Modelagem de Desempenho de Plataformas Servidoras Multi-Camadas

Itamar Viana[1], **João Palotti**[1],
Genaína Rodrigues[1], **Jussara Almeida**[1], **Virgílio Almeida**[1]

[1]Departamento de Ciência da Computação – Universidade Federal de Minas Gerais
Belo Horizonte, Brasil – 31270-010

{itamar, palotti, genaina, jussara, virgilio}@dcc.ufmg.br

Abstract. *Performance modeling is a central task in the capacity management of server platforms. Traditional performance models are created for transactional workloads (purely open models), batch or interactive (purely closed models). However, many real Web applications experience session-based workloads that exhibit a partially-open behavior, which includes components from either open or closed model, particularly with respect to response time. In this paper, we developed an analytical model for performance that captures the main aspects of partially-open workloads of multi-layered Web platforms and compared it with the open and closed models by simulation.*

Resumo. *A modelagem de desempenho é uma tarefa central para o gerenciamento de capacidade de plataformas servidoras. Modelos de desempenho tradicionais são derivados para cargas de trabalho transacionais (modelos puramente abertos), batch ou interativas (modelos puramente fechados). Entretanto, várias aplicações reais da Web consistem em cargas baseadas em sessões com comportamento semi-aberto que incluem componentes dos modelos abertos e fechados, particularmente em relação ao tempo de resposta. Neste artigo, desenvolvemos um modelo analítico de desempenho que captura aspectos de cargas semi-abertas para plataformas Web multi-camadas e analisamos seus compromissos em relação aos modelos fechado e aberto por meio de simulação.*

1. Introdução

Cada vez mais a Internet tem sido um meio fundamental para a realização de transações comerciais. No entanto, para viabilizar essas transações faz-se necessário uma infra-estrutura tal que grande parte das empresas não pode custear a hospedagem de seus serviços. Além de uma configuração de hardware e software de grande porte, essa infra-estrutura deve atender as requisições recebidas obedecendo aos limites acordados quanto à qualidade de serviço prestado por meio de contratos de nível serviço (SLA - *Service Level Agreement*). Esses contratos são firmados entre os clientes e os provedores de serviço, onde violações dos contratos firmados podem representar grandes perdas. Por exemplo, a Amazon observou que uma latência extra de 100ms em suas transações afeta 1% das vendas, enquanto a Google percebeu que cada 0,5 segundos gastos a mais na geração de páginas de busca reduzem seu tráfego em 20% [Linden 2006].

Portanto, um fator fundamental para os provedores de serviço é atender ao contrato SLA enquanto conseguem maximizar seus lucros, o que requer estratégias de

gerenciamento de capacidade com boa relação custo-benefício. Na literatura, existem várias estratégias para o gerenciamento de capacidades [Cunha et al. 2007, Chen et al. 2005, Menascé and Bennani 2006a]. Um componente central dessas estratégias de gerenciamento é o modelo de desempenho através do qual estimativas do tempo de resposta dos serviços podem ser providas. Entre esses modelos de desempenho, existem os trabalhos que utilizaram a abordagem de redes de filas para servidores multi-camadas [Cunha et al. 2007, Urgaonkar et al. 2007, Zhang et al. 2007]. Porém, eles apenas consideram modelos de carga baseados na taxa de chegada de requisições da plataforma Web [Cunha et al. 2007], caracterizando uma carga puramente aberta, ou na população média de usuários que utilizam o sistema simultaneamente [Urgaonkar et al. 2007, Zhang et al. 2007], caracterizando uma carga puramente fechada. No entanto, aplicações reais da Web executam serviços que exibem cargas de trabalho híbridas que não são nem puramente fechadas nem puramente abertas, elas apresentam comportamento de cargas semi-abertas [Schroeder et al. 2006]. No modelo de carga semi-aberto, sessões chegam no sistema de forma independente como no modelo aberto, porém essas sessões interagem com o sistema como no modelo fechado, ou seja, o envio de novas requisições depende da taxa de processamento da plataforma.

Neste trabalho, avaliamos diferentes estratégias de modelagem para previsão de desempenho de aplicações da Web que executam serviços com cargas semi-abertas. Em particular, propomos e comparamos este modelo com os modelos tradicionais, puramente abertos e puramente fechados. No cenário investigado, consideramos uma arquitetura multi-camadas, pois as aplicações Web modernas são em sua grande maioria sistemas complexos compostos principalmente de servidores de apresentação HTTP (p.ex., Apache), servidores de negócio (p.ex., Tomcat) e servidores de persistência (p.ex., MySQL) [Menascé and Almeida 2001]. Cada camada apresenta um nível de multiprogramação (MPL) diferente. Esse nível é definido, usualmente, para não permitir excessos na utilização dos recursos disponíveis.

