Sobre a Amplitude da Elasticidade dos Provedores Atuais de Computação na Nuvem

Rostand Costa[1,2], Francisco Brasileiro[1], Guido Lemos[2], Dênio Mariz[2]

[1]Universidade Federal de Campina Grande
Departamento de Sistemas e Computação
Laboratório de Sistemas Distribuídos
58.429-900, Campina Grande, PB

[2]Universidade Federal da Paraíba
Departamento de Informática
Laboratório de Aplicações de Vídeo Digital
58.050-900, João Pessoa, PB

{rostand.costa,fubica}@lsd.ufcg.edu.br, {guido,denio}@lavid.ufpb.br

Abstract. *The cloud computing paradigm allows the provision of Information Technology infrastructure in the form of a service that clients acquire on-demand and paying only for the amount of service that they actually consume. Many embarrassingly parallel applications could potentially achieve an enormous benefit from the elasticity offered by cloud computing providers. The execution time of these applications is inversely proportional to the amount of computing resources available to process them. Unfortunately, current public cloud computing providers do impose strict limits on the amount of resources that a single user can simultaneously acquire. In this paper we analyze the reasons why traditional providers need to impose such a limit. We show that increases on the limit imposed have a severe impact on the profit achieved by the providers. This leads to the conclusion that new approaches to deploy cloud computing services are required to properly serve embarrassingly parallel applications.*

Resumo. *O paradigma da computação na nuvem permite o fornecimento de infraestrutura de Tecnologia da Informação sob a forma de um serviço que os clientes adquirem sob demanda e pagam apenas pela quantidade de serviços que realmente consomem. Muitas aplicações que processam grandes cargas de trabalho em paralelo poderiam potencialmente se beneficiar da elasticidade oferecida pelos provedores de computação na nuvem. Essas aplicações têm seu tempo de execução inversamente proporcional à quantidade de recursos computacionais usados para processá-las. Infelizmente, os provedores públicos atuais de computação na nuvem precisam impor um limite estrito na quantidade de recursos que um único usuário pode adquirir concomitantemente. Neste trabalho nós analisamos por que esse limite precisa ser imposto. Nossos resultados mostram que aumentos no limite imposto têm um grande impacto na lucratividade do provedor. Desse modo, é preciso pensar em novas formas de oferecer computação na nuvem para que as aplicações estudadas possam ser atendidas de forma apropriada.*

1. Introdução

Computação na nuvem é um paradigma em evolução que permite o fornecimento de Tecnologia da Informação (TI) como um serviço que pode ser adquirido *on line* e sob demanda pelos clientes. Os recursos utilizados para prover serviço aos clientes podem ser rapidamente provisionados e liberados pelos provedores do serviço, que utilizam um modelo de tarifação onde o cliente paga apenas pelo que foi efetivamente consumido. Este paradigma pode ser usado em diferentes níveis da pilha de TI [Stanoevska-Slabeva and Wozniak 2010]. No nível mais baixo da pilha, clientes podem adquirir máquinas virtuais totalmente funcionais executando um determinado sistema operacional, sobre o qual eles podem instalar e executar as suas próprias aplicações. Este tipo de serviço recebeu o nome de IaaS (do inglês, *infrastructure-as-a-service*) [Stanoevska-Slabeva and Wozniak 2010]. Este trabalho está focado nesse tipo de serviço. Dessa forma, no restante deste artigo serão usados os termos computação na nuvem e IaaS com o mesmo propósito.

Ao adquirir recursos de TI de um provedor de computação na nuvem, os clientes podem desfrutar da elasticidade oferecida, podendo aumentar e diminuir o seu consumo de serviços de uma forma virtualmente ilimitada, sem qualquer custo adicional. Em teoria, essa elasticidade ilimitada permitiria aos usuários decidir livremente, por exemplo, se desejam usar 1 recurso por 1.000 horas ou 1.000 recursos por 1 hora, pagando o mesmo preço em ambos os casos.

A elasticidade do modelo de computação na nuvem é particularmente interessante para uma classe importante de aplicações que está se tornando cada vez mais popular. As chamadas aplicações de e-ciência são caracterizadas por cargas de trabalho que requerem computação intensiva. Muitas destas aplicações podem ser paralelizadas trivialmente, através da quebra do trabalho a ser realizado em várias tarefas menores que podem ser processadas independentemente. Esta classe de aplicação é referenciada na literatura como aplicações "embaraçosamente paralelas" ou simplesmente "saco-de-tarefas" (BoT, do inglês *bag-of-tasks*) [Cirne et al. 2003]. Por exemplo, as simulações de Monte Carlo, que podem envolver a execução de milhares de cenários diferentes, podem ser paralelizadas simplesmente pela execução de cada cenário em uma unidade de processamento diferente. Aplicações que processam enormes quantidades de dados podem usualmente ser paralelizadas através da divisão dos dados entre um número de processos idênticos que executam a computação sobre cada bloco de dados independentemente; no final, pode ser necessário realizar algum tipo de consolidação dos processamentos individuais. A renderização de imagens complexas e vídeos se encaixa bem nesta descrição. A lista de aplicações BoT é vasta e engloba não apenas usuários da academia, mas também da indústria e do governo. Além disso, a quantidade crescente de dados gerada e consumida pela sociedade moderna deve aumentar a pressão para executar eficientemente estas aplicações [Hey and Trefethen 2003].

Se o cliente que necessita executar uma aplicação BoT fosse capaz de requisitar de um provedor de computação na nuvem tantas máquinas virtuais quanto as necessárias para maximizar o nível de paralelização da execução da aplicação, isto lhe permitiria executar esta aplicação no menor tempo possível, sem que isso implicasse em um gasto extra com os recursos computacionais usados. A elasticidade do serviço oferecido por um provedor de IaaS é, obviamente, limitada pela quantidade física de recursos que ele dispõe. Acontece que, atualmente, esse limite é muito mais restritivo, uma vez que os provedores de

computação na nuvem em operação restringem a quantidade de recursos que cada cliente pode demandar de cada vez a um número relativamente muito baixo, comparado com a capacidade dos provedores. Por exemplo, no momento em que este texto estava sendo escrito, um dos principais provedores comerciais em atividade limitava em 20 o número de máquinas virtuais que podem ser instanciadas de forma dedicada (*on-demand instances*) e em 100 o número de máquinas virtuais que podem ser instanciadas segundo um modelo *"best-effort"* (*spot instances*) [Amazon 2010]. Para este provedor em particular, clientes podem usar um canal paralelo de negociação para tentar aumentar este limite de forma *ad hoc*, mas como as condições sob as quais uma negociação é bem sucedida não são documentadas, nós consideramos neste artigo apenas o canal de comunicação automático. Aparentemente, o alvo do limite não é o volume da requisição em si, mas o exercício extremo da elasticidade através de grandes alocações com liberações logo em seguida. Como já existem serviços de alta demanda hospedados em provedores de computação na nuvem (ex. Gmail, Twitter, Bing etc.) e também a possibilidade de se negociar alocações superiores, é possível inferir que o limite serve como um regulador do uso intensivo de recursos por períodos curtos.

