igual ao limiar de 70%. A segunda parte consiste na ação de provisionamento associada à condição de provisionamento definida; por exemplo, aumento da capacidade computacional da infraestrutura de

execução. Assim, quando uma condição de provisionamento for satisfeita (primeira parte) então uma ação de provisionamento associada (segunda parte) será disparada.

Pelo menos duas regras devem ser definidas: uma para adicionar recursos à infraestrutura de execução quando necessário e outra para remover recursos que não estão mais em uso. Em um ambiente de IaaS uma ação de provisionamento consiste na definição da quantidade fixa de VMs que deve ser adicionada ou removida da infraestrutura que executa a aplicação caso uma condição de disparo de ação seja satisfeita. Esse tipo de solução funciona como modelos de laço de controle que periodicamente avaliam se alguma condição de provisionamento foi satisfeita, e em caso positivo a ação associada é executada. Isso permite que em uma ambiente de IaaS a infraestrutura de execução possa ser dinâmica e automaticamente ajustada a partir das regras previamente definidas pelo responsável da aplicação, que também é um usuário da solução de provisionamento.

Nesse artigo, avaliamos o provisionamento automático e reativo como um serviço segundo os seguintes aspectos: (i) capacidade de manter a QoS da aplicação no nível desejado, idealmente não violando os SLOs da aplicação; (ii) capacidade de provisionar a aplicação com custos de execução próximos aos praticados por um serviço de provisionamento perfeito, livre de violações de SLO e com custo de execução mínimo; (iii) grau de generalidade em relação às aplicações provisionadas e a independência de características da carga de trabalho destas; e (iv) grau de controle permitindo que configurações diferentes lidem com o *trade-off* custo/QoS. Quanto maior o nível de QoS desejado, maior deve ser o custo e vice versa. É importante que o serviço ofereça esse controle ao seu usuário.

4.1. Modelo e dados de simulação

O modelo de simulação foi implementado na linguagem de programação R[3] [Chambers 2016]. Este modelo simula a interação entre o serviço de provisionamento automático, a aplicação provisionada, a infraestrutura virtual adquirida do provedor de IaaS e os componentes de monitoramento e atuação disponibilizados pelos provedores de IaaS. Dado o viés não intrusivo da solução de provisionamento, o componente de provisionamento opera desassociadamente da aplicação provisionada, de tal forma que as interações com as aplicações restringem-se à alocação ou liberação de VMs e à coleta de dados de utilização de recursos no nível da infraestrutura virtual alocada.

Um rastro de utilização consiste em uma série temporal da utilização de recursos de uma aplicação, onde cada item da série corresponde à média de utilização de CPU para o intervalo de tempo considerado. Cada item no rastro consiste em uma dupla $\langle x, y \rangle$, onde x é a utilização média de CPU no intervalo de tempo considerado ($x \in \mathbb{R}, 0 \leq x \leq 1$) e y é a quantidade de núcleos de CPU alocados à aplicação durante esse tempo ($y \in \mathbb{R}^+$). A demanda média da aplicação (u, $u \in \mathbb{R}^+$), em núcleos de CPU, é calculada com base nos dados de utilização média e alocação de CPU, dada por $u = x \times y$.

Nas simulações, a capacidade da infraestrutura alocada para executar a aplicação em número de núcleos de CPU é equivalente a quantidade de VMs alocadas, uma vez que o modelo de simulação considera VMs com apenas 1 núcleo de CPU. Desta forma, a utilização real simulada de cada máquina virtual entre o intervalo de tempo τ e $\tau + 1$ é computada como sendo o mínimo entre 100% e $\frac{u}{a}$, onde a é o número de VMs alocadas

[3]O código fonte do simulador encontra-se disponível em https://goo.gl/8a39zi

para executar a aplicação no experimento de simulação. Logo, cada VM alocada apresenta a mesma média de utilização para um mesmo intervalo de tempo τ. Assim, se a utilização de CPU no intervalo de tempo τ é maior do que o limite de utilização de CPU definido no SLO da aplicação, então ocorre uma violação de SLO, indicando que a capacidade a de CPU atribuída à aplicação não foi suficiente. O sistema de provisionamento opera com periodicidade configurável, de forma que a cada laço de controle pelo menos um novo item de dado de utilização de CPU é lido do rastro e esse dado é usado para computar a utilização real de CPU das VMs alocadas para o próximo intervalo de tempo.

