Planejamento de Capacidade a Longo Prazo Dirigido por Métricas de Negócio para Aplicações SaaS

David Candeia[1], Raquel Lopes[2], Ricardo Araújo Santos[2]

[1] Instituto Federal de Educação, Ciência e Tecnologia da Paraíba
Campina Grande - PB – Brasil

[2]Universidade Federal de Campina Grande
Laboratório de Sistemas Distribuídos (LSD) – Campina Grande - PB – Brasil

david.maia@ifpb.edu.br, raquel@dsc.ufcg.edu.br, ricardo@lsd.ufcg.edu.br

Resumo. *O planejamento de capacidade tem feito parte da cultura do departamento de TI das empresas ao longo dos anos. No contexto da Computação na Nuvem, provedores de SaaS que compõem sua infraestrutura de TI com recursos adquiridos em provedores de IaaS por um lado economizam ao adquirir instâncias no mercado de reserva, mas por outro precisam prever a longo prazo a quantidade de instâncias necessárias. Este trabalho investiga a importância do planejamento de capacidade neste contexto e como heurísticas simples dirigidas por métricas de negócio impactam no lucro dos provedores de SaaS. Experimentos de simulação foram realizados usando cargas de trabalho sintéticas de comércio eletrônico combinadas a um modelo de utilidade baseado em provedores de SaaS atuais. Os resultados desta avaliação mostram que as heurísticas propostas aumentam, em média, o lucro do provedor de SaaS em* $3,5\%$*, havendo espaços para melhorias. Com isso, o estudo aponta indícios de que o planejamento de capacidade é uma atividade importante neste contexto, contribuindo para um aumento do lucro de provedores de SaaS.*

Abstract. *Capacity Planning is an important activity which has been part of IT department lifecycle through years. In Cloud Computing context, SaaS providers whose IT infrastructure is composed by resources from a IaaS provider can save money by acquiring instances in the reservation market with the additional problem of anticipating how many instances will be needed in a long term. This work investigates the importance of capacity planning in such context and how simple business-driven heuristics can affect SaaS providers profit.Simulation experiments were carried out using sinthetic e-commerce workloads combined to a utility model based on actual SaaS providers.The results show that proposed heuristics can increase SaaS provider profit in* 3.5% *on average, with room for improovements. Therefore, this work provides evidence that capacity planning is an important activity in the cloud computing context and can affect directly SaaS providers profit.*

1. Introdução

Planejamento de capacidade é uma atividade importante na gerência de recursos de provedores de aplicações. Tradicionalmente, uma estimativa da demanda futura da aplicação é utilizada para definir a quantidade de recursos necessários para prover a aplicação em

um certo nível de qualidade de serviço (QoS, do inglês *Quality of Service*) ou para atender certos aspectos de negócio. Considerando a hospedagem de aplicações Web tradicionais, o planejamento de capacidade tipicamente envolve superprovisionamento estático dado que a compra, instalação e configuração de recursos físicos são lentas comparadas às variações na demanda de tais aplicações, que costumam ocorrer na ordem de minutos.

O crescimento do mercado de Computação na Nuvem modificou a maneira de executar o planejamento de capacidade. Serviços providos na nuvem são normalmente divididos em três categorias de acordo com o que está sendo oferecido: (i) **infraestrutura como serviço (IaaS, do inglês *Infrastructure as a Service*)**; (ii) **plataforma como serviço (PaaS, do inglês *Platform as a Service*)**; (iii) **software como serviço (SaaS, do inglês *Software as a Service*)**. Provedores de SaaS podem executar suas aplicações em plataformas oferecidas por provedores de PaaS [Buyya et al. 2011], no entanto, consideramos neste trabalho o caso em que o provedor de SaaS monta sua infraestrutura de TI usando instâncias compradas a provedores de IaaS [Namjoshi and Gupte 2009], obtendo, assim, mais flexibilidade e controle sobre a gerência de sua infraestrutura.

Provedores de IaaS atualmente oferecem instâncias em diferentes mercados, com diferentes modelos de cobrança e QoS. Consideramos dois deles aqui: (i) o mercado **sob demanda**, no qual instâncias disponíveis no provedor podem ser adquiridas a qualquer momento mediante o pagamento de uma taxa de uso a cada intervalo de tempo menor ou igual ao intervalo mínimo de cobrança (tipicamente uma hora); e (ii) o mercado de **reserva**, no qual clientes reservam instâncias para longos intervalos futuros (geralmente maiores que um ano) mediante o pagamento de uma taxa única de reserva e de uma taxa de uso com desconto (comparada à taxa de uso do mercado sob demanda) quando as instâncias são usadas. O provedor de IaaS assegura que as instâncias reservadas estarão disponíveis sempre que o usuário desejar usá-las dentro do intervalo de reserva. Já no mercado sob demanda não há a garantia de que o usuário sempre vai ter o seu pedido de instâncias completamente atendido.

Sistemas de planejamento de capacidade podem, então, explorar os benefícios de tais mercados adquirindo instâncias mais baratas com maior disponibilidade no mercado de reserva, bem como obtendo instâncias no mercado sob demanda para lidar com variações bruscas na demanda da aplicação que não possam ser supridas pelas instâncias reservadas. Ou seja, o mercado sob demanda pode ser considerado para aquisição de instâncias adicionais de modo a prover a aplicação atendendo aos requisitos de QoS estabelecidos. Este artigo concentra-se na gerência de capacidade a longo prazo avaliando heurísticas que definem *quantas instâncias de máquinas virtuais devem ser adquiridas no mercado de reserva considerando estimativas da demanda futura*.

Como contribuições deste trabalho, primeiramente definimos um modelo de negócio para provedores de SaaS que captura a receita obtida dos clientes e os custos de adquirir instâncias nos provedores de IaaS. Este modelo considera a estratégia de *pay-as-you-go*, sendo particularmente diferente dos modelos para as aplicações Web tradicionais avaliadas no passado que tipicamente consideram receitas oriundas de um pagamento único e de atualização de versões. Propomos, também, duas heurísticas de planejamento de capacidade que determinam quantas instâncias adquirir no mercado de reserva buscando maximizar o lucro do provedor de SaaS. Por último, avaliamos as heurísticas propostas através de simulação e as comparamos com outras estratégias. Os resultados reafir-

mam a necessidade do planejamento de capacidade no cenário de Computação na Nuvem, além de demonstrar que as técnicas simples que foram propostas podem aumentar o lucro do provedor em torno de $3,5\%$, mesmo com uma predição imperfeita da demanda.

