Gerenciamento de Capacidade Auto-Adaptativo para Ambientes Virtualizados Multi-Camadas

Ítalo Cunha, Jussara Almeida, Virgílio Almeida, Marcos Santos

[1]Departamento de Ciência da Computação – Universidade Federal de Minas Gerais
Belo Horizonte, Brasil – 30123-970

{cunha, jussara, virgilio, marcos}@dcc.ufmg.br

Abstract. *This paper addresses the problem of hosting multiple applications on a provider's virtualized multi-tier infrastructure. Building from a previous model, we design a new self-adaptive capacity management framework, which seeks to maximize the provider's business objective. Our main contributions are the more accurate multi-queue performance model, as well as the solution of the associated complex optimization model. Our approach is evaluated via simulation experiments with synthetic as well as realistic workloads. The results show that our solution is significantly more cost-effective than the approach it is built upon, which uses a single-resource performance model, and a multi-tier static allocation strategy.*

Resumo. *Este artigo aborda o problema de hospedar várias aplicações numa infra-estrutura virtualizada multi-camadas de um provedor. Partindo de um modelo prévio, nós projetamos um novo arcabouço auto-adaptativo de gerência de capacidade, que procura maximizar o objetivo de negócio do provedor. Nossas principais contribuições são um modelo de desempenho multi-filas mais preciso, bem como a solução do complexo modelo de otimização associado. Nosso método é avaliado através de simulação com cargas sintéticas e realistas. Os resultados mostram que nossa solução é significativamente mais eficiente que o método do qual foi derivada, que usa um modelo de desempenho com uma camada; e uma estratégia de alocação estática multi-camadas.*

1. Introdução

Serviços de Internet e Web geralmente recorrem a terceirização da infra-estrutura computacional como uma alternativa financeiramente atrativa para hospedar seus serviços [Ross and Westerman 2004]. Nesse cenário, o provedor de serviço assina contratos de nível de serviço (*SLA*) com um provedor de infra-estrutura. O provedor de serviço impõe requisitos de qualidade, e o objetivo do provedor de infra-estrutura é encontrar a estratégia mais eficiente para gerenciar seus recursos, compartilhados pelas aplicações hospedadas.

A gerência de capacidade dessa infra-estrutura se torna particularmente desafiadora devido a várias razões. Web-services atuais utilizam complexas e heterogêneas plataformas de serviço multi-camadas compostas de servidores HTTP, servidores de aplicação e, talvez, um banco de dados. Além disso, heterogeneidade de aplicações e flutuações da carga de trabalho [Chase et al. 2001] não podem ser eficientemente acomodadas com estratégias estáticas de gerência de capacidade. Nesse cenário, virtualização de recursos [Barham et al. 2003, Whitaker et al. 2002] cria um ambiente mais custo-efetivo, através da criação de máquinas virtuais (VMs) dedicadas a uma aplicação.

Novas cobranças dos provedores de serviço tornam mais complexa a construção de estratégias custo-efetivas de gerência de capacidade. Existe um crescente interesse

no estabelecimento de contratos onde o pagamento é proporcional aos recursos realmente utilizados [Wilkes et al. 2004]. Estes contratos devem prover garantias não só sobre a taxa de processamento, mas também sobre o tempo de resposta observado para cada requisição [Liu et al. 2005]. Esta última implica em garantias sobre a calda da distribuição do tempo de serviço, ao contrário de requisitos tradicionais sobre a *média* do tempo de resposta. Estes requisitos implicam num aumento da complexidade dos contratos, modelos de negócio e, por fim, soluções de gerência de capacidade mais sofisticadas.

Em [Abrahao et al. 2006], propomos uma solução para gerenciamento de capacidade auto-adaptativa que aborda alguns dos desafios mencionados anteriormente. Nossa solução combina um modelo de custo, construído a partir de um modelo de contrato de dois níveis, um modelo de desempenho baseado em filas e um modelo de otimização que maximiza o objetivo de negócio do provedor. Este artigo expande nosso trabalho prévio e propõe um arcabouço de gerenciamento de capacidade significativamente mais eficiente para sistemas virtualizados multi-camadas. Este arcabouço assume que a infra-estrutura física em cada camada é virtualizada, atribuindo uma máquina virtual local para cada aplicação hospedada. Nesse caso, o modelo de desempenho representa cada máquina virtual como uma fila separada, capturando assim o paralelismo inerente a plataformas multi-camadas não capturado por modelos de camada única.

Nós comparamos nossas abordagens multi-camadas com nosso modelo anterior de camada única e também com uma estratégia de alocação estática multi-camadas. Simulações com cargas de trabalho sintéticas e realistas são usadas para medir a eficiência relativa das estratégias analisadas em vários cenários de interesse. Nossas principais conclusões são as seguintes. Primeiro, nossa abordagem multi-camada auto-adaptativa escala razoavelmente bem para cenários práticos. Segundo, no caso de cargas pesadas e desbalanceadas, ganhos significativos de lucro são obtidos pelo provedor, se comparados com alocação estática multi-camada. Terceiro, as imprecisões de desempenho introduzidas pelo modelo de camada única levam a decisões de alocação muito conservadoras. Em conseqüência disso, a estratégia de camada única é inferior às abordagens multi-camadas auto-adaptativas e estáticas em ordens de magnitude, mesmo quando as aplicações hospedadas são homogêneas e possuem demandas balanceadas através das camadas.

Este artigo é organizado como segue. A seção 2 discute trabalhos relacionados. Nosso ambiente virtualizado multi-camadas e modelo de custo sobre o qual nossas abordagens são construídas são descritos na seção 3. A seção 4 apresenta nosso arcabouço auto-adaptativo. Resultados de simulação são apresentados na seção 5. Conclusões e trabalhos futuros estão disponíveis na seção 6.

2. Trabalhos Relacionados

Trabalhos prévios sobre gerência autônoma de capacidade diferem deste estudo pois não se aplicam a ambientes multi-camada. Nosso trabalho prévio em [Abrahao et al. 2006] combina um modelo de negócio de dois níveis, um modelo de desempenho baseado em filas e um modelo de otimização para alocar dinamicamente a capacidade disponível entre as aplicações, maximizando o objetivo de negócio do provedor. Modelos com um único centro de serviço, como filas M/M/1 e M/G/1, são usados para prover estimativas do tempo de resposta. Este trabalho estende os modelos prévios de desempenho e otimização para arcabouços multi-camadas, aumentando sua precisão e eficiência.