Embora os limites atualmente impostos pelos provedores de IaaS não impeçam que a maioria dos clientes enxerguem o serviço provido como uma fonte infinita de recursos, este não é o caso para a maioria das aplicações BoT. Estas aplicações requerem a instanciação de um sistema com milhares de máquinas virtuais. Além disso, quanto mais máquinas elas puderem usar, mais curto será o tempo de utilização das mesmas. O projeto Belle II Monte Carlo [Sevior et al. 2010], por exemplo, requer de 20.000 a 120.000 máquinas virtuais para o processamento em tempo aceitável dos dados produzidos em três meses de experimentos. Ou seja, eles têm uma altíssima demanda por recursos de forma bastante esporádica. Esse padrão de consumo é muito comum entre os usuários que executam aplicações BoT.

Neste artigo nós fazemos uma análise que tenta identificar as razões que levam os provedores de IaaS a imporem limites que restringem a utilidade de seus serviços para a execução de aplicações da classe BoT. Nossa metodologia baseia-se no uso de simulação. Inicialmente definimos um modelo simplificado de provedores de IaaS, apresentado na Seção 3, e um gerador de cargas de trabalho sintéticas apropriadas para esse modelo, discutido na Seção 4. Em seguida, nós apresentamos o modelo de simulação utilizado (Seção 5.1). Para instanciar o modelo de simulação de forma adequada, nós realizamos um projeto de experimento para identificar as variáveis aleatórias do modelo que tinham um maior impacto na variável de resposta, e dessa forma definir os cenários de experimentação (Seção 5.2). Os resultados das simulações executadas que apresentamos na Seção 6 apontam que aumentos no limite imposto pelo provedor de IaaS levam a impactos substanciais na lucratividade do provedor. Dessa forma, é pouco provável que os provedores de IaaS atualmente em operação possam vir a oferecer um serviço adequado para os usuários que precisam executar aplicações BoT. Na seção de conclusão deste artigo nós indicamos uma possível alternativa para a implantação de um serviço de IaaS que possa atender apropriadamente essa classe de aplicações.

Antes de apresentar o nosso estudo sobre o impacto que o limite no número máximo de recursos que um cliente pode instanciar concomitantemente tem na lucratividade de um provedor de IaaS, na próxima seção nós fazemos uma breve revisão da literatura relacionada com o nosso trabalho.

2. Trabalhos Relacionados

O trabalho de Menascé e Ngo discute como os métodos tradicionais de planejamento de capacidade foram impactados com o advento da computação na nuvem e como o atendimento dos níveis de serviço negociados passam a exigir novos mecanismos, como técnicas de computação autonômica [Menascé and Ngo 2009]. São considerados os pontos de vista tanto do cliente quanto do provedor, com a percepção de que todos os riscos e custos associados com o provisionamento de recursos migram dos ombros dos clientes para os ombros dos provedores. O aprofundamento que fazemos neste artigo nos aspectos de disponibilidade e regulação da demanda pelos provedores confirma esta condição, indicando que novas abordagens são necessárias para a construção de infraestrutura para computação na nuvem tanto para contornar as limitações atuais quanto para aliviar a carga dos provedores.

O estudo de Greenberg *et al.* [Greenberg et al. 2008] mostra que os custos típicos associados para a construção de centros de processamento de dados para nuvens possuem a seguinte distribuição: aquisição de servidores, incluindo hardware e software, respondem por 45% do custo total; montagem da infraestrutura, incluindo refrigeração e instalações lógicas e elétricas, consomem 25% dos recursos; equipamentos e canais de comunicação em geral são responsáveis por 15% do orçamento e os 15% restantes ficam por conta de fornecimento de energia e outras despesas. Um arcabouço para a análise detalhada destes investimentos e como eles compõem o custo total de propriedade [Mieritz and Kirwin 2005] dos provedores foi proposto por Li *et al.* [Li et al. 2009]. Na abordagem proposta, os quatro principais grupos de custos identificados por Greenberg *et al.* são expandidos para oito categorias e, em complemento aos investimentos iniciais, são também considerados os custos relacionados com a operação do centro de processamento de dados, considerando a elasticidade no consumo de recursos da nuvem. Para isto, o arcabouço permite a apuração do custo de amortização e do custo de utilização de cada recurso virtual. No primeiro caso, é definido quanto cada recurso custa por estar disponível para o uso e, no segundo caso, quanto custa ao ser efetivamente usado pelos clientes. No modelo usado no nosso trabalho nós utilizamos esses dois custos para computar o valor do lucro obtido por um provedor de IaaS.

No trabalho de Anandasivam *et al.* [Anandasivam et al. 2009] é introduzida uma versão do conceito de preço auto-ajustável adaptada para computação na nuvem. Considerando um contexto com demandas dinâmicas e imprevisíveis, no qual o provedor precisa decidir como alocar os seus recursos finitos de forma a maximizar a sua lucratividade, a aplicação de um sistema de leilão pode atuar como um influenciador no comportamento de usuários sensíveis ao preço dos recursos. Este mecanismo pode ter impactos positivos no controle da capacidade do provedor. Através do nosso estudo é possível constatar que o limite imposto pelos provedores também é usado como um regulador da demanda dos usuários para conciliar os picos de utilização com a sua capacidade instalada.