O serviço de provisionamento simulado realiza o planejamento de capacidade da infraestrutura de execução para o próximo intervalo de tempo com base em regras de provisionamento reativo predefinidas. Desta forma a alocação de recursos ocorre em conformidade com as demandas simuladas da aplicação que são identificadas e supridas a partir da técnica de provisionamento. Finalmente, ao realizar a liberação de recursos da infraestrutura o simulador considera o modelo de IaaS operado pela Amazon, em que as VMs são tarifadas por hora. Assim, mesmo que o sistema de provisionamento decida pela desalocação de VMs em um dado momento, o desligamento de uma VM só é de fato efetuado se essa decisão ocorrer em sincronia com horas completas de uso dessa VM.

As simulações são alimentadas com rastros de utilização de CPU de 30 aplicações reais com duração média de 8 meses. Cada rastro consiste na medição da média de utilização de recursos de CPU produzida por uma única aplicação, monitorada a cada intervalo de 5 minutos. No período em que esses rastros foram capturados as aplicações estavam superprovidas de recursos de forma estática (a mesma infraestrutura de tamanho fixo executou a aplicação). Esses dados são provenientes de uma parceria realizada com a HP para o desenvolvimento de soluções de provisionamento automático de recursos em ambientes de IaaS[4] [Morais et al. 2013]. A função de distribuição acumulada (FDA) da utilização de CPU dessas aplicações é apresentada na Figura 2. A partir desta, percebe-se que os dados apresentam uma ampla variedade de distribuições de utilização entre as aplicações, com diferentes perfis de consumo de CPU. Além do mais, para uma mesma aplicação existe uma considerável variabilidade de intensidades de utilização de CPU, onde pelo menos metade das aplicações apresentam um desvio padrão da utilização maior que 15%. Dessa forma, considera-se que os dados utilização considerados são representativos para o estudo do provisionamento automático e reativo.

4.2. Provisionamento reativo perfeito

Inicialmente, analisamos a viabilidade de uma solução de provisionamento reativo que atue de forma perfeita, sem erros de provisionamento durante toda a execução da aplicação. Para tal, faz-se necessário que o serviço de provisionamento possua informações exatas sobre as demandas futuras de CPU da aplicação a cada ciclo de controle (intervalo de 5 minutos) para que o planejador de capacidade perfeito decida e aloque a quantidade necessária de VMs (com 1 núcleo de CPU), para assegurar os SLOs definidos para a aplicação com o mínimo custo. Um limiar de utilização de CPU de 100% foi considerado no SLO do serviço de provisionamento (SLA-2). Desta forma, a simulação resulta em um rastro da quantidade ideal de VMs alocadas no tempo para cada aplicação provisionada, assegurando custo mínimo e ausência de violações de SLO.

[4]Infelizmente, devido a questões de confidencialidade esses dados não encontram-se publicados.

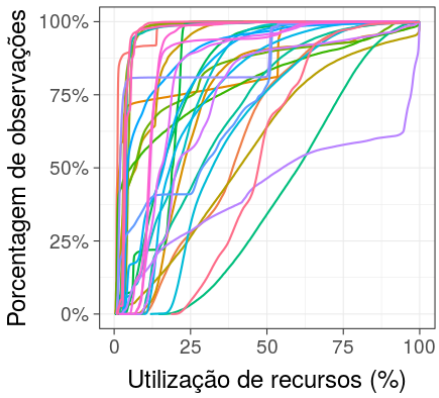

Figura 2. FDA da utilização de CPU das aplicações consideradas.