Uma pequena quantidade de trabalhos usam modelos analíticos para avaliar o desempenho de sistemas para realocar recursos entre as aplicações sendo executadas. Em

[Almeida et al. 2006], nós consideramos um método que leva em consideração o custo de operação da infra-estrutura (ex.: energia) no mecanismo de gerência. O lucro total é uma combinação do modelo de negócio e dos custos de operação. Em [Villela et al. 2004], os autores apresentam métodos para alocar um número fixo de servidores entre um conjunto arbitrário de usuários com diferentes cargas e contratos de serviço. Os autores assumem uma arquitetura de uma camada representada por uma fila M/G/1. Nenhum dos trabalhos anteriores é aplicável a sistemas com múltiplas camadas.

O trabalho em [Liu et al. 2001] faz alocação de recursos visando maximizar lucros de um provedor de comércio eletrônico. Os autores assumem uma carga estática e não consideram virtualização em sua avaliação. A ref. [Menascé and Bennani 2006] considera ambientes virtuais e desenvolve um método de gerência de capacidade para alocação de CPUs, considerando alocação por prioridades e compartilhamento. Nenhum destes dois métodos possui modelos de negócio, levando em consideração apenas métricas da infra-estrutura, o que pode não estar de acordo com o objetivo do provedor (ex.: maximizar lucro). Além disso, estes métodos não se aplicam a ambientes multi-camadas.

Um modelo analítico fechado baseado em filas para serviços multi-camadas é apresentado em [Urgaonkar et al. 2005a]. Este modelo é genérico o suficiente para capturar o comportamento de camadas com diferentes características de aplicação e desempenho. Porém, o foco do modelo é prever o tempo médio de resposta não tendo modelos de otimização para alocação dinâmica de capacidade. Em [Urgaonkar et al. 2005b], os autores apresentam um arcabouço de gerência de capacidade para aplicações multi-camadas com cargas de sessão. Filas G/G/1 são utilizadas, mas a alocação de capacidade só é obtida após uso de aproximações e limitações. Além disso, nosso método provê gerência de capacidade alinhada com os objetivos de negócio do provedor, através de um flexível modelo de negócio.

3. Modelo de Infraestrutura

Esta seção descreve a plataforma alvo (seção 3.1) do nosso arcabouço de gerência de capacidade auto-adaptativo e o modelo de custo (seção 3.2) sobre o qual é construído.

3.1. Plataforma de Hospedagem Virtual

Consideramos um cenário onde um provedor hospeda múltiplos serviços Web de terceiros. Serviços Web típicos são compostos de diferentes tipos de requisições, nós nos referimos a cada tipo de requisição como uma *classe de aplicação* e assumimos que a infra-estrutura hospeda N classes independentes oriundas de todos os serviços.

Nós consideramos que a infra-estrutura do provedor é composta de múltiplas (K) camadas, caso comum para vários serviços Web. Cada camada é responsável por uma tarefa específica no processo de servir uma requisição (ex.: camadas de apresentação, aplicação e banco de dados). Camadas operam em paralelo e requisições visitam camadas em seqüência. Isto é, uma requisição da classe i entra na camada j, é servida, e então deixa o sistema com probabilidade $p_{i,j}$ ($i = 1..N, j = 1..K, p_{i,K} = 1$).

Cada camada é hospedada em *hardware* dedicado, compartilhado por todas as classes. Assumimos que cada camada roda um mecanismo de virtualização, como o Xen [Barham et al. 2003] ou Denali [Whitaker et al. 2002], que provê diferenciação de serviço e isolamento de desempenho para as classes hospedadas. A camada de virtualização possibilita ao provedor aumentar ou diminuir dinamicamente a quantidade

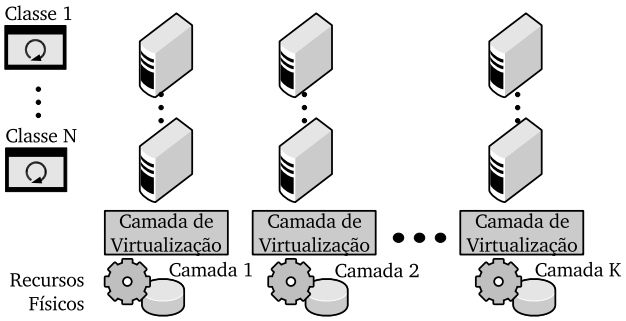


Figura 1. Plataforma de Hospedagem de Serviços Virtualizados Multi-Camada

de recursos físicos dedicados a uma classe numa camada, independentemente. Assim, nós definimos o problema de alocação de capacidade como a determinação das frações de capacidade física para cada classe i em cada camada j.

A plataforma de hospedagem considerada é mostrada na Figura 1. Sobre cada camada da infra-estrutura física, uma camada de virtualização cria N máquinas virtuais (VMs) isoladas, uma para cada classe. Cada requisição de uma classe de aplicação é servida por K VMs dedicadas àquela classe. Assumimos que as VMs usam de algum esquema de controle de admissão para evitar instabilidade e para garantir que os requisitos de tempo de resposta serão atendidos. Focamos na alocação dinâmica de capacidade, com o alvo de maximizar o objetivo de negócio do provedor, descrito a seguir.

3.2. Modelo de Custo

Em [Abrahao et al. 2006], propomos um esquema de cobrança que trata a alta variação das cargas das aplicações em serviços online. Estes normalmente recebem moderadas cargas de trabalho, mas ocasionalmente são subitamente inundados com uma grande sobrecarga de requisições. Este fenômeno, conhecido como *flash crowd*, gera congestionamento na infra-estrutura de serviço, causando atrasos significativos aos usuários. Em contrapartida, nós propomos contratos com dois níveis de requisitos, a dizer, normal e sobrecarregado. Se necessário, nossa abordagem pode ser estendida para mais de dois níveis, especificando múltiplos alvos de desempenho. Na discussão seguinte, nos referimos ao provedor de serviço como *usuário*, que estabelece um contrato com o *provedor* de infra-estrutura para hospedar seu serviço.