2. Trabalhos Relacionados

O termo planejamento de capacidade é comumente usado em dois contextos diferentes: a curto e a longo prazo. O planejamento a curto prazo acontece em tempo de execução, com um Sistema de Provisão Dinâmica de Recursos (DPS, do inglês *Dynamic Provisioning System*) coordenando o uso/alocação/liberação de recursos de acordo com a previsão da carga de trabalho da aplicação nas próximas horas [Urgaonkar et al. 2008, Lee et al. 2010, Sharma et al. 2011]. O planejamento a longo prazo lida com previsões de longos intervalos futuros (próximo ano, por exemplo) antecipando a quantidade de recursos que deverá ser disponibilizada para a aplicação. Considerando centros de dados tradicionais, o planejamento a longo prazo resulta na aquisição de recursos físicos [Marques et al. 2006, Sauve et al. 2006]. Entretanto, neste trabalho, assume-se que o planejamento a longo prazo resulta na reserva de instâncias em um provedor público de IaaS. Apresentamos a revisão de trabalhos de planejamento a longo e a curto prazo, uma vez que ambos os contextos podem ser considerados na elaboração de estratégias de planejamento de capacidade.

Planejamento de Capacidade baseado em métricas técnicas. Heurísticas neste grupo têm como objetivo manter certos níveis de QoS relacionados à disponibilidade da infraestrutura [Janakiraman et al. 2003], ao tempo de resposta da aplicação [Cherkasova et al. 2004] ou, ainda, a uma combinação de tais métricas [Menascé et al. 2001]. A melhor configuração da infraestrutura, que respeite os limites mínimos das métricas, será implantada.

Planejamento de Capacidade baseado em métricas de negócio. Usar apenas métricas técnicas para planejar uma infraestrutura pode gerar configurações economicamente inviáveis dado que custos e receitas não são considerados na tomada de decisão. A Gerência de TI Orientada a Negócios (BDIM, do inglês *Business-driven IT Management*) [Moura et al. 2008] tem por objetivo combinar métricas técnicas e de negócio na tomada de decisão realizada na gerência. Modelos de avaliação foram desenvolvidos considerando o custo de infraestruturas [Stage et al. 2009, Wu et al. 2011, Sharma et al. 2011], as perdas ocasionadas por negar serviço aos usuários [Marques et al. 2006, Sauve et al. 2006] e o lucro do negócio [Lopes et al. 2010, Maciel et al. 2011, Ardagna et al. 2011].

Nosso trabalho se assemelha a outros trabalhos de planejamento de capacidade dirigido a negócios uma vez que consideramos aspectos de negócio no planejamento. Entretanto, três pontos principais destacam nosso trabalho dos demais encontrados em nossa pesquisa: (i) o modelo de negócio considerado neste trabalho é baseado em provedores de SaaS; (ii) não é de nosso conhecimento a existência de trabalhos que exploram o planejamento de capacidade de provedor de SaaS usando instâncias adquiridas em diferentes mercados de um provedor de IaaS; e (iii) as heurísticas propostas adaptaram conceitos de utilização de recursos e Teoria das Filas ao cenário investigado.

3. Modelo de Utilidade

Diversos métodos podem ser utilizados para combinar métricas técnicas e métricas de negócio. Neste trabalho, usamos o conceito da microeconomia denominado de função de utilidade [Wilkes 2008], que busca capturar as preferências que guiam o comportamento de agentes. Mapeamos, então, o lucro do provedor de SaaS em uma função de utilidade e a usamos para guiar as heurísticas de planejamento. O modelo de utilidade proposto define o lucro do provedor de SaaS a partir de dois componentes: (i) da receita obtida com a oferta da aplicação para os clientes; e (ii) do custo da aquisição de instâncias no provedor de IaaS . Este modelo de utilidade considera modelos de receitas e de custos em uso atualmente por provedores de SaaS[1] e IaaS[2].

Receita: Um provedor de SaaS oferece uma aplicação A para um conjunto de clientes $U = \{u_1, u_2, \ldots, u_{|U|}\}$, cada um com requisitos de demanda particulares. Para capturar estas diferenças, o provedor oferece um portfólio de planos de assinatura $P = \{p_1, p_2, \ldots, p_{|P|}\}$ para um intervalo de tempo $D = [n^b, n^e]$ composto por $n^e - n^b$ períodos, onde $n^e \geq n^b$. Em cada período, um cliente $u \in U$ assinante de um plano $p \in P$ realiza um pagamento ao provedor de SaaS. Dessa forma, a receita total do provedor de SaaS no intervalo D é dada por: $\iota(D) = \sum_{n=n^b}^{n^e} i(n)$, onde $i(n)$ é a receita total obtida pelo provedor de SaaS no período n dentro do intervalo D. A receita do provedor de SaaS em cada período n é proveniente do conjunto de pagamentos feitos por cada cliente de SaaS $u \in U$ no período n (Equação 1). A receita $i_k(n)$, referente ao cliente u_k no período n, é composta por um valor fixo pago pelos clientes e por um valor adicional cobrado pelo uso de recursos extras quando o limite de uso no período, estabelecido no plano contratado, for ultrapassado.

$$i(n) = \sum_{k=1}^{k=|U|} i_k(n) \quad | \quad \forall 1 \leq k \leq |U|, u_k \in U \tag{1}$$