3. Um Modelo Simplificado de um Provedor de IaaS

Os serviços oferecidos por provedores de computação na nuvem precisam, normalmente, fornecer garantias de qualidade de serviço (QoS, do inglês *Quality of Service*) que atendam plenamente os requisitos estabelecidos com os clientes que adquirem os seus serviços, expressos através de um acordo de nível de serviço (SLA, do inglês *service level agreement*). Muitas dessas garantias são providas através da manutenção de capacidade

excedente pelo provedor. Por outro lado, os custos do provedor são reduzidos pelas vantagens que a economia de escala pode proporcionar-lhes. Por exemplo, a concentração de sua estrutura em grandes centros de processamento de dados, dedicados e centralizados, e o compartilhamento de recursos físicos através da virtualização são estratégias cruciais para efetivamente oferecer serviços de uma forma economicamente viável. Sua competitividade também é baseada na capacidade de realizar uma multiplexação estatística de picos e vales no uso simultâneo de recursos por um grande número de clientes. Outra vantagem é o nível de automação atingido pelos provedores de computação na nuvem que, entre outras coisas, permite que eles reduzam substancialmente a relação de funcionários por servidores. Adicionalmente, os provedores podem obter um aumento no nível de utilização dos seus serviços através da oferta de um portfólio de serviços que contemple diferentes modelos de precificação [Amazon 2010].

Dentre as muitas propriedade de QoS que um provedor de computação na nuvem precisa observar, neste trabalho nós iremos nos concentrar na disponibilidade do serviço (*service availability*), isto é, na probabilidade de que um cliente que solicita um serviço tenha o seu pedido plenamente atendido [1]. Esta propriedade não deve ser confundida com a confiabilidade do serviço (*service reliability*), que é representada pela probabilidade de que o serviço provido não irá falhar enquanto o cliente estiver usando-o. Por exemplo, no caso de um provedor de IaaS, a sua disponibilidade é afetada quando um cliente solicita uma nova máquina virtual e o provedor é incapaz de instanciar o recurso pedido, enquanto que a sua confiabilidade é afetada sempre que uma máquina virtual que tenha sido instanciada sofre uma falha.

Considerando que os recursos de um provedor são alocados a cada cliente por um intervalo de tempo mínimo, por exemplo, 1 hora, vamos assumir que o tempo é discretizado em intervalos de tamanho fixo (fatias), e modelaremos um provedor IaaS P em um determinado período de observação de tamanho ΔT como uma tupla:

$$P = \langle K, L, U, D, A, C_i, C_u, V, E \rangle,$$

onde:

- K é a quantidade de recursos disponíveis no servidor;
- L é a quantidade máxima de recursos que pode ser alocado em cada fatia de tempo;
- U é o conjunto de usuários registrados no servidor;
- D é a distribuição de demanda desses usuários;
- A é a estratégia de alocação de recursos usada pelo provedor;
- C_i é o custo incorrido pelo provedor para disponibilizar cada recurso individual por fatia de tempo, obtido pelo rateio da amortização do custo total de propriedade pelos recursos disponíveis e pelas fatias possíveis no mesmo período[2];
- C_u é o custo adicional incorrido pelo provedor, por fatia de tempo, gasto somente quando cada recurso individual está sendo efetivamente usado, baseado no conceito de custo de utilização proposto por Li *et al.* [Li et al. 2009] e considerando que algum nível de eficiência energética é praticado [Barroso and Hölzle 2007];
- V é o valor cobrado dos usuários pela utilização efetiva de um recurso por uma fatia de tempo ou fração;

[1] O foco em disponibilidade foi uma simplificação para tornar o modelo tratável, outras dimensões serão abordadas em trabalhos futuros.

[2] Embora os custos descritos possuam um comportamento linear e sejam uma simplificação dos custos reais, de perfil mais complexo, essa simplificação oferece uma boa aproximação e atende as necessidades do nosso modelo.

- E é o encargo para o provedor por cada violação cometida; ele pode ser tangível (ex. compensação para o usuário) ou intangível (ex. dano na imagem do provedor). Neste trabalho nós consideramos apenas o aspecto tangível dos encargos por violações.

Na próxima seção nós apresentamos em detalhes como a demanda D dos usuários U de um provedor P é descrita. Por enquanto, basta assumir que $d(u,t), 0 \leq d(u,t) \leq L, \forall u \in U, 1 \leq t \leq \Delta T$, é a quantidade de recursos demandada pelo usuário u na fatia t. Dependendo do padrão de demanda (D) dos usuários do provedor, da estratégia de alocação (A) e da capacidade (K) do provedor, cada usuário u que requisita $d(u,t)$ recursos na fatia t irá receber uma alocação de recursos associada que é expressa por $a(u,t), 0 \leq a(u,t) \leq d(u,t)$. Quando $a(u,t) < d(u,t)$ temos uma violação na disponibilidade do serviço do provedor. Dessa forma, o número total de violações ocorridas em uma fatia t é dado por:

$$v(t) = \sum_{u \in U} 1 - \lfloor \frac{a(u,t)}{d(u,t)} \rfloor.$$

Seja $\alpha(t)$ a utilização do provedor na fatia de tempo t. $\alpha(t) = \sum_{u \in U} a(u,t)$. Uma maneira de aferir a eficiência do provedor é medir o seu lucro no período de tempo considerado, o qual é representado em nosso modelo através da fórmula:

$$\Lambda = \sum_{t=1}^{\Delta T} [(V - C_u) \times \alpha(t) - v(t) \times E] - K \times C_i \times \Delta T. \quad (1)$$

4. Geração de Cargas de Trabalho Sintéticas para um Provedor de IaaS

Por causa da indisponibilidade de traços de execuções reais ou mesmo caracterizações da carga de trabalho de provedores de IaaS, nós tivemos que criar um gerador de cargas de trabalho sintéticas, para definir a demanda imposta ao provedor modelado na seção anterior.

O uso total do sistema em cada fatia de tempo t, representado por $\alpha(t)$, é resultante do perfil de uso de cada usuário individual ou, em outras palavras, da forma como ele requisita recursos dentro do limite imposto pelo provedor durante o seu tempo de permanência no sistema. Considerando o comportamento do sistema no intervalo de tempo de duração ΔT, algumas categorias de usuários irão emergir. Uma classificação inicial dos usuários está relacionada com o nível de demanda observada no período considerado. Os usuários ativos são aqueles que fizeram alguma demanda por recursos do sistema em um dado intervalo, ou seja, $d(u,t) > 0$ para algum valor de $t, 1 \leq t \leq \Delta T$. Os outros usuários são ditos inativos. Seja U_a o conjunto de usuários ativos; $U_a = \{u | u \in U \wedge \exists t, 1 \leq t \leq \Delta T, d(u,t) > 0\}$.