No modo de operação normal, usuários contratam um nível de serviço que satisfaz suas necessidades durante a maior parte do tempo, enquanto no modo de operação sobrecarregado, um nível de serviço mais elevado é estabelecido, até onde o provedor tem um incentivo para atribuir capacidade extra à aplicação para acomodar ocasionais picos de carga. Do ponto de vista de negócio, esta abordagem pode ser vantajosa tanto para os usuários, que pagam por capacidade extra apenas quando necessário, e para provedores, que podem oferecer planos mais atrativos por operarem com maior flexibilidade.

Para o modo de operação normal, é definida uma taxa de processamento X_i^{NSLA} para cada classe i, que o provedor deve atender se a taxa de chegada de requisições for alta o bastante. No caso de violações do contrato, o provedor concorda em devolver parte da cobrança pelo serviço a seus usuários. Para o modo de operação sobrecarregado, o contrato define $X_i^{SSLA} \geq X_i^{NSLA}$, a taxa de processamento limite até onde o usuário concorda em pagar uma recompensa ao provedor por servir requisições acima de X_i^{NSLA}. Os valores das penalidades e recompensas são calculados usando os parâmetros do contrato c_i

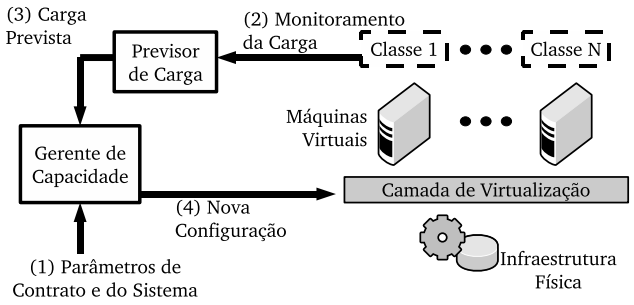


Figura 2. Gerenciamento de Capacidade Auto-Adaptativo (uma camada)

e r_i por unidade de taxa de processamento abaixo ou acima de X_i^{NSLA}, respectivamente.

A taxa de processamento válida é composta de todas as requisições cujo tempo de resposta satisfaça o especificado pelo contrato. Nós consideramos um requisito sobre o tempo de resposta que diz que o tempo de resposta das requisições da classe i não pode exceder um certo limite R_i^{SLA} mais do que $\alpha_i \times 100\%$ das vezes. Em outras palavras, $P(R_i > R_i^{SLA}) \leq \alpha_i$, onde R_i é o tempo de resposta de uma requisição da classe i.

4. Gerenciamento de Capacidade para Serviços Multi-Camadas

Aqui descrevemos nosso modelo auto-adaptativo para gerência de capacidade de serviços multi-camadas. A seção 4.1 apresenta o arcabouço auto-adaptativo, construído a partir de nosso modelo prévio de camada única [Abrahao et al. 2006]. A nova contribuição mais significativa reside no modelo de desempenho mais preciso e no modelo de otimização estendido criado para usá-lo, apresentados nas seções 4.2 e 4.3, respectivamente.

4.1. Arcabouço Auto-Adaptativo

Nosso arcabouço auto-adaptativo propõe um modelo de operação do sistema baseado em ciclos, mostrado na figura 2. A entidade central é o *gerente de capacidade*, que é chamado periodicamente para alocar a capacidade disponível em cada camada entre as aplicações hospedadas, objetivando maximizar o objetivo de negócio do provedor. Nos referimos ao intervalo entre intervenções consecutivas como *intervalo de controle*.

No final de cada *intervalo de controle*, o gerente de capacidade recebe uma previsão da carga de trabalho esperada para cada aplicação no próximo intervalo bem como os requisitos do contrato, os tempos médios de serviço (demandas) e das probabilidades de roteamento de cada classe, $p_{i,j}$. Esses parâmetros podem ser estimados num ambiente pré-produção como detalhado em [Urgaonkar et al. 2005a], e são usados para computar a fração da capacidade disponível em cada camada que deverá ser atribuída a cada aplicação. Eles são também usados para estimar a fração das requisições previstas que podem ser aceitas no sistema sem violar limitações de capacidade. A nova alocação de capacidade é então enviada para a camada de virtualização, que atualiza o mapeamento de recursos.

Os intervalos de controle podem ter duração fixa ou variável, dependendo das características do sistema e da estabilidade das cargas. Sua duração mínima é restringida pelo tempo que o gerenciador de capacidade gasta para reconfigurar o sistema. Estimativas futuras da carga de trabalho são fornecidas por um módulo previsor de carga que implementa algum método de previsão existente [Abraham and Ledolter 1983], e que um mecanismo de controle de admissão (como os discutidos em [Perros and Elsayed 2003])

SÍMBOLO	DESCRIÇÃO
X_i^{NSLA}	Taxa de processamento válida necessária da classe i no modo de operação normal.
X_i^{SSLA}	Taxa de processamento válida máxima da classe i no modo de operação sobrecarregado.
R_i^{SLA}	Requisito de tempo de resposta da classe i (seg).
α_i	Limite superior na probabilidade do tempo de resposta exceder R_i^{SLA} para uma requisição da classe i.
c_i	Valor da penalidade por unidade de taxa de processamento da classe i abaixo de X_i^{NSLA}.
r_i	Valor da recompensa por unidade de taxa de processamento da classe i acima de X_i^{NSLA}.
N	Número de classes hospedadas (VMs em cada camada).
K	Número de camadas.
$\nu_{i,j}$	Máxima utilização planejada para a VM da classe i na camada j.
$d_{i,j}^*$	Tempo médio de serviço de requisições da classe i na infraestrutura física da camada j quando rodando com capacidade total (seg).
$p_{i,j}$	Probabilidade das requisições da classe i deixarem o sistema antes de visitar a camada j.
λ_i^*	Taxa de chegada prevista (reqs/s) para a classe i.

Tabela 1. Parâmetros do Sistema, Previsor de Carga e Contrato

é usado para restringir a taxa de chegada de requisições no intervalo. A construção e avaliação destes módulos está fora do atual escopo.