Custo: O custo total para o provedor de SaaS durante o intervalo D é calculado somando o custo total $c(n)$ em cada período n, ou seja, $\alpha(D) = \sum_{n=n^b}^{n^e} c(n)$. O custo total $c(n)$ para o provedor de SaaS em cada período n, mostrado na Equação 2, é uma combinação de três componentes: (i) do custo $ca(n)$ de uso das instâncias adquiridas no provedor de IaaS; (ii) do custo $cv(n)$ de reserva de instâncias no provedor de IaaS; e (iii) das penalidades $p(n)$ pagas aos clientes do provedor de SaaS sempre que a aplicação violar o Acordo de Nível de Serviço (SLA, do inglês *Service Level Agreement*) definido ao contratar o plano p_j .

$$c(n) = ca(n) + cv(n) + p(n) \tag{2}$$

Dados os modelos de receita e custo, a utilidade do provedor de SaaS durante o intervalo de tempo $D = [n^b, n^e]$, onde $n^e \geq n^b$, é dada pela Equação 3.

$$\upsilon(D) = \iota(D) - \alpha(D) \tag{3}$$

[1]BigCommerce, SalesForce e SurveyMonkey

[2]Amazon EC2 e Rackspace.

O modelo proposto[3] foi usado para: (i) estimar a utilidade de um plano de reserva e guiar as heurísticas de planejamento propostas; (ii) calcular a utilidade obtida por um provedor de SaaS ao implantar um plano de reserva para oferecer a aplicação.

4. Heurísticas de Planejamento de Capacidade

Detalhamos abaixo as duas heurísticas dirigidas por métricas de negócio propostas. Ambas tem como entrada uma predição da carga de trabalho futura (obtida através de dados históricos, por exemplo) contendo dados sobre um intervalo de tempo D com mesma duração do intervalo para o qual se está planejando a capacidade. Ambas produzem um plano de reserva com uma descrição do tipo e quantidade das instâncias a serem reservadas no provedor de IaaS, tal que este plano maximize a utilidade do provedor SaaS. A primeira heurística aplica conceitos de utilização de recursos enquanto a outra baseia-se em Teoria das Filas, ambas adaptadas ao contexto de Computação na Nuvem.

4.1. Heurística baseada em Utilização - UT

Esta heurística possui duas etapas: (i) simulação da execução da carga de trabalho prevista; e (ii) avaliação da simulação para definir um plano de reserva.

A primeira etapa tem como entrada a previsão da carga de trabalho composta pelo tempo de chegada e demanda de processamento, em uma instância base, de cada requisição feita pelos usuários. A heurística UT simula o processamento desta carga prevista usando instâncias adquiridas periodicamente (por exemplo, a cada hora) por um Sistema de Provisão Dinâmica de Recursos (DPS) no mercado sob demanda do provedor de IaaS. Tal simulação assume que o DPS adquire tipos e quantidades adequados de instâncias para atender ao SLA estabelecido.

Na segunda etapa (algoritmo na Figura 1), dados os componentes do custo ca e cv da Equação 2, UT calcula, para cada tipo de instância ofertado pelo provedor de IaaS, qual a menor quantidade de tempo ($limiar_t$) em que uma instância reservada deve ser utilizada para se tornar mais barata que uma instância sob demanda (linha 3). Isto é possível devido ao desconto na taxa de uso de instâncias reservadas em comparação à taxa de uso de instâncias sob demanda, compensando a taxa de reserva paga inicialmente. Por exemplo, de acordo com as taxas praticadas pela Amazon em 2011, este consumo mínimo é de $4136,364$ horas para um intervalo D de um ano. Em seguida, a heurística UT avalia, para cada tipo t, a quantidade de horas de uso de cada instância adquirida na primeira etapa (linhas 4 a 9). Esta contabilidade considera que sempre que uma instância for utilizada em D será sempre a mesma instância. Se duas instâncias forem utilizadas serão sempre as mesmas duas instâncias, sendo uma além da instância anteriormente citada.

Uma vez calculados os limiares de consumo mínimo para cada tipo de instância, a heurística seleciona a maior quantidade n de instâncias que foram usadas em paralelo por um intervalo de tempo maior que o limiar (linha 10) e calcula a quantidade de instâncias a reservar para cada tipo t como a divisão entre a quantidade total de horas (multiplicando o total n de instâncias usadas pela quantidade de tempo $uso_t[n]$ que estiveram usadas em paralelo) e o limiar de consumo mínimo para o tipo t (linha 11). É importante destacar que a heurística UT é dependente do DPS sendo usado, dado que ela considera apenas

[3]Uma versão mais detalhada do modelo de utilidade aqui apresentado pode ser obtida em `http://www.lsd.ufcg.edu.br/~davidcmm/utilityModel`.

os tipos de instâncias adquiridos na primeira etapa, ou seja, tipos não usados na primeira etapa e que poderiam trazer uma melhoria na utilidade não são verificados. A heurística também não faz equivalência entre o consumo de várias instâncias menores no consumo de uma instância com mais recursos.

Dessa forma, os planos de reserva produzidos pela heurística UT tem por objetivo encontrar a quantidade de instâncias de cada tipo t consumidas por um período de tempo maior ou igual ao $limiar_t$, sendo, assim, mais baratas que às instâncias adquiridas no mercado sob demanda do provedor de IaaS.

1: **procedure** PLAN_UT_ETAPA_2 ($consumo_{tipo_1}$, ..., $consumo_{tipo_n}$) ▷ $consumo_{tipo_n}$ é um conjunto de tuplas $\langle hora, quantidade \rangle$ indicando a quantidade de instâncias de $tipo_n$ consumidas em cada hora simulada na etapa 1
2: **for all** t in $tipo_1, tipo_2, \ldots, tipo_n$ **do**
3: Calcula $limiar_t$ baseado nas taxas do tipo t
4: **for all** i in 1, ..., maxInt **do**
5: $uso_t[i] = 0$
6: **end for**
7: **for all** $\langle hora, quantidade \rangle$ in $consumo_t$ **do**
8: $uso_t[quantidade] += 1$
9: **end for**
10: Select n tal que n é o maior índice de uso_t onde $uso_t[n] \geq limiar_t$
11: Reserva $\lceil \frac{uso_t[n] \times n}{limiar_t} \rceil$
12: **end for**
13: **end procedure**