O comportamento de cada categoria de usuário ativo é descrito através do uso das distribuições tradicionalmente associadas na literatura com classes de usuários e sessões de uso [Feitelson 2009, Talby 2006, Jain 1991]. Para geração da carga de trabalho foi aplicada a abordagem de geração hierárquica, usando uma modelagem baseada no usuário [Feitelson 2009]. Esta técnica baseia-se na separação do comportamento dos usuários em três níveis: perfil da população/duração da sessão/atividade dentro da sessão, contemplando aspectos como localidade e amostragem [Feitelson 2009], além de auto-similaridade [Feitelson 2009]. Com isto, é possível a inclusão na carga de trabalho gerada

de longas permanências e ausências (cauda longa [Jain 1991]) e também de comportamentos regulares. O sistema modelado é do tipo fechado, com um número conhecido e finito de usuários ($|U_a|$).

A população de usuários ativos pode ser dividida em dois grupos, considerando a regularidade de demanda dos mesmos. Usuários ativos regulares são aqueles com uso ininterrupto. O conjunto de usuários regulares é descrito da seguinte forma: $U_r = \{u | u \in U_a \wedge \forall t, 1 \leq t \leq \Delta T, d(u,t) > 0\}$. Os usuários eventuais são os usuários ativos não regulares, ou seja, $U_e = U_a - U_r$.

Nós assumimos que os usuários regulares têm apenas uma sessão, cuja duração engloba pelo menos todo o intervalo de ΔT fatias considerado. Por outro lado, para os usuários eventuais o tempo de sessão é governado pelas seguintes variáveis aleatórias:

- $\tilde{o}$: duração (em fatias) de cada sessão de um usuário eventual, seguindo uma distribuição uniforme com limite inferior l_o e limite superior u_o [Jain 1991]; e
- $\tilde{i}$: intervalo entre sessões, seguindo uma distribuição pareto com parâmetros k_i e s_i [Jain 1991].

Dentro de cada sessão o usuário pode estar "em atividade" ou em "espera" (*think time*), que indicam, respectivamente, se o usuário está efetivamente usando recursos, ou não. O comportamento de cada usuário em sessão pode ser definido pela quantidade de recursos que ele utiliza, pela duração deste uso e também pelo tempo que ele fica sem usar os recursos do sistema. Desta forma, cada atividade pode ser caracterizada pela tupla $\mathcal{A} = \langle r, n, e \rangle$, onde r e n representam a quantidade de recursos requisitados por fatia de tempo e a duração da atividade em número de fatias, respectivamente, e e representa o tempo de espera até a próxima fatia quando o usuário estará em atividade. A mudança na quantidade de recursos, embora possível, implica no início de outra atividade. A seguir, serão descritos os perfis de uso de cada categoria de usuário da nossa população.

O perfil de uso dos usuários regulares foi modelado de uma forma simplificada. Usuários regulares apresentam atividades ininterruptas (sem espera) que duram uma fatia de tempo. Em cada sessão o número de recursos demandados é baseado na variável aleatória $\tilde{m}$ com distribuição normal, média τ e variância σ, onde τ é o *ticket* médio dos usuários regulares, dado por:

$$\tau = \frac{\frac{\sum_{t=1}^{\Delta t} \sum_{u \in U_r} a(u,t)}{\Delta T}}{|U_r|}.$$

O perfil de atividade dos usuários regulares é definido como:

$$\mathcal{A}_{regular} = \langle \tilde{m} \sim N(\tau, \sigma), 1, 0 \rangle.$$

Esta abordagem permite contemplar eventuais aumentos ou diminuições do tamanho das atividades dos usuários regulares que, através de compensação, não alterem substancialmente a utilização total dos usuários regulares do sistema em cada fatia de tempo. Mudanças mais abruptas no comportamento de usuários regulares que afetam este relacionamento serão tratadas adiante.

O comportamento em sessão dos usuários eventuais, por sua vez, é baseado em três variáveis aleatórias:

- $\tilde{s}$: quantidade de recursos alocados em cada atividade, seguindo uma distribuição uniforme entre 1 e L [Jain 1991];

- $\tilde{d}$: duração (em fatias) de cada atividade, seguindo uma distribuição exponencial com média λ_d [Jain 1991]; e
- $\tilde{t}$: intervalo (em fatias) entre atividades (*think time*), seguindo uma distribuição exponencial com média λ_t [Jain 1991].

O perfil de atividades dos usuários eventuais é definido como:

$$\mathcal{A}_{eventual} = \langle \tilde{s} \sim U(1, L), \tilde{d} \sim E(\lambda_d), \tilde{t} \sim E(\lambda_t) \rangle.$$

Dois perfis particulares de usuários eventuais foram também modelados para cobrir as seguintes situações: a) usuários regulares apresentando uma demanda não usual por recursos motivada por *flurries* ou *flashmobs* em seus serviços [Jung et al. 2002]; b) usuários eventuais com utilização intensiva e sensível ao tempo (aplicações BoT) [Sevior et al. 2010] que consomem todos os recursos disponíveis. Estes perfis são definidos da seguinte forma:

$$\mathcal{A}_{flashmob} = \langle U(\tau + 1, L), \tilde{d} \sim E(\lambda_d), \tilde{t} \sim E(\lambda_t) \rangle;$$
$$\mathcal{A}_{BoT} = \langle L, \tilde{d} \sim E(\lambda_d), \tilde{t} \sim E(\lambda_t) \rangle.$$

5. Descrição dos Experimentos

O principal objetivo dos experimentos de simulação é observar: i) a capacidade mínima necessária para atendimento de todas as solicitações dentro do limite; ii) a ociosidade do sistema em cada cenário; e iii) o resultado operacional com diferentes limites. Em seguida apresentaremos como o modelo de simulação foi implementado e como os cenários de simulação foram instanciados.