Os parâmetros usados pelo gerente de capacidade, descrevendo o modelo de negócios, requisitos de contrato, configuração do sistema e características da carga são definidos na tabela 1. Nós assumimos que todas as requisições de uma classe são estatisticamente indistinguíveis, tendo todas o mesmo tempo médio de serviço na infraestrutura de cada camada (rodando com capacidade total), dados por $d_{i,j}^*$. O parâmetro $\nu_{i,j}$, um limite superior na utilização planejada da VM atribuída à classe i na camada j ($0 \leq \nu_{i,j} < 1$), é introduzido para evitar degradação do tempo de resposta devido à saturação, garantindo assim um certo nível de estabilidade nas VMs.

O gerente de capacidade é construído de um modelo de otimização que liga um modelo analítico de desempenho com o modelo de custo de duas camadas dirigido a contratos já apresentado na seção 3.2.

4.2. Modelo de Desempenho

Esta seção apresenta um modelo de filas analítico para estimar métricas de desempenho usadas pelo gerente de capacidade, a saber: utilização dos recursos em cada camada, taxa de processamento do sistema e a probabilidade de violação do tempo de resposta do contrato para cada classe. Nosso modelo assume que chegadas de requisições para cada classe seguem um processo de Poisson. Requisições da classe i aceitas no sistema chegam na primeira camada com taxa λ_i^{acc}. Nós assumimos que classes possuem tempos de serviço exponencialmente distribuídos em cada camada, deixando o estudo de outros padrões, específicos de aplicação, para trabalhos futuros.

Sob estas premissas, o sistema virtualizado multi-camadas é modelado como N redes de filas em seqüência, uma para cada aplicação, como mostrado na Figura 3. A

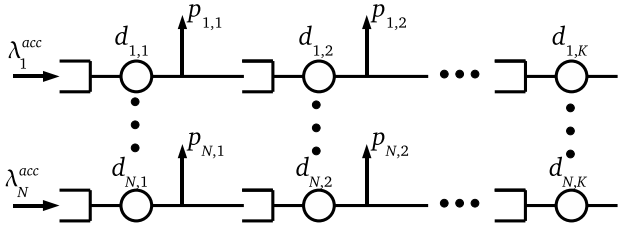


Figura 3. Modelo do Sistema Multi-Camadas.

VM atribuída a cada classe em cada camada é representada como uma fila M/M/1 com escalonamento PCPS [Kleinrock 1975]. Esta fila é comumente usada como um modelo razoável para centros de serviços transacionais [Villela et al. 2004].

Como é garantido que cada aplicação terá acesso a pelo menos a quantidade de recursos que foi atribuída a ela, nós estimamos o tempo médio de serviço de uma requisição da classe i na camada j, $d_{i,j}$, pelo tempo médio de serviço na camada com capacidade total, inflada pela fração atualmente atribuída à classe i, dada por $f_{i,j}$. Em outras palavras, $d_{i,j} = d^*_{i,j}/f_{i,j}$. Dadas as probabilidades de roteamento $p_{i,j}$, a taxa de chegada efetiva pra classe i na camada j é dada por $\lambda^e_{i,j} = \lambda^{acc}_i \prod_{k=1}^{j-1}(1 - p_{i,k})$. Note que $\lambda^e_{i,1} = \lambda^{acc}_i$. Além disso, $\lambda^e_{i,j} = \lambda^{acc}_i$ se $p_{i,j} = 0, \forall i \in N$ e $j = 1..K-1$.

Dada a condição de fluxo balanceado [Kleinrock 1975], que diz que todas as requisições aceitas são processadas, a taxa de processamento da classe i é dada pela taxa de chegada λ^{acc}_i. A utilização da camada j pela classe i é $\rho_{i,j} = \lambda^e_{i,j} d_{i,j}$ [Kleinrock 1975].

O componente mais desafiador do nosso modelo de desempenho é a estimativa da probabilidade de uma requisição da classe i violar o tempo de resposta do contrato. Dado o tempo de residência de uma requisição da classe i na camada j, $R_{i,j}$, e o tempo de resposta do sistema $R_i = \sum_{j=1}^{K} R_{i,j}$, nosso objetivo é estimar $P(R_i \geq R^{SLA}_i)$.

Note que $R_{i,j}$ é o tempo de resposta de uma fila M/M/1, exponencialmente distribuído com parâmetro $\gamma_{i,j} = 1/d_{i,j} - \lambda^e_{i,j}$ [Kleinrock 1975]. Lembre também que a soma de K variáveis exponenciais com taxas $\gamma_{i,j}$ ($j = 1..K$) segue uma distribuição hipoexponencial [Trivedi 2002]. Logo:

$$P(R_i \geq R^{SLA}_i) = \sum_{j=1}^{K} \left(\prod_{k=1, k\neq j}^{K} \frac{\gamma_{i,k}}{\gamma_{i,k} - \gamma_{i,j}} \right) e^{-\gamma_{i,j} R^{SLA}_i} \quad (1)$$

Devido ao fato deste modelo capturar o paralelismo que existe num ambiente multi-camadas, nós esperamos que a equação 1 seja mais precisa do que qualquer abordagem de camada única. Num modelo de camada única, a melhor aproximação é derivada do tempo de resposta de uma única fila M/M/1, exponencialmente distribuído com parâmetro $1/\sum_{j=1}^{K} d_{i,j} - \lambda^{acc}_i$ [Abrahao et al. 2006]. A equação 1, porém, é muito mais complexa, o que pode comprometer o tempo de solução do modelo. Assim, nós também consideramos uma aproximação derivada da Inequação de Chebyshev [Kleinrock 1975], que provê o seguinte limite superior à probabilidade de violações para a classe i:

$$P(R_i \geq R^{SLA}_i) \leq \frac{Var[R_i]}{(R^{SLA}_i - E[R_i])^2} \quad (2)$$

$$
\begin{aligned}
max \quad & \sum_{i=1}^{N} g_i(\lambda_i^{acc}) & & \\
s.t. \quad & 0 \leq \lambda_i^{acc} \leq min(\lambda_i^*, X_i^{SSLA}) & \forall i \in N \quad & (a) \\
& d_{i,j} = \frac{d_{i,j}^*}{f_{i,j}} & \forall i \in N, j \in K \quad & (b) \\
& \rho_{i,j} = \lambda_{i,j}^e d_{i,j} \leq \nu_{i,j} & \forall i \in N, j \in K \quad & (c) \\
& \sum_{i=1}^{N} f_{i,j} \leq 1 & \forall j \in K \quad & (d) \\
& f_{i,j} \geq 0 & \forall i \in N, j \in K \quad & (e) \\
& P(R_i \geq R_i^{SLA}) \leq \alpha_i & \forall i \in N \quad & (f)
\end{aligned}
$$

Figura 4. Modelo de Otimização do Gerente de Capacidade.