Figura 1. Etapa 2 da Heurística UT

1: **procedure** PLAN_RF ($cargaPrevista$, ρ) ▷ $cargaPrevista$ representa um resumo da carga para um intervalo D
2: $T = \sum_{m=1}^{total\_de\_horas} \bar{S}_m * \bar{\lambda}_m$
3: Calcula $limiar_t$ baseado nas taxas do tipo t
4: **for all** t in tipoInstâncias **do**
5: $MAX_t = \lfloor T / limiar_t \rfloor$
6: **end for**
7: $planosPossíveis \leftarrow$ monta todos os planos de reserva com quantidades de instâncias variando de 0 a MAX_t
8: **for all** plano in planosPossíveis **do**
9: utilidade[plano] $\leftarrow 0$
10: **for all** hora **m** in cargaPrevista **do**
11: $resReq \leftarrow$ calcula quantidade de requisições processadas por instâncias reservadas (utilização limitada a ρ)
12: $onDemReq \leftarrow$ calcula quantidade de requisições processadas por instâncias sob demanda
13: $nãoProcessadas+ =$ calcula quantidade de requisições não processadas
14: $violadas+ =$ calcula quantidade de requisições que violaram o SLA
15: $horasReservadas+ = \lceil resReq * \bar{S}_m \rceil$
16: $horasSobDemanda+ = \lceil onDemReq * \bar{S}_m \rceil$
17: **end for**
18: $utilidade[plano] = estimaReceita() - estimaCusto(horasReservadas, horasSobDemanda) - estimaPenalidade(nãoProcessadas, violadas)$
19: **end for**
20: **return** plano que maximiza utilidade[plano]
21: **end procedure**

Figura 2. Heurística RF

4.2. Heurísticas baseada em Teoria das Filas - RF

A heurística RF elabora um plano de reserva baseado em conceitos da Teoria das Filas (algoritmo da Figura 2). A heurística recebe como entrada um valor alvo de utilização média (ρ) para as instâncias adquiridas no mercado de reserva e um sumário estatístico para cada hora da carga prevista para o intervalo D composto por: taxa média de chegada de requisições ($\bar{\lambda}$), tempo médio de serviço das requisições ($\bar{S}$), número médio de usuários e tempo médio de espera do usuário. A heurística RF estima, então, a carga total T para o período D, em horas de CPU, baseado nestes dados (linha 2) e calcula para cada tipo de instância oferecido pelo provedor de IaaS o mínimo de tempo durante o qual uma instância reservada deve ser utilizada ($limiar_t$), de maneira similar ao que foi realizado pela heurística UT descrita na Seção 4.1.

Para cada tipo t, a heurística calcula a quantidade MAX_t de instâncias do tipo t que sozinhas seriam suficientes para executar a carga total T, considerando o valor de $limiar_t$ (linhas 4 a 6). A partir dos valores de MAX_t, a heurística RF elabora o conjunto

de planos de reserva com todas as combinações possíveis de quantidades de instâncias tal que para cada tipo t se tenha uma quantidade de instâncias entre 0 e MAX_t (linha 7).

Para efetuar o cálculo da utilidade de cada possível plano de reserva, a heurística RF precisa estimar para cada hora do intervalo de tempo D, quantas instâncias serão necessárias para executar a demanda sem violar o SLA (linhas 10 a 17). Para executar a demanda RF utiliza instâncias adquiridas no mercado de reserva e instâncias adquiridas no mercado sob demanda. Cada instância reservada, pertencente ao plano de reserva sob avaliação, tem seu uso limitado ao valor alvo de utilização média ρ. Caso as instâncias reservadas, com utilização máxima ρ, não sejam suficientes para executar a demanda passa-se a utilizar instâncias sob demanda para executar o restante da demanda. No entanto, há ainda a possibilidade de que algumas requisições não possam ser atendidas devido ao risco de negação do pedido de compra de instâncias no mercado sob demanda. O não atendimento de algumas requisições pode resultar no pagamento de penalidades de acordo com o modelo de utilidade aplicado (linha 18). O plano de reserva escolhido (linha 20) será aquele que levar à maior utilidade obtida de acordo com a Equação 3.

Esta heurística é mais flexível por considerar várias combinações de diferentes tipos de instâncias, além de ter um valor alvo de utilização média ρ configurável, permitindo que a abordagem de reserva seja mais ou menos conservadora de acordo com o valor deste limiar (quanto menor, mais máquinas serão reservadas).

5. Avaliação

5.1. Modelo de Simulação

As heurísticas propostas na Seção 4 foram avaliadas com experimentos de simulação. Optamos por não usar simuladores existentes (como o CloudSim[4]) dada a dificuldade de implantar o modelo de utilidade considerado, uma vez que tratam de detalhes que não são o foco deste trabalho (e.g. modelos de consumo de energia e de alocação de máquinas virtuais). Implementamos uma extensão do *framework* SaaSim [Santos 2012][5] considerando técnicas de Verificação & Validação propostas em [Sargent 2005]. O modelo implementado[6] considera um provedor de SaaS, que oferece uma mesma aplicação para diferentes clientes, cuja infraestrutura é composta por instâncias adquiridas em um provedor de IaaS. A simulação foi dividida em duas fases: (i) planejamento de capacidade, com a execução da heurística para produzir um plano de reserva; e (ii) implantação do plano de reserva, com a execução de uma carga de trabalho sintética em uma infraestrutura baseada no plano de reserva produzido no planejamento de capacidade.

Na primeira fase, a heurística é usada para produzir o plano de reserva a partir de uma previsão da carga de trabalho para o intervalo D futuro. Consideramos impossível que o preditor, ao estimar a carga futura, produza uma estimativa perfeita para o intervalo D e para modelar a precisão da previsão usada, consideramos um erro relacionado à quantidade de clientes de SaaS submetendo requisições à aplicação. Sendo assim, um erro positivo superestima o número de clientes que compõem a carga do sistema, enquanto um erro negativo produz uma previsão subestimada dessa quantidade. Por exemplo, um erro

[4]http://www.cloudbus.org/cloudsim/

[5]Disponível em `http://github.com/ricardoas/saasim`

[6]Disponível em `http://code.google.com/p/saasim-david`

de predição de 10% indica que se a carga futura será gerada por 100 clientes de SaaS, a previsão estima uma carga com 110 clientes. Um erro de predição de -10% indica que se a carga futura será gerada por 100 clientes, a carga prevista será composta por 90 clientes.