5.1. Implementação do Modelo de Simulação

Para ser resolvido por simulação, o modelo proposto foi implementado usando a ferramenta Möbius [Deavours et al. 2002]. Esta plataforma permite a realização de simulação de eventos discretos e resolução numérica ou analítica de modelos de sistemas que podem ser descritos em uma variedade de formalismos diferentes.

Um dos formalismos suportados é o *Replicate/Join* usado para criação do modelo composto *ActiveUsers* (Fig. 1). Este modelo contém quatro submodelos atômicos, modelados no formalismo *Stochastic Activity Network* (SAN), representando os quatro perfis de usuários descritos: Regular, Eventual, FlashMob e BoT. O uso dos nós *Replicate* permite a criação do número desejado de instâncias de cada perfil de usuário associado e também o compartilhamento de estado entre as instâncias de um mesmo tipo de submodelo. O nó *Join*, por sua vez, permite o compartilhamento de estado entre instâncias de submodelos de tipos diferentes. Desta forma, a carga de trabalho sintética foi construída através da atividade autônoma e combinada de uma instância do submodelo Regular, cuja demanda em cada fatia é multiplicada por $|U_r|$, e $|U_e|$ instâncias dos submodelos Eventual, FlashMob e BoT, de acordo com a distribuição de atividade configurada para cada tipo de perfil.

O submodelo Eventual (Fig. 2) representa o comportamento de um usuário do perfil Eventual. Conforme descrito na Seção 4, um usuário consome recursos através de uma série de estágios. Cada usuário de perfil Eventual é inicializado randomicamente em um dos estágios possíveis (*OnSession* ou *OffSession*) que é controlado pelo lugar *On*. A partir deste momento, as atividades *OffTime* e *OnTime* começam a regular a alternância do

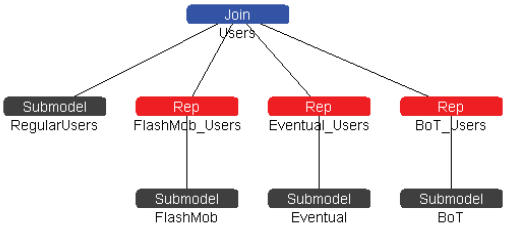

Figura 1. O Modelo Composto dos Usuários Ativos de um Provedor IaaS

usuário em sessões de uso e períodos de inatividade, controlados pelas variáveis aleatórias $\tilde{o}$ e $\tilde{i}$, respectivamente. Uma nova atividade para o usuário em sessão é atribuída (conforme descrito no perfil Eventual e usando as variáveis aleatórias $\tilde{d}$ e $\tilde{s}$) através da porta de saída *SetActivity* após um período de espera (*think time*) ser cumprido. A duração de cada espera é gerida pela atividade temporizada *NewThinkTime* (variável aleatória $\tilde{t}$). O lugar *ActivityControl*, por sua vez, controla a duração de cada atividade individual, fatia a fatia, através da atividade temporizada *NewCycle*. Os outros submodelos Regular, FlashMob e BoT possuem modelagem similar e por limitação de espaço não serão discutidos aqui. Entretanto, o modelo Möbius completo usado neste trabalho pode ser encontrado no sítio http://www.lsd.ufcg.edu.br/∼rostand/IaaSModel.zip.

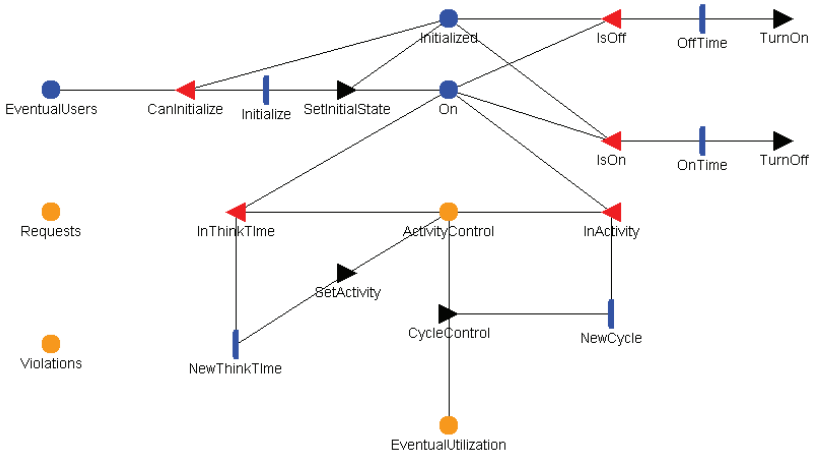

Figura 2. O modelo atômico (SAN) de um usuário do perfil Eventual

Através da configuração dos parâmetros do sistema, modelados como variáveis globais, foi usado o simulador do Möbius para executar o modelo proposto e obter as medidas de interesse desejadas ao longo do tempo, incluindo a quantidade de recursos usados, o número de solicitações e o número de violações cometidas.

5.2. Parâmetros do Sistema

Para atribuição dos parâmetros do sistema foram usadas duas estratégias: projeto de experimento (DoE, do inglês *Design of Experiment*) e varredura de parâmetros. A parte dos parâmetros relacionada com a geração da carga sintética e associada com as distribuições descritas na Seção 4 foi tratada através de um DoE do tipo 2^k fatorial [Jain 1991]. Através do DoE foi possível analisar o efeito dos parâmetros das variáveis aleatórias $\tilde{o}$ (duração da sessão), $\tilde{i}$ (intervalo entre sessões), $\tilde{s}$ (duração da atividade), $\tilde{t}$ (*think time*) e também

Fator	Baixo	Alto	Efeito Estimado	Soma dos Quadrados	% Cont.
A: Limite superior u_o (em fatias) para $\tilde{o}$	36	108	$0,0624$	$0,0312$	$6,53$
B: Limite inferior k_i (em fatias) para $\tilde{i}$	120	360	$-0,0315$	$0,0080$	$1,66$
C: Média λ_d (em fatias) para $\tilde{d}$	$0,0625$	$0,1875$	$0,0726$	$0,0421$	$8,83$
D: Média λ_t (em fatias) para $\tilde{t}$	$0,125$	$0,375$	$-0,0220$	$0,0039$	$0,81$
E: L (em quantidade de recursos)	20	100	$0,2143$	$0,3673$	$77,05$

Tabela 1. Fatores, níveis e efeitos para DoE 2^k fatorial ($k = 5$)