Estudamos os compromissos dos modelos aberto, fechado e semi-aberto via simulação. Estendemos o simulador proposto em [Cunha et al. 2007, Cunha et al. 2008] para incorporar o conceito de sessão e as restrições de MPL por camada, visando modelar de forma realista uma plataforma servidora Web. Na análise de resultados, utilizamos vários cenários nos quais avaliamos qual a precisão dos modelos nas estimativas do tempo de resposta médio em relação ao simulador para diferentes níveis de multiprogramação e intensidades da carga.

Por fim, nossos resultados mostram os principais compromissos dos modelos aberto, fechado e semi-aberto em relação às estimativas do tempo de resposta ao variar os limites de MPL por camada. No caso que o limite de MPL é justo, o modelo fechado apresentou os melhores resultados, seguido do modelo semi-aberto. Porém, de acordo com o aumento do limite de MPL, os modelos semi-aberto e aberto tornam-se melhores do que o fechado, sendo o semi-aberto superior ao aberto. Em todos os cenários de teste, as estimativas do tempo de resposta do modelo semi-aberto foram intermediárias às estimativas dos modelos fechado e aberto, sendo que o modelo fechado tende a subestimar o tempo de resposta e o aberto a superestimar. Sendo assim, em geral, o modelo semi-aberto foi melhor do que os outros modelos. Atribuímos isso a três fatores fundamentais. Em primeiro lugar, nosso modelo de desempenho leva em consideração o conceito de sessões do usuário que não esta presente no modelo puramente aberto. Em segundo lugar, consideramos a distribuição de sessões no sistema, enquanto que no modelo fechado é utilizado

apenas a média do número de sessões. Em terceiro lugar, consideramos que a rejeição de uma requisição em uma camada impacta no número de visitas por requisição nas outras camadas, aprimorando, portanto, o modelo puramente fechado proposto em [Urgaonkar et al. 2007].

Este artigo está organizado como se segue. Na seção 2 discutimos modelos de desempenho comuns na literatura, o modelo aberto e o fechado. Na seção 3 apresentamos nosso modelo analítico de desempenho baseado em cargas semi-abertas. Na seção 4 analisamos os principais compromissos dos modelos aberto, fechado e semi-aberto por meio de simulação e, finalmente, na seção 5 tecemos nossas conclusões e elucidamos as direções futuras do nosso trabalho.

2. Revisão da literatura

Nessa seção iremos apresentar de forma conceitual os modelos de desempenho fechado e aberto que utilizamos como base. O modelo aberto foi utilizado em [Cunha et al. 2007], num contexto de gerenciamento de capacidade e o modelo fechado foi proposto em [Urgaonkar et al. 2007].

Os modelos fechado e aberto, assim como o modelo semi-aberto possuem como premissa que os tempos de serviço em cada camada são exponencialmente distribuídos. O impacto dessa premissa foi avaliado em um contexto de gerenciamento de capacidade em [Cunha et al. 2008] para casos em que os tempos de serviço reais não seguem distribuições exponenciais, apresentando diferentes níveis de variabilidade. As conclusões apresentadas em [Cunha et al. 2008] mostram que o modelo baseado na premissa de tempos exponencialmente distribuídos levou a resultados melhores que vários outros modelos que explicitamente capturam a maior variabilidade nos tempos de serviço.

2.1. Modelo Aberto

O modelo aberto tomado como base foi utilizado em [Cunha et al. 2007], em um contexto de gerenciamento de capacidade. Nesse modelo a carga é baseada em requisições, sendo que uma requisição é independente da outra e a chegada das requisições seguem um processo Poisson. Uma requisição chega na primeira camada e deixa o sistema na camada i com probabilidade p_i. Foi assumido que cada camada possui tempo de serviço exponencialmente distribuído, sendo a infra-estrutura modelada por uma rede de filas M/M/1 como mostrado na figura 1. Nessa figura, λ^r, simboliza a taxa de chegada de requisições, d_i, a demanda de cada requisição na camada i e Q_i a fila que modela a camada i. O ponto forte do modelo utilizado em [Cunha et al. 2007] é que ele oferece uma forma de calcular a probabilidade $P(r \geq R^{SLA})$ do tempo de resposta r das requisições excederem um tempo limite, R^{SLA} (definido no contrato SLA) quando as filas são M/M/1. Essa probabilidade é calculada utilizando a inversa da distribuição de probabilidade hipoexponencial dada por, $P(r \geq R^{SLA}) = \sum_{i=1}^{K} \left(\prod_{k=1,k\neq i}^{K} \frac{\gamma_k}{\gamma_k - \gamma_i} \right) e^{-\gamma_i R^{SLA}}$, sendo $\gamma_i = \frac{1}{d_i} - \lambda^r$.