A segunda fase tem por objetivo avaliar o plano de reserva produzido. Nesta etapa, simulamos a execução de uma carga de trabalho sintética em uma infraestrutura composta segundo o plano de reserva e avaliamos a utilidade do provedor de SaaS face aos diferentes erros de predição da carga de trabalho. É importante lembrar que a carga de trabalho de cada cliente de SaaS é composta pelo agregado de requisições dos usuários finais que acessam a aplicação adquirida pelo cliente de SaaS.

Consideramos que as requisições são distribuídas para execução na infraestrutura por um balanceador de carga que segue uma política de *round-robin* e que distribui as requisições nas instâncias proporcionalmente ao número de núcleos de CPU de cada uma. Cada instância executa as requisições segundo um modelo de processamento bem consolidado [Menasce et al. 2004]. Cada instância atende paralelamente um número máximo de m requisições controladas por um conjunto de m fichas que representam os processos/*threads* disponíveis (Figura 3). Uma requisição ao chegar à instância adquire uma ficha e entra na fila de processamento. Caso não existam fichas disponíveis, a requisição é direcionada para espera em uma fila de **backlog** de política **first-come, first-served** até que uma ficha se torne disponível. Por fim, requisições que chegam a uma instância cujo **backlog** está cheio são descartadas.

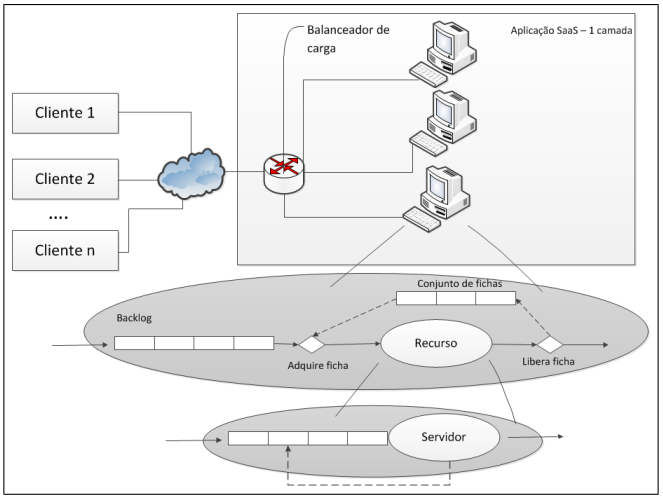

Figura 3. Modelo do Sistema: visão geral sobre filas e processamento de requisições

A fila de processamento é modelada com uma política de **time sharing**[7]. Além da demanda por processamento, cada requisição tem uma demanda de transferência de dados. Esta demanda é atendida pelo provedor de IaaS independentemente da escolha e negociação de instâncias. Cada cliente de SaaS tem, ainda, uma demanda por armazenamento relacionada à hospedagem da aplicação e aos registros de usuários. Estas duas demandas são sempre atendidas e calculamos seu custo conforme modelo apresentado na

[7]Uma requisição pode utilizar a CPU por um intervalo Δ, tipicamente muito pequeno, e em seguida a CPU é alocada para outra requisição que esteja esperando para utilizar a CPU. Desta forma, todas as requisições progridem simultaneamente e os atrasos relacionados às condições de contenção são capturados.

Seção 3.

Dada a demanda tipicamente variável das cargas de trabalho de aplicações Web [Crovella and Bestavros 1996], o controle de quantas instâncias estão sendo usadas no momento para compor a infraestrutura é feita por um DPS. Neste trabalho, usamos um DPS não realista que, a partir do conhecimento da carga futura, decide a quantidade de instâncias necessárias. Apesar desta simplificação não ser realista, ela é importante para focarmos a avaliação na qualidade do plano de reservas produzido pelas heurísticas de planejamento.

5.2. Instância do Modelo de Simulação

O planejamento fatorial completo realizado apontou o número de clientes de SaaS e o erro de predição da carga de trabalho como os fatores mais significativos. Os experimentos foram realizados de modo a explorar diversas combinações destes fatores enquanto os outros receberam níveis fixos. Sabemos que nossa análise não é exaustiva e que diferentes níveis poderiam ter sido utilizados para estes fatores fixos, porém confiamos que nossa abordagem foi suficiente para avaliar as tendências das heurísticas propostas e obter ideias de trabalhos futuros.

Para instanciar o modelo de utilidade da Seção 3 usamos dados de provedores bem conhecidos de SaaS e IaaS. Para o modelo de receita, três dos planos oferecidos em 2011 pelo provedor de SaaS **BigCommerce** foram tomados como modelo: *Bronze*, *Gold* e *Diamond*. O **BigCommerce** tarifa seus clientes mensalmente (logo, n é igual a 1 mês). Além disso, uma margem de contribuição de 30% foi escolhida para cada plano de acordo com o que é praticado no mercado[8].

Quanto ao modelo de custo, o provedor de IaaS simulado é inspirado no serviço provido pela **Amazon EC2** em 2011. Três tipos de instâncias foram considerados: *small* (1 núcleo virtual), *large* (2 núcleos virtuais) e *xlarge* (4 núcleos virtuais). A única diferença considerada entre cada tipo de instância é o total de núcleos virtuais. Após uma instância ser requisitada ao provedor de IaaS existe um período, fixado aqui como 5 minutos [Wu et al. 2011], para que a instância e a aplicação iniciem. Por último, modelamos o risco de que um pedido por instâncias no mercado sob demanda seja negado como 10%. Esse valor é intimamente ligado à imagem do provedor de IaaS no mercado e acreditamos que o valor usado seja razoável.