Parâmetro	Valor
Duração da Sessão ($\tilde{o}$)	$l_o = 1$ hora e $u_o = 72$ horas
Intervalo entre Sessões ($\tilde{i}$)	$k_i = 240$ horas e $s_i = 2$
Duração da Atividade ($\tilde{d}$)	$\lambda_d = 0.125$ (8 horas)
Espera entre Atividades ou *think time* ($\tilde{t}$)	$\lambda_t = 0.25$ (4 horas)
ΔT	8.760 horas (1 ano)
Número de Usuários Ativos ($\|U_a\|$)	{ 625; 1.250; 2.500; 5.000 }
Percentual de Atividade Eventual	{ 25%; 35%; 45%; 55%; 65%; 75%; 85%; 95% }
Percentual de Usuários com Perfil *FlashMob*	1%
Percentual de Usuários com Perfil *BoT*	{ 10%; 15%; 20%; 25% }
Limite (L)	{ 20; 30; 40; 50; 60; 70; 80; 90; 100 }
Ticket Médio (τ)	2 recursos

Tabela 2. Parâmetros Usados na Simulação

do valor de L sobre uma das variáveis de resposta do sistema: a utilização máxima do sistema em um dado intervalo $(max(\alpha(t)) \; \forall \; t, 1 \leq t \leq \Delta T)$. Os níveis atribuídos para o experimento fatorial são apresentados na Tabela 1.

Foram conduzidas várias repetições dos 32 experimentos para obter médias com 95% de intervalo de confiança com erro máximo de 5% para mais ou para menos. A contribuição de cada fator está exibida na Tabela 1, com destaque para o fator predominante L que teve participação de $77,05\%$. A única interação relevante (acima de $0,5\%$) foi BC que apresentou uma contribuição de $2,53\%$. Como resultado da análise dos efeitos através de ANOVA [Jain 1991], o F-Value de $158,6521$ implica que o modelo é significativo. O R^2 ajustado indica que o modelo explica $96,83\%$ da variação observada e o R^2 de predição está dentro de $0,20$ do R^2 ajustado, representando uma boa capacidade de predição do modelo. Maiores detalhes sobre este estudo, incluindo os gráficos de diagnóstico, cubo e interação, podem ser encontrados no mesmo endereço citado na Seção 5.1. De acordo com os resultados, a variação dos quatro primeiros fatores não afetou o comportamento da variável de resposta que ocorreu em função da variação de L.

Para a realização das simulações, os valores destes quatro parâmetros foram ajustados para os valores medianos entre os níveis "Alto" e "Baixo" usados no DoE e para os parâmetros *Percentual de Atividade Eventual*, *Percentual de Usuários com Perfil BoT*, *Número de Usuários Ativos* e L foi aplicada uma varredura de parâmetros. A Tabela 2 mostra como o sistema foi configurado para os experimentos.

6. Resultados e Análise

No primeiro experimento, o objetivo foi observar como a lucratividade do provedor era impactada com o aumento do limite imposto pelo provedor (L). Nesse experimento nós consideramos uma situação em que a disponibilidade do provedor deve ser mantida em 100%. Para tal, a estratégia de alocação (A) simulada é tal, que para qualquer fatia t, sempre é possível alocar recursos para um usuário u que tenha uma demanda positiva

$(d(u,t) > 0)$ e, portanto, $a(u,t) = d(u,t) \forall u \in U \wedge 1 \leq t \leq \Delta t$. Dessa forma, considerando a Equação 1, como as penalidades serão nulas e a receita líquida da execução de uma mesma carga de trabalho é constante, o lucro do provedor é afetado apenas pela capacidade que precisa ser mantida para atender o nível de disponibilidade desejado. Para garantir condições similares de carga do sistema, o número de usuários ativos foi mantido constante para este experimento em 5.000 usuários. Entretanto, foi feita uma varredura dos parâmetros *Percentual de Atividade Eventual* e *Percentual de Usuários com Perfil BoT* para simular diferentes cenários de atividade regular e eventual e diferentes participações dos usuários com perfil *BoT*. Esta classe de usuários é especialmente interessante para esta análise porque possuem cargas de trabalho de alto volume e sensíveis ao tempo.

Para cobrir todas as combinações dos parâmetros de entrada foram realizadas 288 simulações - repetidas até que as médias obtidas tivessem 95% de intervalo de confiança e erro máximo de 5% para mais ou para menos. A resposta de interesse observada foi a utilização máxima $(max(\alpha(t)))$ observada em todas as fatias de tempo de cada configuração do sistema simulado, já que esta define a capacidade mínima necessária para garantir 100% de disponibilidade. Os resultados obtidos estão exibidos graficamente na Fig. 3.