O problema do modelo aberto é que ele não captura a interação do usuário com o sistema e não modela comportamento de sessão que os usuários apresentam. Devido a isso, ele tende a superestimar o tempo de reposta [Schroeder et al. 2006]. Uma vantagem do modelo aberto é sua simplicidade, sendo de fácil acoplamento a um modelo de otimização [Cunha et al. 2007], podendo ser utilizado, sem modificações, em uma estratégia de gerenciamento de capacidade.

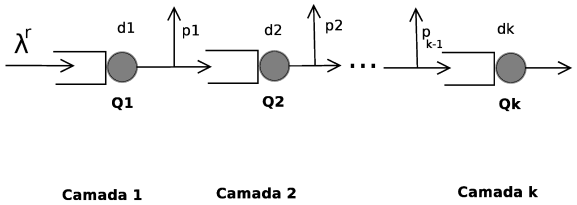


Figura 1. Rede de Filas - Modelo Aberto

2.2. Modelo Fechado

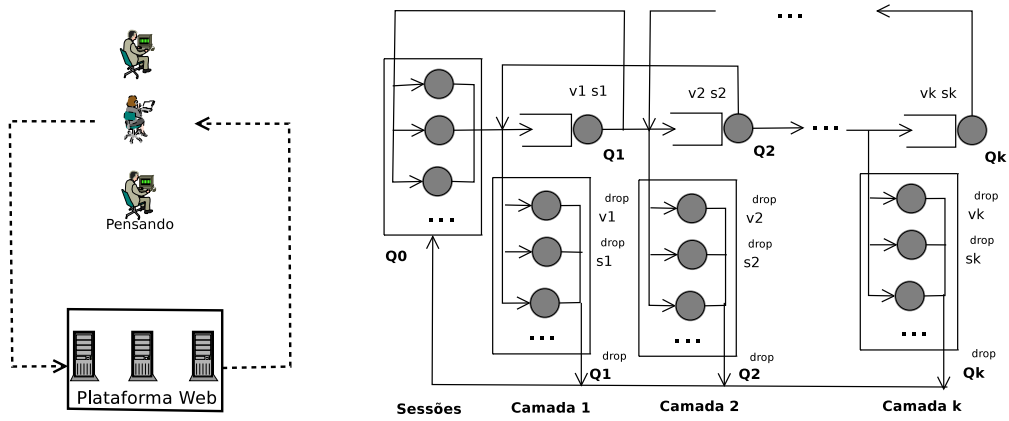
O modelo fechado utilizado como base foi proposto em [Urgaonkar et al. 2007]. Ele é capaz de modelar o comportamento de sessão e capturar a interação dos usuários com o sistema. Um usuário é responsável por uma sessão no servidor, sendo que suas requisições são submetidas em seqüência. Após receber uma resposta de uma requisição, o cliente gasta um tempo pensando, para enviar outra requisição, como mostrado na figura 2-a. Portanto a taxa de chegada de requisições é dependente da taxa de processamento da infra-estrutura. No modelo fechado é assumido que o número de sessões no sistema é fixo, sendo que quando uma sessão termina outra sessão ocupa seu lugar imediatamente.

No modelo fechado, a plataforma Web foi modelada por meio de uma rede de filas Q1 . . . Qk , sendo que cada fila representa uma camada da aplicação, como mostrado na figura 2-b. A política de serviço utilizada na modelagem das filas foi *processor sharing*, devido a essa política ser uma boa aproximação para as políticas de escalonamento presentes nos sistemas operacionais.

Com relação à modelagem de sessão, o modelo fechado assume que existe um número fixo de sessões executando no sistema, sendo utilizado no modelo o número médio de sessões que executaram simultaneamente no sistema. Uma sessão apenas gera uma requisição após receber a resposta da requisição anterior e ter passado um tempo pensando. Em sistemas reais, quando uma requisição chega na camada i, ela gera uma ou mais requisições na camada $i + 1$. Para capturar este comportamento no modelo é permitido que uma requisição faça múltiplas visitas para cada fila durante sua execução. Isto é alcançado com a introdução de uma transição de cada fila para a sua predecessora, como mostrado na Figura 2-b. Após uma requisição ser processada na fila Q_i ela pode retornar para a fila Q_{i-1} com uma probabilidade p_i ou seguir para a fila Q_{i+1} com uma probabilidade $(1 - p_i)$. As únicas exceções são a fila da última camada Q_k, na qual todas as requisições devem retornar para a fila anterior e a primeira fila Q_1, na qual a transição para a fila anterior representa o fim da requisição.