Nós simulamos o planejamento de capacidade e a execução da carga de trabalho para um período D de 1 ano. Foram usados 50 e 100 clientes de SaaS distribuídos uniformemente entre os planos ofertados pelo provedor de SaaS. Os erros de predição da carga de trabalho foram 40% e -40%. Para cada combinação dos valores, foram executadas 70 cargas sintéticas de trabalho diferentes de modo a calcularmos intervalos com 95% de confiança.

De acordo com Arlitt et al. [Arlitt et al. 2001], um dia de carga possui um período de pico entre 9:00 e 21:00, e as semanas possuem dias com mais e menos carga que dias típicos. Um pico de carga corresponde a 2 vezes a média de requisições, enquanto períodos mais suaves apresentam 50% da média de requisições. Estas invariantes foram combinadas com os limites de uso, os preços e as margens de contribuição dos planos

[8] `http://biz.yahoo.com/p/sum_qpmd.html`

de SaaS para calcular as taxas de chegada de requisições diárias para cada plano considerado (Tabela 1). Além disso, alguns eventos especiais (e.g. Dia das Mães) causam picos de carga em relação às semanas típicas [Arlitt et al. 2001]. As cargas utilizadas nas simulações consideram estas invariantes e foram geradas pelo gerador de cargas GEIST [Kant et al. 2001], enquanto as previsões das cargas foram derivadas destas cargas. O GEIST gera uma carga considerando uma distribuição de Poisson como a distribuição marginal do processo de chegada e, em seguida, adiciona as propriedades de multifractal e autossimilaridade presentes neste tipo de carga de trabalho.

	Plano de SaaS		
	Bronze	Gold	Diamond
Dia típico	$0,058$ req/s	$0,176$ req/s	$0,650$ req/s
Dia de pico	$0,117$ req/s	$0,350$ req/s	$1,300$ req/s
Dia suave	$0,029$ req/s	$0,090$ req/s	$0,325$ req/s

Tabela 1. Taxas de chegada (requisições/segundo) para uma semana típica

5.3. Apresentação e Análise de Resultados

As heurísticas UT e RF foram avaliadas frente a três estratégias de referência: (i) uma estratégia que usa apenas instâncias do mercado sob demanda - denominada de ON; (ii) uma estratégia que considera o pico da demanda de instâncias e reserva 20% deste pico fazendo uso de instâncias *large* - denominada de SUPER[9]; e (iii) uma estratégia ótima que, conhecendo o funcionamento futuro do DPS, testa planos de reserva contendo da menor até a maior quantidade de instâncias utilizadas pelo DPS e escolhe o plano com maior utilidade estimada - denominada de OP.

Nossa análise considera a utilidade do provedor de SaaS e o **ganho** de utilidade, em porcentagem, de cada heurística em relação à estratégia ON segundo a Equação 4.

$$ganho(\upsilon_A(D), \upsilon_{ON}(D)) = 100 * \frac{(\upsilon_A(D) - \upsilon_{ON}(D))}{|\upsilon_{ON}(D)|} \quad (4)$$

Primeiramente, verificamos a viabilidade do planejamento realizado pelas heurísticas propostas. A hipótese nula $\upsilon_{SUPER}(D) = \upsilon_{ON}(D) = \upsilon_{UT}(D) = \upsilon_{RF}(D)$ foi rejeitada em cenários com 100 clientes de SaaS pela análise de variância (ANOVA) realizada. Realizamos uma análise *post-hoc* para avaliar se alguma heurística obteve utilidades similares às utilidades de ON e concluímos que $\upsilon_{UT}(D), \upsilon_{RF}(D) > \upsilon_{ON}(D) > \upsilon_{SUPER}(D)$. Logo, as heurísticas propostas apresentam ganhos diferentes entre si, e diferentes de zero, ou seja, aumentam a utilidade do provedor de SaaS em relação à utilidade de ON. A avaliação estatística dos cenários com 50 clientes conduz às mesmas conclusões.

O segundo passo da análise *post-hoc* buscou quantificar os **ganhos** das heurísticas avaliadas analisando os cenários de 50 e 100 clientes de SaaS. A Figura 4 apresenta os **ganhos** obtidos com cada heurística/estratégia para erros de predição de -40% e 40%.

A estratégia SUPER reserva o maior número de núcleos virtuais usando instâncias do tipo *large*, enquanto as outras heurísticas usam instâncias do tipo *small*, o que a leva

[9]A estratégia SUPER reserva 20% do pico da demanda por instâncias dado que 20% é uma utilização esperada para uma infraestrutura planejada para o pico da carga de trabalho [Armbrust et al. 2009].