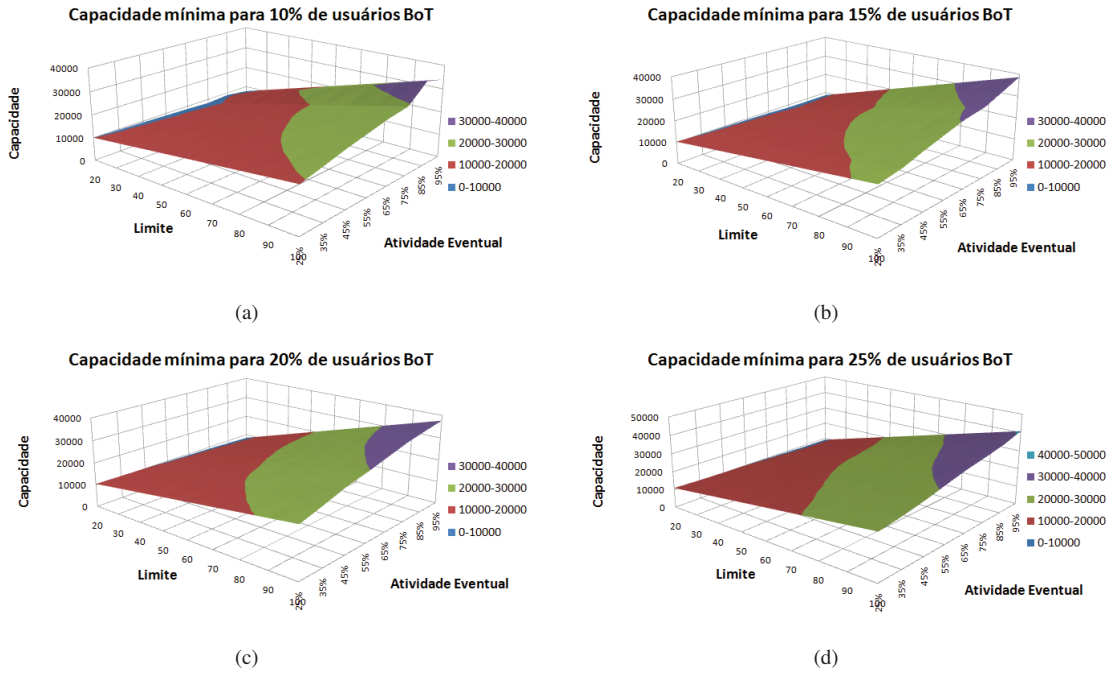

Figura 3. Capacidade mínima para atender 100% dos pedidos dos usuários em diferentes cenários de limite (L), Percentual de Atividade Eventual e Percentual de Usuários com Perfil BoT

Como pode ser observado, mesmo assumindo uma população de tamanho constante, a capacidade mínima necessária aumenta à medida que o limite é incrementado. Esta demanda por maior capacidade instalada associada com o aumento da alocação máxima de recursos por usuário já está presente mesmo em cenários onde a atividade regular é dominante (25% de usuários eventuais e 10% com perfil BoT). Onde a atividade eventual é mais preponderante com 95%, dos quais 25% dos usuários possuem perfil BoT, o aumento necessário da capacidade instalada chega a representar quase o dobro.

O segundo experimento teve como finalidade observar como o incremento na capaci-

dade instalada afeta o nível de utilização do sistema. Usando os valores de $max(\alpha(t))$ obtidos no experimento anterior como a capacidade instalada do provedor (K), os mesmos cenários foram novamente executados (288 simulações, médias com 95% IC e erro máximo de 5%) para obter a ociosidade apresentada pelo sistema. A ociosidade é representada pela razão entre a utilização total observada em ΔT e a capacidade total disponível no mesmo período: $(\frac{\sum_{t=1}^{\Delta T} \alpha(t)}{K \times \Delta T})$. Os resultados obtidos indicaram uma variação da ociosidade proporcional à variação do limite, com amplitude variando consistentemente de 20% a 65% de capacidade ociosa em todas as combinações de atividade eventual e perfil BoT simuladas. A Fig. 4 ilustra a ociosidade encontrada em quatro cenários de atividade eventual, todos com 10% de usuários com perfil BoT.

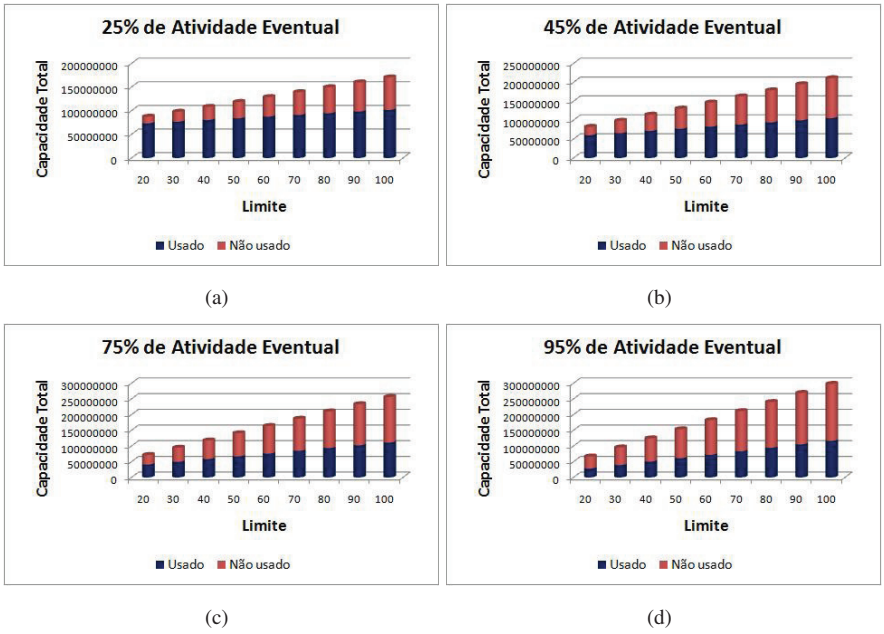

Figura 4. Ociosidade observada com a variação de limite para diferentes cenários de Atividade Eventual e com 10% de usuários com perfil BoT

Este aumento proporcional da ociosidade com o aumento do limite ganha um impacto significativo nos custos do provedor. Considerando o preço cobrado pelo principal provedor de IaaS e usando a fórmula para cálculo do lucro (Equação 1), foi realizada uma terceira análise. Foram aplicadas diferentes margens de lucro aos valores obtidos nos experimentos anteriores para identificar a partir de que ponto a operação do provedor se torna equilibrada, ou seja, sem lucro nem prejuízo, em cada configuração. Foi observado que à medida que o limite é incrementado o ponto de equilíbrio da operação só é alcançado quando a margem de lucro também é aumentada, com reflexos diretos na competitividade do provedor. Na Fig. 5, que traz o estudo referente a variação de 25 a 75% de atividade eventual e com 10% de usuários com perfil BoT, pode ser visto que a margem de lucro necessária para empatar receitas e despesas chega a 300% no maior valor considerado para o limite.

Os mesmos experimentos foram repetidos para quantidades diferentes de usuários ativos para observar se havia variação do comportamento em escalas diferentes. As curvas observadas são bastante similares para todos os valores entre 625 e 5.000 [3] usuários

[3] 5000 foi o máximo de instâncias do modelo que a ferramenta suportou simular.