$E[R_i]$ e $Var[R_i]$ são a média e a variância do tempo de resposta para a classe i, dadas por $E[R_i] = \sum_{i=1}^{K} \frac{1}{\gamma_{i,j}}$ e $Var[R_i] = \sum_{i=1}^{K} (\frac{1}{\gamma_{i,j}})^2$.

4.3. Modelo de Otimização

O componente central do nosso gerente de capacidade é o modelo de otimização mostrado na figura 4. A sua principal variável de decisão é o vetor $f_{i,j}$. A função objetivo expressa o objetivo de negócio do provedor, dado pela soma, através de todas as classes, do balanço total das penalidades e recompensas. Vamos descrever primeiro as restrições.

A restrição (a) diz que a taxa admitida de requisições para cada classe é limitada pela taxa de chegadas predita e pela taxa de processamento máxima que dá lucro para o provedor quando a classe está operando em modo sobrecarregado. A restrição (b) define o tempo médio de serviço da classe i na sua VM da camada j. A restrição (c) limita a utilização na camada j pela classe i, $\rho_{i,j}$, à utilização máxima planejada para aquela VM. As restrições (d) e (e) impõem limites óbvios ao vetor $f_{i,j}$. Finalmente, a restrição (f) expressa o requisito de tempo de resposta do contrato. Duas variantes do modelo são criadas a partir do uso das equações 1 e 2 nessa restrição.

Voltemos agora à formulação de g_i, o lucro do provedor obtido da classe i. Lembre que recompensas são pagas ao provedor sempre que $\lambda_i^{acc} > X_i^{NSLA}$. A restrição (a) garante um limite superior nessas recompensas através de X_i^{SSLA}. Penalidades ocorrem sempre que $\lambda_i^{acc} < min(\lambda_i^*, X_i^{NSLA})$. Assim, o lucro total obtido da classe i, g_i, é:

$$
g_i = \begin{cases} -c_i \left(min(\lambda_i^*, X_i^{NSLA}) - \lambda_i^{acc} \right) & \lambda_i^{acc} \leq X_i^{NSLA} \\ r_i \left(\lambda_i^{acc} - X_i^{NSLA} \right) & \lambda_i^{acc} > X_i^{NSLA} \end{cases} \tag{3}
$$

Note que g_i, e conseqüentemente a função objetivo, cresce com λ_i^{acc}. Porém, λ_i^{acc} é limitada pela carga e contrato (restrição (a)), pelas limitações de utilização dos recursos (restrição (c)), e, acima de tudo, pelo requisito de tempo de resposta do contrato (restrição (f)). As duas últimas restrições ligam os valores de λ_i^{acc} às variáveis de decisão $f_{i,j}$.

O modelo de otimização mostrado na figura 4 é uma extensão, para ambientes multi-camadas, do proposto em [Abrahao et al. 2006]. Como anteriormente, o maior de-

safio, do ponto de vista da otimalidade e do tempo de solução, está na função objetivo linear por partes e na restrição (f) sobre o tempo de resposta. A função linear por partes pode ser transformada num conjunto de restrições lineares, facilmente tratáveis, se $c_i > r_i, \forall i$. Várias soluções foram encontradas para resolver os problemas criados pela restrição (f), a mais eficiente delas aproxima os pontos onde a função hipoexponencial não está definida ($\gamma_{i,x} = \gamma_{i,y}, x \neq y$), por uma distribuição de Erlang. Essa aproximação é assintoticamente exata e permite a solução veloz e correta do modelo.

Nosso modelo foi implementado e testado em AMPL [Fourer et al. 1993], uma linguagem de modelagem para programação matemática. Testamos diferentes solucionadores, e todos convergiram para a mesma solução em todos os testes.

5. Resultados

Nesta seção avaliamos a relação custo-eficiência do nosso arcabouço, comparando-o com nossa antiga estratégia auto-adaptativa de camada única [Abrahao et al. 2006] e com alocação de capacidade estática. A principal métrica de comparação é o balanço obtido por cada estratégia. Em nossos experimentos nós consideramos um ambiente com duas camadas ($K = 2$).

Nós avaliamos as duas variantes do nosso gerente de capacidade que estimam a probabilidade de violações do tempo de resposta com a distribuição hipoexponencial e a inequação de Chebyshev. No modelo de uma camada esta probabilidade é estimada a partir da distribuição exponencial do tempo de resposta de uma fila M/M/1 [Abrahao et al. 2006], com a demanda média de cada classe sendo a soma das demandas em cada camada. Finalmente, para alocação estática assumimos a melhor alocação de capacidade em cada camada, atribuindo uma fração fixa da capacidade da camada j à classe i proporcional à sua utilização média pela classe i através de toda a carga; usamos a distribuição hipoexponencial (equação 1) para calcular a taxa das requisições que podem ser admitidas no sistema de forma a satisfazer os requisitos de contrato. Nos referimos a estas estratégias como *hipoexponencial*, *Chebyshev*, *exponencial* e *estática*.

Nós construímos um simulador orientado por eventos que modela o sistema. Para as estratégias auto-adaptativas, o simulador é acoplado a um solucionador do modelo de otimização, que calcula o vetor de alocação de capacidade $f_{i,j}$ e as taxas de chegadas admitidas λ_i^{acc}, do próximo intervalo. Durante todos os intervalos, o tempo de resposta de cada requisição, bem como a taxa de processamento e utilização de todas as classes são coletados e usados para calcular o balanço do provedor.

Usamos um controle de admissão conservador que aceita uma requisição da classe i com probabilidade $\lambda_i^{acc}/\lambda_i^*$, mantendo a premissa de chegadas seguindo um processo Poisson para as requisições aceitas. Assumimos também que todas as requisições visitam ambas camadas, e que a utilização máxima planejada para todas as VMs é 95% ($p_{i,j} = 0$ e $\nu_i = 0.95$). Finalmente, como nosso foco é o custo-benefício do novo método, assumimos que não existe limite de tempo para adaptar o sistema e previsão ideal da carga, onde as taxas de chegadas futuras são conhecidas *a priori*. A seleção e avaliação de um método de previsão de carga é deixada para trabalho futuro. Todos os resultados apresentados são médias de 5 execuções (20 na seção 5.1), com desvio padrão abaixo de 2% das médias.