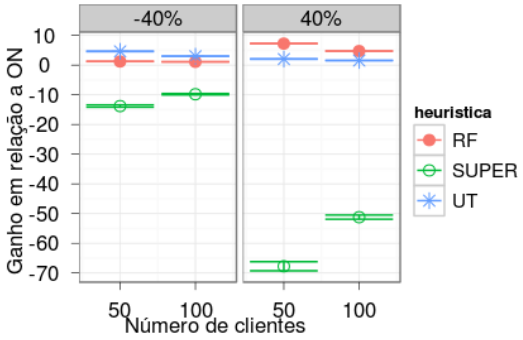

Figura 4. Ganhos de utilidade: erros de predição de -40% e 40%

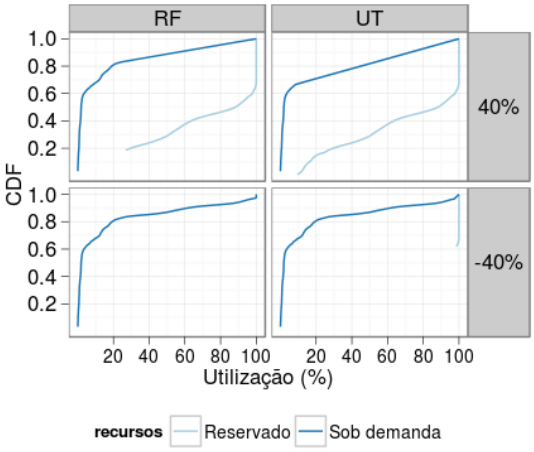

Figura 5. CDF da taxa de utilização das instâncias

a obter os maiores custos de infraestrutura. Este aumento no custo se deve ao aumento na ociosidade da infraestrutura uma vez que se precisa, na verdade, de instâncias menores. Como consequência, SUPER obteve, em todos os cenários, utilidades inferiores às utilidades obtidas por ON, logo é melhor não realizar um planejamento do que superprovisionar a infraestrutura com a estratégia SUPER. As heurísticas UT e RF apresentaram **ganhos** médios de $1,5\%$ a $7,3\%$, indicando que bons planos podem aumentar o lucro do provedor de SaaS. Considerando o conhecimento total da carga de trabalho futura, a estratégia OP apresenta **ganhos** da ordem de 11% a 16% [10]. Apesar dos valores encontrados para os **ganhos** serem baixos, o ganho monetário para o provedor de SaaS é mais significativo quão maiores forem a receita e a quantidade de clientes.

O próximo passo da análise consistiu em avaliar o comportamento de cada heurística. As heurísticas RF e UT reservam menos instâncias que SUPER e, além disso, usam apenas instâncias *small*. A Figura 5 mostra que as instâncias reservadas por UT e RF apresentam altas taxas de utilização durante o intervalo D (i.e. taxas de utilização superiores ao limiar de $47,8\%$ da Amazon), com as instâncias de RF apresentando taxas de 100% para erros de predição de -40%. Esta alta utilização conduz a uma redução de custos em relação aos custos de ON e SUPER. Em cenários com superestimativa da carga, UT e RF reservam algumas instâncias que apresentam baixa utilização e um custo superior ao custo de instâncias sob demanda. Como consequência, o custo total da infraestrutura é aumentado e os **ganhos** das heurísticas reduzidos.

Avaliando os planos de reserva de RF e UT observa-se que RF sempre reserva menos instâncias que UT. Isto acontece porque a heurística RF busca atingir 75%[11] de taxa de utilização para suas instâncias reservadas, ao passo que UT busca uma taxa de $47,8\%$ de utilização, aumentando, assim, o número de instâncias reservadas. Como con-

[10]Os intervalos de 95% de confiança para a média dos ganhos com 50 e 100 clientes de SaaS foram, respectivamente, $(15,748\%; 16,026\%)$ e $(11,527\%; 11,624\%)$. O erro de predição da carga é zero para a estratégia OP.

[11]Este valor se baseia no limiar de utilização a partir do qual o tempo de resposta passa a apresentar crescimento exponencial.

sequência destas escolhas, em cenários de subestimativa da carga, UT utiliza uma maior quantidade de instâncias mais baratas, aumentando sua utilidade. Por outro lado, em cenários de superestimativa da carga, UT utiliza uma maior quantidade de instâncias mais caras, obtendo, assim, utilidades inferiores às utilidades de RF.

Para considerar a dificuldade de predição da carga de trabalho baseada em histórico e em tendências de mercado, realizamos uma análise de sensibilidade do erro de predição da carga. Esta análise tentou refletir a possibilidade de uso de preditores que resultam em diferentes erros de predição. Os erros de predição considerados foram: -40%, -20%, -10%, 0%, 10%, 20% e 40%.

A análise demonstrou que UT e RF apresentam comportamentos diferenciados para erros de superestimativa e de subestimativa da carga. Para cenários de superestimativa da carga, a natureza mais conservadora de RF (reservas menores) fez com que a mesma obtivesse melhores **ganhos** que UT. Para cenários de subestimativa da carga, UT apresentou os melhores resultados. É importante destacar que erros de predição menores contribuem para melhores **ganhos** de ambas as heurísticas. Quando o erro de predição foi zero, os intervalos de confiança para a média dos **ganhos** obtidos por UT e RF para 100 clientes de SaaS foram de $(5,21\%; 5,26\%)$ e $(5,15\%; 5,18\%)$, respectivamente. A estratégia OP para 100 clientes obtém um intervalo de confiança para a média dos **ganhos** de $(11,527\%; 11,624\%)$ sugerindo que, apesar das heurísticas propostas terem aumentado a utilidade do provedor de SaaS, existe espaço para melhorias.

6. Conclusões e Trabalhos Futuros

O foco deste trabalho é demonstrar que o planejamento de capacidade não deve ser negligenciado no contexto de Computação na Nuvem. Para isso, um modelo de utilidade que captura aspectos de negócio relacionados à oferta de SaaS foi elaborado e utilizado para guiar a reserva de instâncias em um provedor de IaaS realizada por duas heurísticas propostas, UT e RF. As heurísticas propostas foram avaliadas em um cenário de planejamento de capacidade para um intervalo de um ano através de experimentos de simulação, usando cargas de trabalho sintéticas de comércio eletrônico e comparadas tanto com uma estratégia de superprovisionamento (SUPER) como com uma estratégia que não reserva instâncias (ON). Nossos resultados mostram que RF conduz a um aumento médio de $3,77\%$ no lucro do provedor de SaaS, enquanto UT conduz a um aumento médio de $3,19\%$. A estratégia SUPER obteve os piores lucros devido às más escolhas de tipo e quantidade de instâncias reservadas. Conclui-se que, para os cenários observados, não se deve superprovisionar a infraestrutura (usar a heurística SUPER) já que heurísticas simples de planejamento melhoram o lucro do provedor de SaaS.

A análise de sensibilidade mostrou que a qualidade da predição da carga de trabalho influencia os resultados das heurísticas. Grandes provedores de SaaS tendem a possuir grandes quantidades de dados históricos e a investir em boas técnicas de predição de carga, obtendo erros pequenos e explorando melhor as heurísticas propostas. Todavia, pequenos provedores podem não ter acesso a estas possibilidades, obtendo erros maiores na predição e, assim, não conseguindo explorar ao máximo as heurísticas propostas.