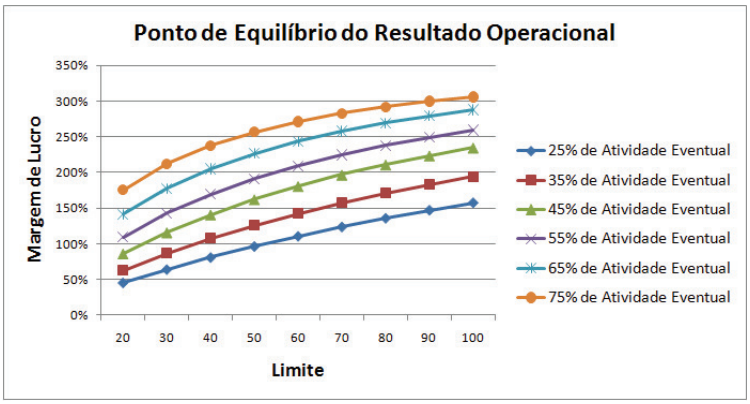

Figura 5. Equilíbrio do resultado operacional com diferentes valores de limite

quando são mantidas as mesmas condições de limite e perfis de atividade.

7. Conclusão

A partir da modelagem de um provedor de IaaS e da geração de uma carga de trabalho sintética, nós simulamos diversos cenários de utilização. A análise dos resultados aponta que não só a atribuição de um limite para alocação de recursos é necessário como o valor atribuído tem impacto relevante nos investimentos necessários em infraestrutura para garantir um nível adequado de disponibilidade do provedor. Também foi observado que usuários com utilização eventual e intensa pressionam a capacidade mínima necessária e aumentam a ociosidade do sistema, aumentando os custos operacionais do provedor. Mantido o mesmo perfil da população e o mesmo valor de limite, a dinâmica do sistema independe da quantidade de usuários não sendo, portanto, um contexto onde a economia de escala possa significar uma melhoria direta.

O uso de modelo mais próximo da realidade, mesmo com o impacto na sua tratabilidade, nos pareceu a opção mais adequada para este estudo. Para mitigar a complexidade do modelo e a inexistência de dados de campo, usamos técnicas como o design de experimento, para identificar as variáveis independentes mais importantes, e a varredura de parâmetros para instanciação de um amplo espectro de cenários. Obtivemos resultados consistentes em todos os cenários simulados.

Os resultados ajudam a entender a necessidade do uso de um limite e como o seu impacto na lucratividade do provedor está diretamente relacionado com o padrão de utilização da população de usuários, nos fazendo concluir que algumas categorias de usuários/aplicações que se beneficiariam de uma elasticidade mais ampla, continuarão sendo mal servidas se o modelo atual de provisionamento de recursos for mantido.

Os próximos passos da nossa pesquisa incluem a investigação de alternativas para minimizar os custos envolvidos com a disponibilidade de capacidade pelos provedores. Estes custos são um dos principais obstáculos para a oferta de elasticidade em condições mais flexíveis, mesmo que ainda limitada, mas que permitam que classes de aplicações de uso intenso possam se beneficiar das vantagens do modelo de computação na nuvem. A descoberta, federação e revenda de recursos já amortizados em outros contextos podem representar um caminho promissor, pois se baseiam na existência de capacidade ociosa em contextos onde os custos de disponibilidade já foram absorvidos por outros negócios ou finalidades. Dentre estes contextos com capacidade de processamento ociosa e amortizada, podemos citar data centers privados, grades P2P e recursos computacionais não

convencionais, como dispositivos móveis, telefones celulares, receptores de TV Digital etc.

O maior desafio é dotar os provedores comerciais de computação na nuvem com mecanismos, tecnológicos e comerciais, que permitam a montagem dinâmica de infraestruturas computacionais sobre estes recursos de forma a atender aplicações com diferentes requisitos de QoS e com diferentes expectativas de custos.

Agradecimentos

Os autores gostariam de agradecer a Jacques Sauvé pelas várias sugestões sobre o modelo de simulação e o gerador de cargas de trabalho sintéticas e a equipe do Möbius pelo prestativo suporte no uso da ferramenta. Francisco Brasileiro gostaria de agradecer o apoio financeiro do CNPq.

Referências

Amazon (2010). Amazon Web Services.

Anandasivam, A., Buschek, S., and Buyya, R. (2009). A Heuristic Approach for Capacity Control in Clouds. In *2009 IEEE Conference on Commerce and Enterprise Computing*.

Barroso, L. A. and Hölzle, U. (2007). The Case for Energy-Proportional Computing. *Computer*, 40(12):33–37.

Cirne, W., Paranhos, D., Costa, L., Santos-Neto, E., and Brasileiro, F. e. a. (2003). *Running Bag-of-Tasks applications on computational grids: the MyGrid approach*. IEEE.

Deavours, D., Clark, G., Courtney, T., and Daly, D. e. a. (2002). The Mobius framework and its implementation. *IEEE Transactions on Software Engineering*, 28(10):956–969.

Feitelson, D. G. (2009). *Workload Modeling for Computer Systems Performance Evaluation*. 0.30 edition.

Greenberg, A., Hamilton, J., Maltz, D. a., and Patel, P. (2008). The cost of a cloud. *ACM SIGCOMM Computer Communication Review*, 39(1):68.

Hey, A. J. G. and Trefethen, A. E. (2003). *The Data Deluge: An e-Science Perspective*, chapter 36, pages 809–824. Number January 2003. Wiley and Sons.

Jain, R. (1991). *The Art of Computer Systems Performance Analysis*. Number Ch. 32 – Queueing Networks. John Wiley and Sons.

Jung, J., Krishnamurthy, B., and Rabinovich, M. (2002). *Flash crowds and denial of service attacks*. ACM Press, New York, New York, USA.

Li, X., Li, Y., Liu, T., Qiu, J., and Wang, F. (2009). The Method and Tool of Cost Analysis for Cloud Computing. *2009 IEEE International Conference on Cloud Computing*.

Menascé, D. A. and Ngo, P. (2009). Understanding Cloud Computing: Experimentation and Capacity Planning. In *2009 Computer Measurement Group Conference*.

Mieritz, L. and Kirwin, B. (2005). Defining Gartner Total Cost of Ownership.

Sevior, M., Fifield, T., and Katayama, N. (2010). Belle monte-carlo production on the Amazon EC2 cloud. *Journal of Physics: Conference Series*, 219(1):012003.

Stanoevska-Slabeva, K. and Wozniak, T. (2010). Cloud Basics: An Introduction to Cloud Computing. *Grid and Cloud Computing*, pages 47–61.

Talby, D. (2006). User Modeling of Parallel Workloads by User Modeling of Parallel Workloads. (December).