A partir desses rastros foi realizada uma análise *post-mortem* para identificar a partir das ações de provisionamento realizadas, quais seriam as regras de provisionamento reativo mais apropriadas. Consideramos que cada ação de provisionamento necessita de no máximo 5 minutos para ser efetivada, tempo referente a capacidade de reação da técnica de provisionamento e em conformidade com a mínima periodicidade disponibilizada pelos dados de simulação. Assim, para uma ação de provisionamento ocorrida no tempo τ a configuração de provisionamento necessária para realizar essa ação é computada com base em dados obtidos no tempo $t - 2$. Por exemplo, se no intervalo de tempo τ, 2 VMs foram acrescentadas pelo serviço perfeito, então a análise *post-mortem* identifica qual seria a regra de provisionamento disparada em $t - 2$ que resultaria nessa mesma decisão.

Com base nas regras descobertas nós verificamos a viabilidade de se configurar um serviço reativo de provisionamento que seja próximo ao ótimo. Isto é possível se as mesmas regras (ou regras semelhantes) se repetem para uma aplicação e, idealmente, para várias aplicações. Não foi esse o resultado encontrado, como veremos nas seções a seguir.

4.2.1. Não há predominância de configuração de limiares

Com base nas configurações inferidas do processo de provisionamento perfeito identificamos todas as diferentes regras necessárias para gerar o provisionamento perfeito. A fim de reduzir o espaço de possibilidades de configuração, os valores dos limiares inferidos a partir do provisionamento perfeito foram sumariados em faixas de valores com tamanho de 5% (por exemplo, um limiar de 31% passa a pertencer a faixa de 35%). Desta forma, os limiares originalmente inferidos foram categorizados em 20 grupos de limiares, com valores entre 5% e 100%, em passos de 5%.

Computamos a frequência de diferentes limiares nas regras de provisionamento de cada uma das aplicações, agrupados por ação de provisionamento. A Figura 3 (à esquerda) mostra o diagrama de caixa da frequência de uso do limiar mais frequente no provisionamento de cada aplicação. Observa-se que para 50% das aplicações provisionadas a mediana de frequência dos limiares mais predominantes foi de apenas aproximadamente 17%, para ambos os tipos de ação de provisionamento. Além da não predominância de um limiar por aplicação, também observa-se uma quantidade significativa de diferentes configurações de limiares que são usadas por aplicação. Em média, 17 diferentes confi-

gurações são necessárias por aplicação, como apresentado na Figura 3 (à direita).

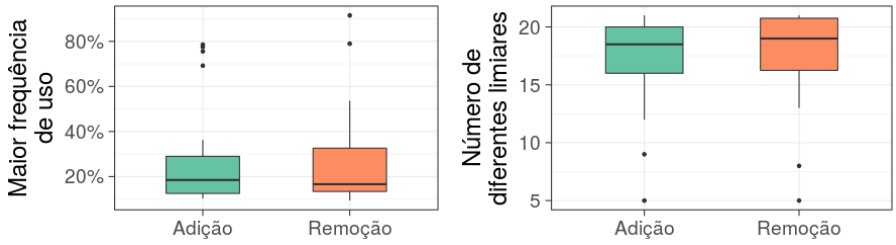

Figura 3. Análise por aplicação da frequência de uso do limiar mais comum (à esq.) e do número de diferentes configurações de limiares por aplicação (à dir.).

Desta forma, no que se refere à predominância de configuração no provisionamento reativo perfeito, a frequência de uso de configurações de limiares e da quantidade de diferentes configurações por aplicação observadas evidenciam a dificuldade de uso da abordagem reativa para gerar um cenário de provisionamento perfeito dessas aplicações. Em adição à não predominância de regras específicas, regras incompatíveis foram muitas vezes identificadas, como por exemplo, o mesmo limiar era associado ora à ação de remoção, ora à ação de adição.