As simplificações consideradas conduzem a algumas ameaças de validade que podem ser investigadas em trabalhos futuros. Quanto à *validade externa*, usamos cargas de trabalho artificiais de comércio eletrônico criadas pelo GEIST, um gerador antigo, dado

que não encontramos um gerador baseado em estudos recentes. Seria interessante usar cargas reais coletadas de aplicações. Apesar disso, alimentamos o modelo de utilidade com informações de provedores de IaaS e SaaS reais e, por isso, nossos resultados são importantes para traçar uma visão geral da atuação das heurísticas no cenário considerado. Quanto à *validade de construção*, a aplicação foi modelada com uma única camada e sessões de usuários não foram consideradas, o que pode ser melhorado com o uso de um modelo mais apurado do tipo de aplicação investigado. Por fim, consideramos importante investigar o impacto da negação de serviço no desempenho das heurísticas de planejamento de capacidade e pretendemos implementar melhorias nas heurísticas propostas com base no que pode ser aprendido com o estudo da estratégia OP.

Referências

Ardagna, D., Panicucci, B., and Passacantando, M. (2011). A game theoretic formulation of the service provisioning problem in cloud systems. In *Proceedings of the 20th international conference on World wide web*, pages 177–186. ACM.

Arlitt, M., Krishnamurthy, D., and Rolia, J. (2001). Characterizing the scalability of a large web-based shopping system. *ACM Transactions on Internet Technology*, 1(1):44–69.

Armbrust, M., Fox, A., Griffith, R., Joseph, A. D., Katz, R., Konwinski, A., Lee, G., Patterson, D., Rabkin, A., Stoica, I., and Zaharia, M. (2009). Above the Clouds : A Berkeley View of Cloud Computing Cloud Computing : An Old Idea Whose Time Has (Finally) Come. *Computing*, pages 07–013.

Buyya, R., Broberg, J., and Goscinski, A. (2011). *Cloud computing: Principle and Paradigms*. Wiley Online Library.

Cherkasova, L., Tang, W., and Singhal, S. (2004). An sla-oriented capacity planning tool for streaming media services. In *Dependable Systems and Networks, 2004 International Conference on*, pages 743–752. IEEE.

Crovella, M. and Bestavros, A. (1996). Self-similarity in world wide web traffic: evidence and possible causes. In *ACM SIGMETRICS Performance Evaluation Review*, volume 24, pages 160–169. ACM.

Janakiraman, G., Santos, J., and Turner, Y. (2003). Automated multi-tier system design for service availability. In *Proceedings of the First Workshop on Design of Self-Managing Systems*. Citeseer.

Kant, K., Tewari, V., and Iyer, R. (2001). Geist: Generator of ecommerce and internet server traffic. In *Proc. of Int. Symposium on Performance Analysis of Systems and Software*.

Lee, Y., Wang, C., Zomaya, A., and Zhou, B. (2010). Profit-driven service request scheduling in clouds. In *Proceedings of the 2010 10th IEEE/ACM International Conference on Cluster, Cloud and Grid Computing*, pages 15–24. IEEE Computer Society.

Lopes, R., Brasileiro, F., and Maciel, P. (2010). Business-driven capacity planning of a cloud-based it infrastructure for the execution of web applications. In *Parallel & Distributed Processing, Workshops and Phd Forum (IPDPSW), 2010 IEEE International Symposium on*, pages 1–8. IEEE.

Maciel, P., Brasileiro, F., Santos, R., Maia, D., Lopes, R., Aquino de Carvalho, M., Costa Ribeiro, R., Andrade, N., and Mowbray, M. (2011). Business-driven short-term management of a hybrid it infrastructure. *Journal of Parallel and Distributed Computing*.

Marques, F., Sauvé, J., and Moura, A. (2006). Business-oriented capacity planning of it infrastructure to handle load surges. In *Network Operations and Management Symposium, 2006. NOMS 2006. 10th IEEE/IFIP*, pages 1–4. IEEE.

Menasce, D., Almeida, V., Dowdy, L., and Dowdy, L. (2004). *Performance by design: computer capacity planning by example*. Prentice Hall.

Menascé, D., Barbará, D., and Dodge, R. (2001). Preserving qos of e-commerce sites through self-tuning: A performance model approach. In *Proceedings of the 3rd ACM conference on Electronic Commerce*, pages 224–234. ACM.

Moura, A., Sauve, J., and Bartolini, C. (2008). Business-driven it management-upping the ante of it: exploring the linkage between it and business to improve both it and business results. *Communications Magazine, IEEE*, 46(10):148–153.

Namjoshi, J. and Gupte, A. (2009). Service Oriented Architecture for Cloud Based Travel Reservation Software as a Service. *2009 IEEE International Conference on Cloud Computing*, pages 147–150.

Santos, R. A. (2012). Saasim - um framework para simulação de software as a service. Master's thesis, Federal University of Campina Grande.

Sargent, R. (2005). Verification and validation of simulation models. In *Proceedings of the 37th conference on Winter simulation*, pages 130–143. Winter Simulation Conference.

Sauve, J., Marques, F., Moura, A., Sampaio, M., Jornada, J., and Radziuk, E. (2006). Optimal design of e-commerce site infrastructure from a business perspective. In *System Sciences, 2006. HICSS'06. Proceedings of the 39th Annual Hawaii International Conference on*, volume 8, pages 178c–178c. IEEE.

Sharma, U., Shenoy, P., Sahu, S., and Shaikh, A. (2011). A cost-aware elasticity provisioning system for the cloud. In *Distributed Computing Systems (ICDCS), 2011 31st International Conference on*, pages 559–570. IEEE.

Stage, A., Setzer, T., and Bichler, M. (2009). Automated capacity management and selection of infrastructure-as-a-service providers. In *Integrated Network Management-Workshops, 2009. IM'09. IFIP/IEEE International Symposium on*, pages 20–23. IEEE.

Urgaonkar, B., Shenoy, P., Chandra, A., Goyal, P., and Wood, T. (2008). Agile dynamic provisioning of multi-tier internet applications. *ACM Transactions on Autonomous and Adaptive Systems (TAAS)*, 3(1):1.

Wilkes, J. (2008). Utility functions, prices, and negotiation. *Market-Oriented Grid and Utility Computing*, pages 67–88.

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