Para capturar as requisições que são rejeitadas na camada i devido ao nível de multiprogramação ser alcançado, foi adicionada uma transição na fila Q_i para um subsistema de filas com infinitos centros de serviços Q_i^{drop}, com tempos de serviço S_i^{drop}, como mostrado na Figura 2-b. Requisições que são rejeitadas na fila Q_i, sofrem algum atraso, S_i^{drop}, no subsistema Q_i^{drop} antes de retornarem para Q_0. Isto modela o atraso entre quando a requisição é rejeitada na camada i e quanto tempo leva para esta informação se propagar até o cliente que iniciou a requisição.

A metodologia proposta por [Urgaonkar et al. 2007] utiliza um algoritmo iterativo conhecido como Análise de Valores Médios (MVA) [Lazowska et al. 1984] para calcular o tempo de resposta médio e a taxa de processamento média das requisições. Um parâmetro importante desse modelo é o número de visitas, V_i que uma requisição faz em média



(a) Alto nível (b) proposto por [Urgaonkar et al. 2007]

Figura 2. Modelo Fechado em alto nível e proposto por [Urgaonkar et al. 2007]

por camada e a taxa de requisições que são rejeitadas, V_i^{drop}, na camada i [1]. A taxa de visitas V_i é calculada utilizando a cadeia de Markov mostrada na figura 3, para quando não existe limite de MPL. O estado Q_0 representa o usuário enviando uma requisição e os estados $Q_1 \ldots Q_k$ representam as camadas da plataforma Web, sendo que as variáveis p_i representam a probabilidade de uma requisição estando na camada i retornar para a camada $i-1$. O sistema linear 1 foi montando a partir da cadeia de Markov, mostrada na figura 3, considerando que a quantidade de visitas de um estado é igual ao fluxo que entra naquele estado e utilizando a lei de conservação do fluxo [Lazowska et al. 1984], que diz que todo fluxo que entra em um estado deve sair do mesmo.

Quando existem limites de multiprogramação, é possível fornecer as probabilidades p_i^{drop} de estouro dos limites L_i utilizando uma heurística que modela cada camada como uma fila M/M/1/L_i, de capacidade L_i. Com isso, o valor de V_i^{drop} é dado por $V_i \times p_i^{drop}$ e a taxa de visitas V_i deve ser atualizada para $V_i(1 - p_i^{drop})$. Porém, essa heurística não considera que a mudança na quantidade de visitas de uma camada afeta na quantidade de visitas das outras camadas. Apresentaremos na seção 3, uma heurística que captura a dependência entre as taxas de visitas das camadas quando existem limites de MPL.

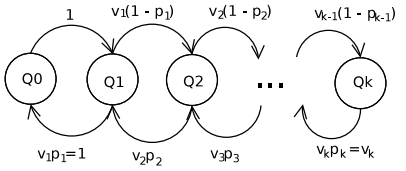


Figura 3. Taxa de Visitas de Uma Requisição em Uma Camada

$$
\begin{aligned}
V_1 &= 1 + V_2 \times p_2 \\
V_2 &= V_1 \times (1 - p_1) + V_3 \times p_3 \\
V_3 &= V_2 \times (1 - p_2) + V_4 \times p_4 \\
&\ldots \\
V_j &= V_{j-1} \times (1 - p_{j-1}) + V_{j+1} \times p_{j+1} \quad (1 < j < k) \\
&\ldots \\
V_k &= V_{k-1} \times (1 - p_{k-1})
\end{aligned}
\tag{1}
$$

[1]Lembrando que a demanda das requisições por camada i é dada por $D_i = S_i \times V_i$ e o tempo de atraso que as requisições rejeitadas na camada i é dado por $D_i^{drop} = S_i^{drop} \times V_i^{drop}$.

Um problema do modelo fechado é que ele não captura a variação do número de sessões executando no sistema, ele assume que esse número é fixo, devido a isso, o modelo fechado tende a subestimar o tempo de resposta e a taxa de processamento de requisições [Schroeder et al. 2006]. Outro problema é que o modelo fechado utiliza um algoritmo iterativo (MVA), sendo ele de difícil acoplamento a um modelo de otimização. Uma proposta de modificação para o acoplamento do modelo fechado a um modelo de otimização pode ser encontrada em [Trivedi 2002], porém nessa proposta o modelo fechado perde precisão em suas estimativas de desempenho.

3. Modelo Semi-aberto

Como mencionado na seção 2, os modelos aberto e fechado não capturam todas as características de cargas Web reais [Menascé and Bennani 2006b, Schroeder et al. 2006]. O primeiro não admite a existência de sessões, enquanto o segundo não captura a distribuição do número de sessões no sistema. Devido a isso, propomos um modelo de desempenho que utiliza um modelo de carga semi-aberto para capturar melhor o comportamento de sistemas Web reais. Este modelo é um modelo híbrido que mescla as características dos modelos fechado e aberto. O modelo semi-aberto fornece como saída o tempo de resposta por requisição, a taxa de processamento de requisições e a probabilidade do tempo de resposta exceder um limite definido no contrato SLA.