4.2.2. Não há padrão em transições de limiares usados consecutivamente

Um outro aspecto sobre a configuração de limiares para o provisionamento perfeito dessas aplicações consiste na variabilidade temporal dos limiares das regras aplicadas no provisionamento. Além da necessidade de decidir quais as configurações de limiares devem ser usadas para cada aplicação, também deve-se conhecer como ocorrem as mudanças de uma configuração para outra diferente ao longo do provisionamento. A Figura 4 (à esquerda) apresenta o percentual de ocorrência de transições entre limiares de utilização diferentes, ou seja a frequência de mudança de uso de limiares entre intervalos de provisionamento perfeito. Para metade das aplicações, em 77% das ações de provisionamento consecutivas ocorrem mudanças no limiar da regra de provisionamento utilizada.

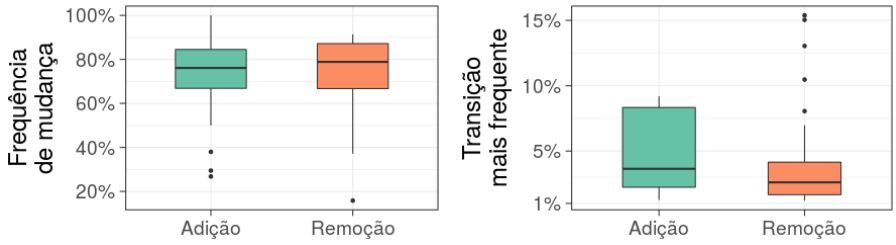

Figura 4. Análise da frequência de transições entre diferentes limiares de utilização (à esq.) e da frequência da transição mais recorrente no provisionamento por aplicação (à dir.)

Uma outra questão que buscamos responder diz respeito a padrões de transição de um limiar para outro usados em sequência. Verificamos que a frequência de recorrência de uma mesma transição entre regras não é considerada significativa. Como pode ser visto

na Figura 4 (à direita), para todos os cenários, em 90% das mudanças de limiares mais recorrentes por aplicação a frequência de observação dessas transições é menor que 9% do total de transições realizadas. Desta forma, observa-se que o percentual de ocorrência de uma mesma transição entre limiares também desfavorece a eficiência de configuração de uma solução perfeita de provisionamento reativo baseado em utilização de CPU.

4.2.3. Existe uma relação forte entre carga de trabalho e ações de provisionamento

Realizamos uma análise para entender a relação entre a variação da utilização de CPU medida na infraestrutura de execução e a quantidade de operações de provisionamento necessárias no provisionamento perfeito. Calculamos de forma pareada a correlação entre o desvio padrão do consumo de CPU por aplicação e a quantidade de operações de provisionamento realizadas no provisionamento reativo perfeito de cada aplicação. Os coeficientes de Spearman $(0,91)$ e Kendall $(0,75)$ mostram que essa relação é de forte a muito forte para as aplicações consideradas. Ou seja, quanto maior for a variabilidade da carga de trabalho da aplicação maior será a necessidade de realizar operações de provisionamento para a abordagem perfeita.

Além disso, também existe uma correlação significativa entre a variabilidade de carga de trabalho e o número de diferentes configurações de limiares de provisionamento. Os coeficientes de correlação de Spearman e Kendall entre o desvio padrão do uso de CPU e a quantidade de diferentes limiares usados no provisionamento perfeito apresentam correlações com valor de $0,61$ e de $0,45$, respectivamente. Assim, assume-se que a configuração dos limiares de provisionamento são dependentes de características específicas da carga de trabalho das aplicações provisionadas. Esse resultado limita significativamente o desempenho da abordagem reativa em relação à eficiência e à capacidade de generalidade de configuração para um conjunto heterogêneo de aplicações.