5.1. Gerente de Capacidade Escalável

Nós avaliamos quão escalável é nosso arcabouço multi-camadas para configurações com até 60 classes. Nosso foco foi na abordagem hipoexponencial, devido à sua maior com-

	Cenário 1		Cenário 2	
Classe i	$d^*_{i,1}(ms)$	$d^*_{i,2}(ms)$	$d^*_{i,1}(ms)$	$d^*_{i,2}(ms)$
1	0.6	0.4	1	0.7
2	0.4	0.6	0.7	1

Tabela 2. Tempo Médio de Serviço Por Camada para os Cenários 1 e 2

Cenário	R_i^{SLA}	X_i^{NSLA}	X_i^{SSLA}	c_i	r_i	α_i
1 e 2	0.1 s	500 req/s	1200 req/s	1.0	0.5	0.1
3	210 s	0.08 req/s	10 req/s	3500	1750	0.1
4	105 s	0.08 req/s	10 req/s	1750	875	0.1

Tabela 3. Valores dos Parâmetros do Modelo de Negócio ($i = 1..N$)

plexidade e maiores tempos de solução. Nossos experimentos foram realizados usando o solucionador não linear SNOPT [Gill et al. 2002], num computador com um processador AMD Sempron 2400 com 2GHz e 512MB de RAM.

O tempo médio de solução do modelo aumenta com o número de classes. Um ajuste linear dos dados indica que o tempo médio de solução do modelo, tipicamente abaixo de 1 segundo, cresce com uma fração pequena do número de classes. Concluindo que nosso gerente de capacidade escala bem para cenários práticos.

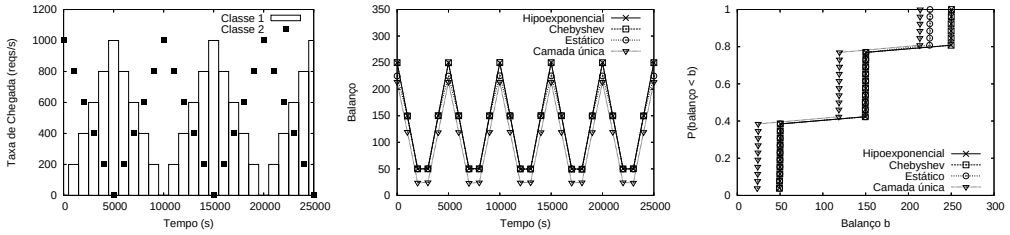
5.2. Cargas Sintéticas

Esta seção apresenta resultados de simulação para cargas sintéticas e duas classes de aplicação. Dois cenários ilustram os principais compromissos e benefícios de nossa solução. O intervalo de controle é escolhido como 1000 segundos em ambos os cenários.

No *cenário 1*, as requisições de cada classe chegam de acordo com o processo de Poisson periódico não-homogêneo mostrado na figura 5-a. Taxas de chegada variam desde 0 a 1000 requisições por segundo, com degraus e períodos de 1000 e 10000 segundos, respectivamente. Ambas as cargas possuem o mesmo perfil, porém com um deslocamento em seus períodos. Este é um cenário interessante para as abordagens auto-adaptativas, que podem redirecionar capacidade ociosa de VMs com baixa carga para outras que estejam sobrecarregadas. O gerente de capacidade é chamado ao final de cada intervalo, que coincide com os instantes quando as taxas de chegada mudam.

Tempos médios de serviço para cada classe em cada camada e valores dos parâmetros do modelo de negócio são mostrados nas tabelas 2 e 3, respectivamente. Note que as classes 1 e 2 possuem gargalos nas camadas 1 e 2, respectivamente. Neste caso, nossas abordagens auto-adaptativas multi-camadas conseguem atribuir dinamicamente, pra cada aplicação, mais recursos na camada que ela mais necessita. De qualquer forma, note que o desbalanceamento no tempo de serviço não é muito significativo. Além disso, ambas as classes possuem os mesmos valores para os parâmetros do modelo de negócio por que nosso interesse está em avaliar a eficiência relativa das abordagens analisadas.

A figura 5-b mostra o balanço final do provedor para cada modelo ao longo da simulação. O padrão repetitivo da curva é devido ao padrão periódico das cargas. As abordagens hipoexponencial e Chebyshev resultam em balanços quantitativamente similares ao longo da simulação. Notamos que ambas abordagens provêem ganhos marginais (11%) sobre a abordagem estática quando as classes possuem cargas complementares, mesmo sendo esse o melhor cenário para as abordagens auto-adaptativas, onde elas po-



(a) Perfis das Cargas (b) Balanço X Tempo (c) FDC do Balanço

Figura 5. Cargas e Resultados de Simulação para Cargas Sintéticas: Cenário 1

dem reatribuir capacidade entre VMs. Isto deve-se ao fato da carga imposta em cada camada ser muito leve. Assim, a capacidade de auto-adaptação não é um diferencial. Os lucros são ditados principalmente pelas oportunidades de capitalizar de aplicações em modo de operação sobrecarregado, que são as mesmas para todas as abordagens.

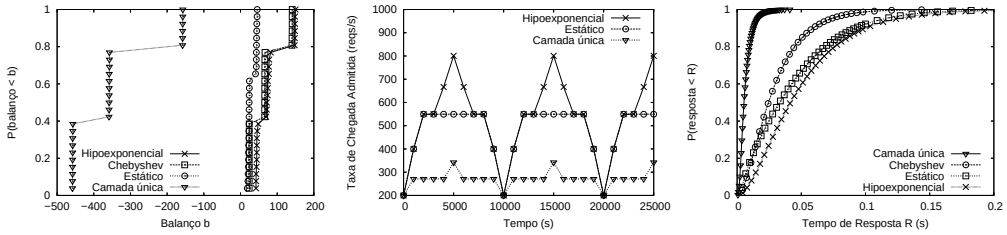
Por outro lado, o tempo de serviço médio usado pela abordagem exponencial é 1ms, fazendo com que a infraestrutura esteja sub-provisionada para as taxas de requisições agregadas. Os ganhos no balanço total resultante de nossas abordagens auto-adaptativas multi-camadas sobre a de camada única variam de 17% (nos picos) a 103% (nos vales). De fato, quando ambas as cargas possuem carga similar, a abordagem de camada única, baseada num modelo simplificado do sistema, é significativamente sobrepujada até pela abordagem estática. A figura 5-c sumariza estes resultados, mostrando as distribuições acumuladas do balanço em todos os intervalos. As abordagens auto-adaptativas multi-camadas apresentam retorno total 28% maior que a abordagem de camada única.