Com relação ao modelo semi-aberto, em [Schroeder et al. 2006] foi feita uma análise de desempenho em uma plataforma realista comparando o desempenho do sistema quando aplicado à cargas aberta, fechada e semi-aberta. Mas, não foi apresentado uma modelagem analítica para o problema. Já, em [Menascé and Bennani 2006b] foi feita uma modelagem analítica, mas apenas para o caso de camada única. Além disso, não foi capturado o conceito de sessão do usuário e a precisão do modelo semi-aberto apresentado não foi avaliada com relação a outros modelos.

Na Figura 4, mostramos nosso modelo em alto nível, no qual sessões chegam seguindo um processo Poisson. Cada sessão possui um tamanho, $nreq$, que segue uma distribuição Poisson representando o número de requisições que uma sessão gera antes de terminar. Uma requisição só será enviada por uma sessão após ela receber a resposta da anterior e passar um tempo pensando. Comparado ao modelo fechado em [Urgaonkar et al. 2007], nosso modelo de desempenho captura a distribuição no número de sessões no sistema, enquanto que no outro modelo [Urgaonkar et al. 2007] foi utilizado o valor médio do número de sessões e assumido que após uma sessão deixar a plataforma, outra ocupa seu lugar imediatamente.

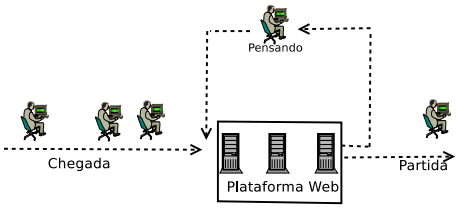


Figura 4. Interação dos usuários com a plataforma

Primeiramente, como dito na seção 2, vamos explicar a heurística que propomos para estimar o número médio de visitas, V_i, que uma requisição faz em uma camada e a taxa de requisições,V_i^{drop} ,que são rejeitadas na camada i, sendo que, esse cálculo da

taxa de visitas é um aprimoramento do que foi proposto em [Urgaonkar et al. 2007]. Na Figura 5, temos a versão estendida da Figura 3 para quando existem diferentes níveis de multiprogramação (MPL) por camada. Os estados Q_i ($1 \leq i \leq K$) representam as requisições que foram servidas com sucesso na camada i. Foram adicionados novos estados D_i e Tp_i para cada estado da cadeia anterior. Os estados D_i representam as requisições que foram rejeitadas na camada i e os estados Tp_i foram adicionados para facilitar a modelagem da cadeia, com eles não é preciso alterar os valores p_i utilizados na cadeia anterior. Como na cadeia anterior, as arestas representam o número de requisições que saem de um estado para outro. Para simplificar sua visualização, não foram colocados todos os valores das arestas na Figura 5. Esses valores podem ser facilmente deduzidos, pois a soma das probabilidades das transições que saem de qualquer estado devem somar 1. Por fim, os valores p_i^d representam a probabilidade de uma requisição ser rejeitada p_i^{drop} na camada i.

Na metodologia apresentada por [Urgaonkar et al. 2007] não é possível saber a priori os valores de p_i^{drop}. Nós propomos uma nova metodologia para estimá-los. Para capturar a mudança dos valores do número de visitas de uma camada nas outras camadas propomos um processo iterativo. No caso, sempre que for atualizado a taxa de visitas V_i por $V_i \times (1 - p_i^{drop})$ deve ser atualizado a taxa de visitas da próxima camada utilizando o sistema de equações 2, sendo que o cálculo dos valores de V_i e V_i^{drop} devem ser repetidos até que eles convirjam para algum valor ou seja alcançado um número limite de interações.

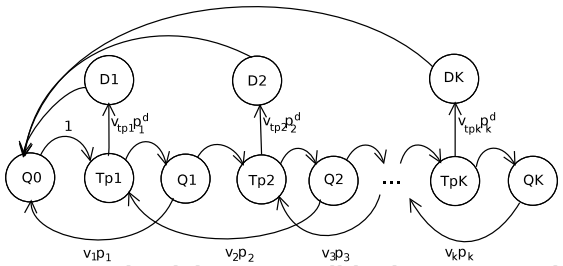


Figura 5. Taxa de Visitas Estendida de Uma Requisição

Tendo como base o fluxo que entra em cada estado, podemos gerar o sistema de Equações 2. A partir desse sistema linear, podemos perceber que a mudança na taxa de visitas V_i afetará na taxa de visitas V_{i+1}, diferente do que foi proposto por [Urgaonkar et al. 2007] e mostrado nos sistema de Equações 1.