4.3. Abordagens reativas na prática

Os objetivos de provisionamento automático de aplicações em ambientes de IaaS envolvem um *trade-off* entre a redução do custo de provisionamento e a redução do número de violações de SLO. Em um cenário de provisionamento perfeito esses objetivos conflitantes são otimizados, gerando uma execução da aplicação sem violações de SLO com o menor custo possível de execução. O controle desses objetivos pode ser realizado a partir da configuração dos limiares de provisionamento. Quanto maior for o limiar de provisionamento, seja ele de adição ou remoção, prioriza-se a redução de custos de execução e consequentemente aumenta-se a chance de violações de SLO. Por outro lado, quanto menores forem os limiares, maior será o custo de execução da aplicação, já que VMs são adicionadas ao primeiro sinal de aumento de utilização e só são removidas quando a utilização está suficientemente baixa. Por consequência, a probabilidade de ocorrência de violações nesse caso é menor.

Essa relação fica evidente ao analisarmos o provisionamento reativo considerando diferentes configurações de limiares. Simulamos cenários de provisionamento com limiares de adição de VMs variando de 20% a 90%, em passos de 10%, com limiares de remoção a uma distância absoluta do limiar de adição variando de 10% a 80%, também

em passos de 10% [5] e ações de provisionamento que correspondem a adição ou remoção de uma VM com um núcleo de CPU. A Figura 5 apresenta o resultado em termos de violações de SLO[6] e custo do serviço reativo com as diferentes configurações. No eixo X são apresentados os diversos limiares de adição usados e a cor de cada diagrama de caixa indica o limiar de remoção usado (ver os rótulos acima do gráfico). O *trade-off* fica evidente nos resultados: para os menores limiares de adição/remoção temos os maiores custos e consequentemente as menores violações de SLO. E o oposto também é visto, os maiores limiares de adição/remoção levam aos menores custos e maiores violações de SLO. É evidente que ao controlar esses limiares estamos controlando o quanto de violação aceitamos (ou o quanto de custo estamos dispostos a pagar).

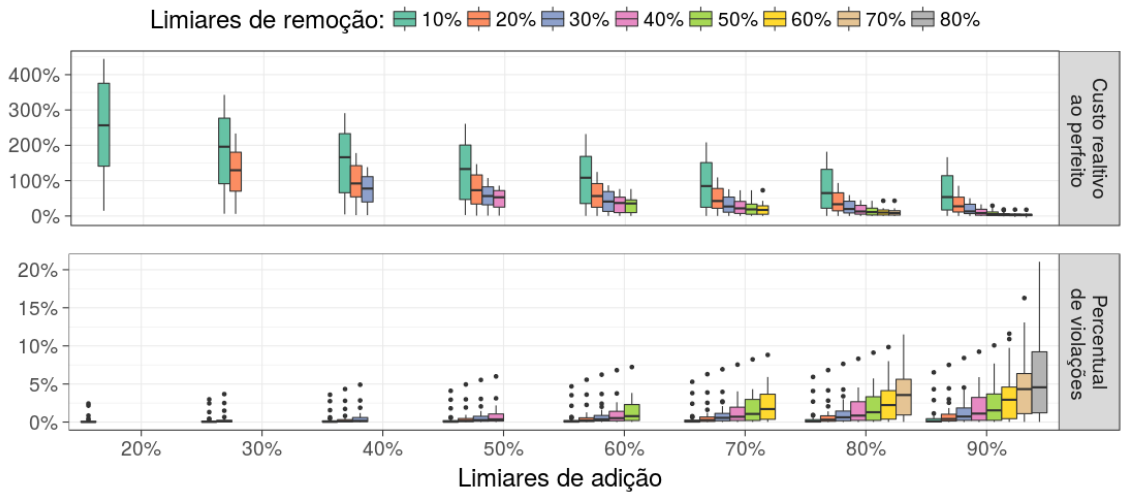

Figura 5. Desempenho da abordagem de provisionamento reativo em termos do custo de provisionamento e do percentual de violações de SLO.