Vamos avaliar agora o *cenário 2*, caracterizado por uma carga de trabalho mais pesada e com tempos médios de serviço em cada camada mais desbalanceados. Os perfis de carga são idênticos aos mostrados na figura 5-a, porém as taxas variam de 200 a 1200 chegadas por segundo. Os parâmetros do sistema são mostrados nas tabelas 2 e 3.

A figura 6-a mostra o balanço do provedor para cada abordagem. A abordagem hipoexponencial leva a melhores balanços que a Chebyshev, quando as taxas de requisições de cada classe estão balanceadas. O ganho no balanço total é cerca de 20%. Como esperado, o erro da aproximação fica mais significativo com cargas mais pesadas. Ao contrário do cenário prévio, ambas abordagens superam significativamente a estratégia estática (até 580%) em intervalos onde as classes possuem taxas de chegada complementares. Além disso, a estratégia de camada única, com retornos flutuando entre -458 e -157, é sobrepujada pelas três estratégias por ordens de magnitude.

Notamos que dois fatores primários que impactam a custo-eficiência das estratégias de gerenciamento de capacidade são habilidade de adaptar-se a mudanças na carga e a precisão do modelo de desempenho, que impactam as decisões de alocação e controle de admissão. A alocação de capacidade fixa penaliza significativamente a abordagem estática para cargas pesadas e heterogêneas (ex.: cenário 2). Além disso, em ambos os cenários analisados (e em um cenário omitido com cargas totalmente balanceadas e aplicações homogêneas), a abordagem de camada única é significativamente penalizada pelo seu modelo de desempenho simples. Para satisfazer as restrições de tempo de resposta, a abordagem de camada única é forçada a fazer decisões de alocação e controle de admissão mais conservadoras, levando a balanços piores.

Esta conclusão é ilustrada nas figuras 6-b e 6-c, que mostram a taxa de requisição



(a) FDC do Balanço (b) λ_1^{acc} (c) FDC do Tempo de Resposta

Figura 6. Métricas de Desempenho para Cargas Sintéticas: Cenário 2

Cenário / Média dos Tempos de Serv. (s)		Classe i							
		1	2	3	4	5	6	7	8
3	$d^*_{i,1}$	0.5	2.0	1.5	3.0	-	-	-	-
	$d^*_{i,2}$	3.0	1.5	2.0	0.5	-	-	-	-
4	$d^*_{i,1}$	0.25	1.25	0.75	0.9	0.85	1.0	0.5	1.5
	$d^*_{i,2}$	1.5	0.5	1.0	0.85	0.9	0.75	1.25	0.25

Tabela 4. Tempo Médio de Serviço Por Camadas nos Cenários 3 e 4.

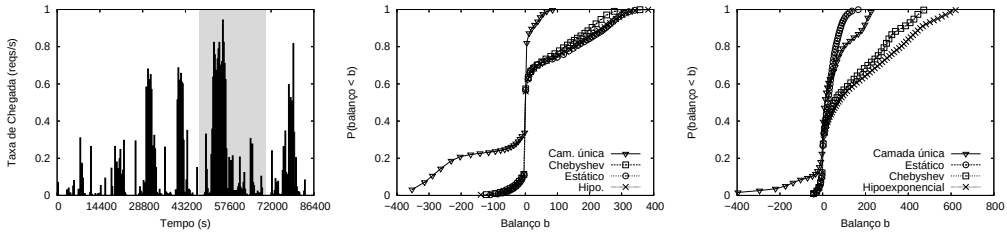
admitida e a distribuição do tempo de resposta para cada classe. As taxas para a abordagem de Chebyshev, omitidas, estão entre as taxas das abordagens hipoexponencial e estática. Note que a decisão de alocação e controle de admissão mais agressiva tomada pelas abordagens hipoexponencial e estática levam a um maior número de violações do tempo de resposta. De qualquer forma, a figura 6-c mostra que a restrição do contrato ($P(R > 0.1) < 0.1$)) é satisfeita por todas as abordagens.

Finalmente, para verificar o impacto de erros na predição da carga de chegada, rodamos simulações com diferentes intervalos de controle para o *cenário 2*. Nós escolhemos durações de intervalos que não coincidem com os instantes nos quais a carga muda. Para intervalos com durações de 300 e 600 segundos, a diminuição do lucro comparado com os resultados mostrados na figura 6-a é apenas 5% (8%) e 11% (15%), respectivamente, para o modelo hipoexponencial (Chebyshev). Concluímos que nossa solução é razoavelmente robusta a erros de predição de carga.

5.3. Perfis de Carga Realistas

Nesta seção avaliamos as abordagens de gerência de capacidade para cargas mais realistas. Construimos novas cargas a partir de logs contendo o número de chegadas, para intervalos de 5 minutos, de 4 aplicações reais de comércio eletrônico, por um período de 3 meses (de 23/11/2004 a 23/2/2005). Acordos de confidencialidade nos impedem de informar as fontes. Todos os quatro logs possuem perfis similares, com picos nos mesmos instantes. Taxas de requisição variam muito, com pico de 17 e uma média de 0.078 requisições por segundo. A figura 7-a mostra a variação da taxa de chegada para uma classe num dia típico. Cargas realistas são construídas assumindo que as requisições seguem processos Poisson, com taxas mudando a cada intervalo de 5 minutos contidos nos logs.