$$V_{Tpi} = \begin{cases} 1 + V_2 \times p_2 & i = 1 \\ V_{i-1} \times (1 - p_{i-1}) + V_{i+1} \times p_{i+1} & 1 < i < k \\ V_{i-1} \times (1 - p_{i-1}) & i = k \end{cases}$$

$$V_i = V_{Tpi} \times (1 - p_i^{drop}) \quad 1 \leq i \leq k$$

$$V_i^{drop} = V_{Tpi} \times p_i^{drop} \quad 1 \leq i \leq k \tag{2}$$

Agora vamos explicar como nosso modelo semi-aberto faz para calcular o tempo de resposta médio por requisição e a taxa de processamento média de requisições. Para capturar a distribuição do número de usuários no sistema, utilizamos a cadeia de Markov apresentada na Figura 6. Dessa forma, podemos determinar a taxa de processamento de sessões e requisições, além do tempo de resposta por requisição baseado na distribuição do número de sessões ativas. Cada estado dessa cadeia representa o número de sessões ativas na plataforma. A transição do estado N para o estado $N + 1$ acontece com a taxa

de chegada de sessões λ e a transição do estado $N+1$ para o estado N ocorre com a taxa de processamento de sessões $x(N)$ quando existem N sessões na plataforma. Os valores de $x(N)$ são calculados utilizando o modelo fechado [Urgaonkar et al. 2007] para quando existem N usuários executando no sistema, sendo que ao utilizar o modelo fechado imediatamente estamos capturando que a plataforma possui limites de MPL por camada, pelo fato desse modelo capturar isso. Iremos limitar a cadeia de Markov apresentada na figura 6 pelo número máximo de sessões, M, que podem estar executando no sistema, ou seja, o limite superior de sessões que a infra-estrutura consegue servir sem que alguma camada apresente utilização de 100%. A utilização de cada camada dada, portanto, por $U_i = \lambda \times S_i \times V_i \times nreq$, sendo $nreq$ o número médio de requisições de uma sessão.

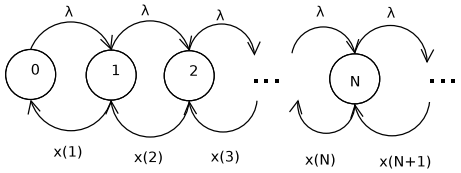


Figura 6. Cadeia de Markov do número de sessões no sistema

Como base na cadeia de Markov apresentada na figura 6 montamos um sistema de equações lineares para determinar qual a probabilidade $P(n)$ de existirem n $(0 \leq n \leq M)$ sessões executando no sistema. Esse sistema linear pode ser resolvido aplicando-se as seguintes equações $P(0) = 1/(1 + \sum_{j=1}^{M} \prod_{i=1}^{j} \frac{\lambda}{x(i)})$ e $P(n) = P(0) \prod_{i=1}^{n} \frac{\lambda}{x(i)}$

Temos assim que o número médio de sessões no sistema N^{avg} é dado por $N^{avg} = \sum_{j=1}^{M} P(j) \times j$. Finalmente, pela lei de conservação do fluxo [Lazowska et al. 1984], a taxa de processamento médio de sessões será igual a taxa de chegada de sessões λ e a taxa de processamento médio de requisições será $\lambda/nreq$. Pela lei de Little [Lazowska et al. 1984], podemos derivar o tempo de resposta por requisição R^r como $R^r = (N^{avg}/\lambda - Z^r \times (nreq - 1))/nreq$, sendo Z^r o tempo gasto pensando por requisição.

Por fim, iremos utilizar a desigualdade de Markov [Kleinrock 1975] para calcular a probabilidade de uma requisição exceder o tempo de resposta definido no contrato SLA, $P(r \geq R^{SLA}) = R^r/R^{SLA}$, sendo R^r o tempo de reposta médio. Iremos utilizar essa desigualdade pelo fato dela apenas precisar como entrada do tempo de resposta médio. Também iremos utilizar a desigualdade de Markov no modelo fechado, pois em [Urgaonkar et al. 2007] não foi apresentado uma forma de calcular $P(r \geq R^{SLA})$.

4. Análise de Resultados

Esta seção analisa e compara os resultados obtidos com os modelos analíticos e os obtidos com o simulador. As métricas consideradas são o tempo de resposta médio por requisição R^r e a probabilidade do tempo de resposta ser maior do que o especificado no contrato SLA, R^{sla}. Os resultados apresentados para o simulador são valores médios de 5 execuções, com coeficiente de variação abaixo de 2%. Na sub-seção 4.1 iremos apresentar o modelo do simulador seguido da análise dos resultados na sub-seção 4.2.