Controlar esses limiares, no entanto, não é trivial, pois o que comumente buscamos é a manutenção da QoS desejada com o menor custo possível de provisionamento. Existem valores de limiares de adição/remoção que levam várias aplicações a poucas violações? Agrupamos os resultados conforme o percentual de violação de SLOs observado. Três classes foram definidas: $= 0\%$ (sem violações), $\leq 0,1\%$ e $\leq 1\%$. A partir da Figura 6 é possível identificar a quantidade de aplicações cujas violações de SLO enquadram-se nas diferentes classes considerando as diferentes configurações analisadas. Verificamos que nenhuma das configurações de provisionamento simuladas foi suficiente para atingir os objetivos de todas as aplicações, mesmo para o cenário mais flexível com um limite máximo de 1% de violações de SLO. Nesse cenário o percentual médio de aplicações cujo limite de violações foi respeitado é de cerca de 67%.

Restringindo o limite de violações para 0,1% ocorre um redução da capacidade da solução reativa em atingir o objetivo de QoS da aplicação, com uma média de sucesso para aproximadamente 30% das aplicações e de 60% para as configurações com melhor desempenho (barras mais a esquerda). Para o cenário mais conservador, em que violações de SLO não são aceitas, a meta só é atingida por 26% das aplicações, no melhor caso.

[5]Limiares de adição são sempre configurados com valores maiores que os dos limiares de remoção.
[6]Percentual de intervalos de controle em que a utilização de CPU atingiu 100%.

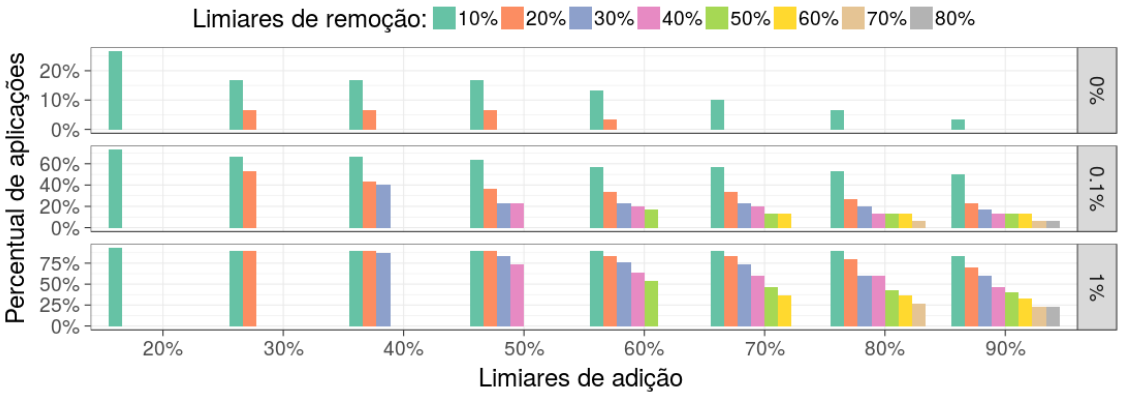

Figura 6. Análise do percentual de aplicações em que foi possível atingir os objetivos de QoS em termos dos limites de violações de SLO.

É importante lembrar que o cumprimento dos objetivos de QoS estão associados a custos de provisionamento, e quanto mais restritivo o limite aceitável de violações de SLO maior será o custo de provisionamento. Para as configurações mais conservadoras (menores limiares de adição/remoção) o custo pode chegar a ser até 400% maior que o custo alcançado pelo provisionamento perfeito. Na prática, um custo proibitivo. A avaliação do custo incorrido para cada um dos objetivos de SLO pode ser melhor observada na Figura 7. O diagrama de caixa mostra, para cada aplicação cujos objetivos de QoS foram satisfeitos, o menor custo de provisionamento relativo ao provisionamento perfeito, que consiste na seleção não realista do melhor cenário de configuração.