Dois novos cenários são considerados. No *cenário 3*, simulamos as 4 classes dos nossos logs. O *cenário 4* usa um número maior, 8, de aplicações, cujas cargas são construídas duplicando cada um dos 4 logs originais, e deslocando as requisições duplicadas 6 horas para o futuro. O tempo médio de serviço em cada camada é escolhido de forma a



(a) Carga da Casse 1 (b) Cenário 3 (c) Cenário 4

Figura 7. Perfil da Carga e FDC do Balanço para Cargas Realistas

fazê-la sub-provisionada para atender a carga agregada. Além disso, a demanda agregada em cada camada bem como os requisitos de taxa de processamento do contrato são fixos e dados nas tabelas 3 e 4. O intervalo de controle é ajustado para 5 minutos.

As distribuições acumuladas do balanço para ambos os cenários estão mostrados nas figuras 7-b e 7-c. A abordagem hipoexponencial resulta nos melhores balanços. No cenário 3, a estratégia estática é tão boa quanto a hipoexponencial. Os perfis de carga muito similares das quatro aplicações deixam pouco espaço para ganhos devido ao gerenciamento dinâmico. Note, porém, a degradação significativa no cenário 4, que tem mais oportunidades de alocação dinâmica de capacidade entre as 8 classes. Neste caso, a abordagem hipoexponencial resulta em aumento de 429% no balanço total.

Como no cenário 2, a abordagem hipoexponencial apresenta melhor custo-benefício do que a abordagem Chebyshev, levando a ganhos de 20% e 26% no retorno médio diário nos cenários 3 e 4, respectivamente. Novamente, a abordagem de camada única é sobrepujada por ordens de magnitude.

6. Conclusões e Trabalhos Futuros

Neste artigo apresentamos um arcabouço auto-adaptativo de gerenciamento de capacidade para sistemas virtualizados multi-camadas. Ele implementa um modelo de desempenho de múltiplas filas preciso, que captura gargalos específicos de aplicação e o paralelismo inerente a arquiteturas multi-camadas. Nós rodamos simulações para vários cenários. Nossas principais conclusões são três. Primeiro, nossa solução auto-adaptativa multi-camadas é escalável e significativamente mais eficiente que alocação estática para cargas pesadas e desbalanceadas. Segundo, o modelo de desempenho simplificado e impreciso de camada única leva a decisões de alocação conservadoras, que comprometem sua eficiência. Finalmente, nossa abordagem multi-camadas é robusta e pode ser aplicada para gerência de capacidade de ambientes virtualizados sujeitos a variação de capacidade.

Direções para trabalhos futuros incluem: (a) estender nossos modelos para diferentes padrões de tráfegos de aplicação e modelar os recursos de cada camada individualmente; (b) incluir custos operacionais e de gerência (energia); (c) conceber modelos de negócio mais ricos; e (d) avaliação adicional da aplicabilidade do nosso arcabouço para ambientes virtualizados sob ataques.

Agradecimento

Este trabalho foi desenvolvido em colaboração com a HP Brasil R&D.

Referências

Abraham, B. and Ledolter, J. (1983). *Statistical Methods for Forecasting*. John Wiley and Sons.

Abrahao, B., Almeida, V., Almeida, J., Zhang, A., Beyer, D., and Safai, F. (2006). Self-Adaptive SLA-Driven Capacity Management for Internet Services. In *IEEE/IFIP NOMS Conference*, Vancouver, Canada.

Almeida, J., Almeida, V., Ardagna, D., Francalanci, C., and Trubian, M. (2006). Resource Management in the Autonomic Service-Oriented Architecture. In *IEEE International Conference on Autonomic Computing*, Dublin, Ireland.

Barham, P., Dragovic, B., Fraser, K., Hand, S., Harris, T., Ho, A., Neugebauer, R., Pratt, I., and Warfield, A. (2003). Xen and the Art of Virtualization. In *19th ACM SOSP*, Bolton Landing, NY.

Chase, J., Anderson, D., Thakar, P., Vahdat, A., and Doyle, R. (2001). Managing Energy and Server Resources in Hosting Centers. In *18th ACM SOSP*, Banff, Canada.

Fourer, R., Gay, D., and Kernighan, B. (1993). *AMPL: A Modeling Language for Mathematical Programming*. Boyd and Fraser.

Gill, P., Murray, W., and Saunders, M. (2002). SNOPT: An SQP Algorithm for Large-Scale Constrained Optimization. *SIAM Journal on Optimization*, 12(4):979–1006.

Kleinrock, L. (1975). *Queueing Systems*. John Wiley and Sons.

Liu, X., Zhu, X., Singhal, S., and Arlitt, M. (2005). Adaptive Entitlement Control to Resource Containers on Shared Servers. In *9th IFIP/IEEE IM*, Nice, France.

Liu, Z., Squillante, M. S., and Wolf, J. L. (2001). On Maximizing Service-Level-Agreement Profits. In *3rd ACM Conference on Electronic Commerce*, Tampa, Florida.

Menascé, D. and Bennani, M. (2006). Autonomic Virtualized Environments. In *IEEE International Conference on Autonomic and Autonomous Systems*, Silicon Valley, CA.

Perros, H. G. and Elsayed, K. M. (2003). Call Admission Control Schemes: A Review. *IEEE Communications Magazine*, 34(11):82–91.

Ross, J. W. and Westerman, G. (2004). Preparing for Utility Computing: The Role of IT Architecture and Relationship Management. *IBM Systems Journal*, 43(1):5–19.

Trivedi, K. (2002). *Probability and Statistics with Reliability, Queuing and Computer Science Applications*. John Wiley and Sons.

Urgaonkar, B., Pacifici, G., Shenoy, P., Spreitzer, M., and Tantawi, A. (2005a). An Analytical Model for Multi-Tier Internet Services and its Applications. In *MASCOTS*, Banff, Canada.

Urgaonkar, B., Shenoy, P., Chandra, A., and Goyal, P. (2005b). Dynamic Provisioning of Multi-Tier Internet Applications. In *2nd IEEE ICAC*, Seattle, Washington.

Villela, D., Pradhan, P., and Rubenstein, D. (2004). Provisioning Servers in the Application Tier for e-Commerce Systems. In *12th IEEE WQoS*, Passau, Germany.

Whitaker, A., Shaw, M., and Gribble, S. (2002). Scale and Performance in the Denali Isolation Kernel. In *5th OSDI*, Boston, MA.

Wilkes, J., Mogul, J., and Suermondt, J. (2004). Utilification. In *11th ACM SIGOPS European Workshop: Beyond the PC*, Leuven, Belgium.