4.1. Modelo do Simulador

O simulador utilizado para avaliar os modelos analíticos é uma extensão do que foi proposto em [Cunha et al. 2007, Cunha et al. 2008]. Em [Cunha et al. 2007] foi desen-

Tabela 1. Parâmetros do Simulador

Símbolo	Definição
λ	Taxa de chegada de sessões por segundo
$nreq$	Tamanho da sessão
K	Número de camadas
S_k	Tempo de serviço na camada k
S_k^{drop}	Tempo de atraso na camada k para as requisições rejeitadas
p_k	Probabilidade da requisição retornar para a camada $k-1$
Z^r	Tempo pensando em segundos

volvido um simulador orientado a eventos, no qual cada camada era modelada por meio de uma fila, com política primeiro a chegar, primeiro a ser servido (FCFS) e com tempo de serviço distribuído exponencialmente. O simulador anterior não possuía o conceito de sessões, apenas capturava a chegada de requisições, sendo que a chegada de uma nova requisição era independente das anteriores. Além disso, ele não capturava que as camadas possuem restrições nos limites de MPL.

Para incorporar o comportamento de sessão e as restrições de MPL por camada, modificamos o simulador com base no modelo do sistema apresentado na Seção 3. O simulador estendido modela a chegada de sessões, λ, por meio de um processo Poisson. Modela também as restrições de MPL por camada, L_k, e o tempo de atraso, S_k^{drop}, que as requisições rejeitadas sofrem, exponencialmente distribuído. Cada sessão possui um tamanho, também exponencialmente distribuído, $nreq$, que representa o número de requisições antes do término da mesma. Uma nova requisição somente será enviada por um usuário de uma sessão após receber resposta da requisição prévia e de passar um tempo pensando, Z^r, que é exponencialmente distribuído. Em síntese, os parâmetros utilizados no simulador são mostrados na Tabela 1.

Temos ainda que após uma requisição ser processada na fila Q_k ela pode retornar para a fila Q_{k-1} com uma probabilidade p_k ou seguir para a fila Q_{k+1} com uma probabilidade $(1-p_k)$. As únicas exceções são a fila da última camada Q_K, na qual todas as requisições devem retornar para a fila anterior e a primeira fila Q_1, na qual a transição para a fila anterior representa o fim da requisição. Além disso, quando uma requisição for rejeitada, a sessão que enviou a requisição irá tentar reenviar essa requisição novamente um número de vezes pré-definido até desistir.

4.2. Resultados

Nos testes, configuramos a infra-estrutura para possuir duas camadas, isto é $K = 2$, com tempo de serviço homogêneo, $S_1 = S_2 =$ 0,01s. Temos que a probabilidade de uma requisição retornar para uma camada anterior é $p_1 =$ 0,5 e $p_2 =$ 1,0, o tempo médio que o usuário gasta pensando é $Z^r =$0,5s, o tamanho médio da sessão é $nreq = 4$, o número máximo de vezes que o usuário reenvia uma requisição é 3 e o tempo de atraso médio para as requisições rejeitadas em ambas as camadas é de $1s$. As demandas das camadas foram escolhidas com base nos valores apresentados em [Menascé and Bennani 2006b].

Mantivemos fixo o limite de MPL da segunda camada em $L_2 = 1000$. Esse valor foi escolhido para que essa camada não rejeite requisições, pois iremos focar no nível de multiprogramação da primeira camada, facilitando a análise dos resultados. Nos testes,

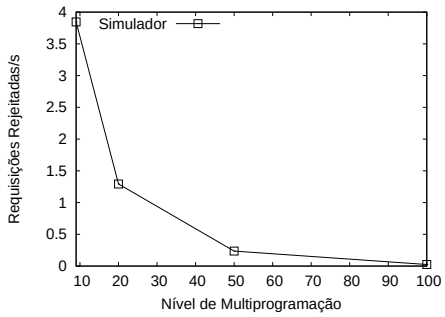


Figura 7. Taxa de requisições rejeitadas para $\lambda = 12$ (fixo)

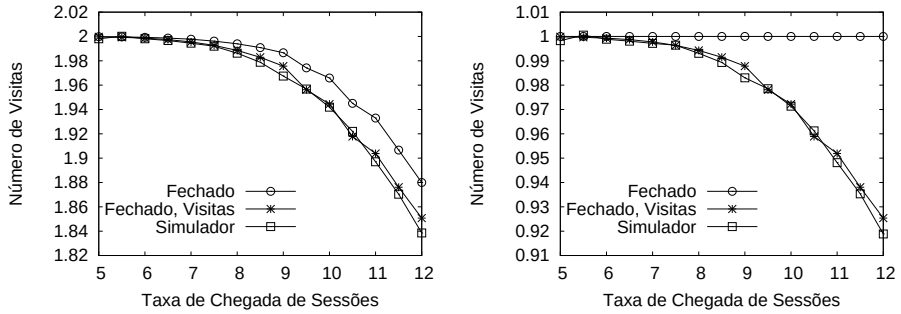
variamos o limite de MPL da primeira camada, L_1, e a taxa de chegada sessões λ. Os resultados qualitativos são os mesmos para quando variamos o limite de MPL da segunda camada. Utilizamos para avaliar os modelos em relação ao simulador, o conceito de erro absoluto dado pela equação, $ERRO = ((\sum_{i=1} |e_i|)/(\sum_{i=1} y_i)) \times 100$, sendo e_i a diferença entre o valor do modelo analítico e do simulador e y_i os valores coletados do simulador.