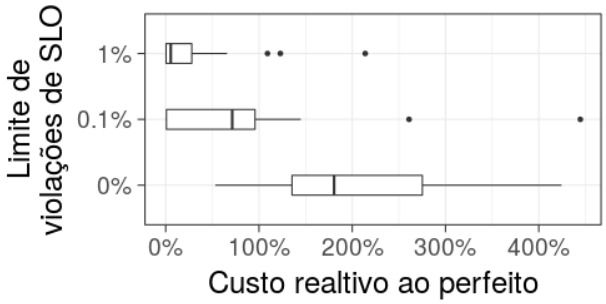

Figura 7. Análise de custo relativos ao provisionamento perfeito para diferentes cenários de limites de violações de SLO.

Para a classe com 0% de violações de SLO, metade das aplicações que atingiram esse objetivo apresentaram custo 180% maior que o custo do provisionamento perfeito correspondente. Para um limite máximo de $0,1\%$ de violações de SLO o custo médio é 71% maior que o obtido no provisionamento perfeito. Apenas no cenário mais flexível, em que o limite de violações de SLO é de 1%, o custo de provisionamento da abordagem reativa aproxima-se dos custos obtidos pela abordagem perfeita, com uma elevação de no máximo 6% de custo para metade das aplicações cujo objetivo foi atingido.

5. Discussão

A abordagem reativa não é adequada para a construção de um serviço de provisionamento no ambiente de IaaS. Várias são as razões que nos levam a essa conclusão. Primeiramente, a configuração das regras de reação é sensível às características das cargas de trabalho das aplicações, estando diretamente relacionadas com os perfis de consumo de CPU, o que contesta a generalidade e independência da solução em relação à aplicação provisionada. Um outro ponto que merece destaque é que a nossa análise não revelou uma configuração de limiar predominante para uma aplicação, muito menos para várias aplicações, o que inviabiliza o uso de uma configuração padrão. É senso comum acreditar que adicionar nós quando se chega em utilização de $\approx 70\%$ e remover nós quando se chega em $\approx 40\%$ é adequado, quando a nossa análise demonstra que não. Além disso, a configuração da quantidade de VMs provisionadas por ação disparada é em um outro fator, não tratado nesse trabalho, de ineficiência da abordagem. Finalmente, a abordagem reativa não é, na prática, bem sucedida em cumprir os objetivos de QoS das aplicações e, quando cumpre, leva a custos muito elevados. Em outras palavras, mesmo que soubéssemos como configurar o sistema (o que não sabemos, por isso uma varredura de parâmetros foi realizada), a abordagem reativa não foi bem sucedida em lidar com o *trade-off* custo/QoS.

6. Conclusões e Trabalhos Futuros

Neste trabalho foi realizada uma análise sobre a eficiência e limitações de abordagens reativas de provisionamento automático em IaaS. A análise foi realizada a partir da complexidade de configuração de regras de provisionamento perfeito e do desempenho de uma abordagem reativa prática em relação a violações de SLO e custo de provisionamento.

Foi demonstrado que a eficiência da técnica está relacionada com os padrões de carga de trabalho das aplicações, o que inviabiliza o seu uso como um serviço genérico de provisionamento. Além disso, mesmo para um conjunto com as configurações de provisionamento mais eficientes por aplicação, o desempenho da técnica mostra-se não eficaz em assegurar os objetivos mais restritivos de QoS da aplicação, mesmo com elevados custos de provisionamento em comparação a um cenário de provisionamento perfeito.

Como trabalhos futuros, pretende-se avaliar a construção de serviços de provisionamento automático a partir de abordagens de provisionamento mais sofisticadas, que busquem adaptar-se às variabilidades presentes nas cargas de trabalho. Além de uma avaliação da abordagem de provisionamento considerando diferentes métricas de utilização de recursos como base do provisionamento automático e reativo em ambientes de IaaS.

Agradecimentos

Essa pesquisa foi parcialmente financiada pela CAPES e pelo projeto EU-BRA BigSea (MCTI/RNP). Francisco Brasileiro é pesquisador do CNPq (processo 311297/2014-5).

Referências

Amazon (2016). Amazon auto scaling. `https://goo.gl/s1Zzx9`. Nov. 2016.

Azure, M. (2016). Autoscale a cloud service. `https://goo.gl/2CNo66`. Nov. 2016.

Bonvin, N., Papaioannou, T., and Aberer, K. (2011). Autonomic sla-driven provisioning for cloud applications. In *Cluster, Cloud and Grid Computing, 11th IEEE/ACM International Symposium on*, CCGRID '11, Newport Beach, CA, USA.

Calcavecchia, N., Caprarescu, B., Di Nitto, E., Dubois, D., and Petcu, D. (2012). Depas: a decentralized probabilistic algorithm for auto-scaling. *Computing*, 94:701–730.

Calheiros, R., Vecchiola, C., Karunamoorthy, D., and Buyya, R. (2012). The aneka platform and qos-driven resource provisioning for elastic applications on hybrid clouds. *Future Generation Computer Systems*.

Chambers, J. (2016). The R project for statistical computing. `https://goo.gl/NrCZaJ`. Nov. 2016.

da Rosa Righi, R., Rodrigues, V. F., Rostirolla, G., da Costa, C. A., Roloff, E., and Navaux, P. O. A. (2017). A lightweight plug-and-play elasticity service for self-organizing resource provisioning on parallel applications. *Future Generation Computer Systems*.

Fito, J., Goiri, I., and Guitart, J. (2010). Sla-driven elastic cloud hosting provider. In *Parallel, Distributed and Network-Based Processing, 18th Euromicro International Conference on*, PDP '10, Pisa, Italy.

Ghanbari, H., Simmons, B., Litoiu, M., and Iszlai, G. (2011). Exploring alternative approaches to implement an elasticity policy. In *Cloud Computing (CLOUD), IEEE International Conference on*.

Google (2016). Google cloud autoscaler. `https://goo.gl/LjITDz`. Nov. 2016.

Lim, H., Babu, S., Chase, J., and Parekh, S. (2009). Automated control in cloud computing: challenges and opportunities. In *Automated control for datacenters and clouds, 1st Workshop on*, ACDC '09, Barcelona, Spain.

Lorido-Botrán, T., Miguel-Alonso, J., and Lozano, J. A. (2014). A review of auto-scaling techniques for elastic applications in cloud environments. *Journal of Grid Computing*.

Marshall, P., Keahey, K., and Freeman, T. (2010). Elastic site: Using clouds to elastically extend site resources. In *Cluster, Cloud and Grid Computing, 10th IEEE/ACM International Conference on*, CCGRID '10, Melbourne, Victoria, Australia.

Morais, F., Brasileiro, F., Lopes, R., Araújo, R., Satterfield, W., and Rosa, L. (2013). Autoflex: Service agnostic auto-scaling framework for iaas deployment models. In *Cluster, Cloud and Grid Computing, 13th IEEE/ACM International Symposium on*, CCGRID '13, Delft, Netherlands.

Morais, F., Lopes, R., and Brasileiro, F. (2016). Instance type selection in proactive horizontal auto-scaling. In *Cloud Computing Technology and Science, 8th IEEE International Conference on*, CloudCom '16, Luxembourg.

Netto, M. A., Cardonha, C., Cunha, R. L., and Assunçao, M. D. (2014). Evaluating auto-scaling strategies for cloud computing environments. In *Modelling, Analysis & Simulation of Computer and Telecommunication Systems, 22nd IEEE International Symposium on*, MASCOTS '14, pages 187–196. IEEE.

Rackspace (2016). Rackspace autoscale. `https://goo.gl/dxRR8Z`. Nov. 2016.

Seung, Y., Lam, T., Li, L., and Woo, T. (2011). Cloudflex: Seamless scaling of enterprise applications into the cloud. In *Computer Communications, 30th IEEE International Conference on*, INFOCON '11, Shanghai, China.