Para cada teste, variamos a taxa de chegada de sessões λ de 8 a 12 sessões/s com incrementos de 0,5. Os outros parâmetros foram mantidos fixos. Executamos quatro testes, um para cada valor do limite de MPL da primeira camada ($L_1 = \{9, 20, 50, 100\}$) e com $L_2 = 1000$. O objetivo desses testes foi verificar o impacto do limite de multiprogramação na precisão dos modelos aberto, semi-aberto e fechado. O valor de $L_2 = 1000$ é como se a segunda camada não possuísse limite de MPL. Assumindo que a camada K é modelada por uma fila M/M/1, o número médio de requisições em cada fila é dado por $\frac{U_k}{1-U_k}$ [Menascé et al. 2004]. Isso significa que, para os valores de $L_i = \{9, 20, 50, 100, 1000\}$ a utilização máxima da camada ficará restrita a respectivamente 90%, 95%, 98%, 99% e 100% [2]. Porém, quanto menor for o limite de multiprogramação mais requisições serão rejeitadas. A Figura 7 mostra a taxa de requisições rejeitadas por segundo em relação ao limite de multiprogramação para a taxa de chegada de sessões fixa em $\lambda = 12$. Podemos perceber que à medida que o limite de multiprogramação aumenta a taxa de requisições rejeitadas diminui, porém o tempo de reposta médio de requisição aumenta.

Começamos avaliando nossa estratégia para calcular a quantidade média de visitas que uma requisição faz por camada proposta na sessão 3. A Figura 8-a mostra os resultados para a primeira camada e a Figura 8-b os da segunda camada. A curva, com o rótulo "Fechado, Visitas" é a nossa estratégia e a curva com rótulo "Fechado" é a estratégia proposta por [Urgaonkar et al. 2007]. Os resultados mostrados são para valores de $L_1 = 9$ e $L_2 = 1000$. Para a primeira camada podemos perceber que ambas as estratégias foram qualitativamente semelhantes, com nossa estratégia sendo ligeiramente melhor. Porém, considerando a segunda camada, à medida que o valor de λ cresce, o erro da estratégia proposta por [Urgaonkar et al. 2007] também aumenta, para $\lambda = 12$ sessões/s ela apresentou um valor 9% maior do que o medido no simulador enquanto que a nossa estratégia apresentou um valor apenas 1% maior. Isso foi devido à nossa estratégia considerar que a rejeição de uma requisição em uma camada impacta no número de visitas por requisição das outras camadas.

Agora avaliamos os resultados dos tempos de reposta por requisição dos modelos

[2] A utilização U_k da camada k é dada por $U_k = d_k^r \times X$, sendo d_k^r a demanda por requisição na camada k e X a taxa de processamento de requisições do sistema.



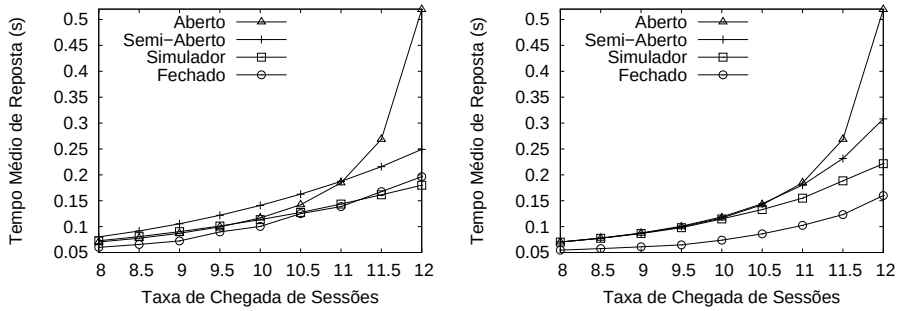
(a) Camada 1 (L_1=9) (b) Camada 2 (L_2=1000)

Figura 8. Taxa de visitas x Taxa de chegada de sessões (L_1 de 9 e 20)

aberto [Cunha et al. 2007], fechado [Urgaonkar et al. 2007] e semi-aberto (nossa proposta) em relação ao simulador. As Figuras 9-a e 9-b mostram o tempo de resposta médio por requisição para os valores de $L_1 = 9$ e $L_1 = 20$, respectivamente, e a tabela 2 mostra os erros absolutos de cada modelo em relação ao simulador para cada valor de L_1. Considerando a Figura 9-a, quando o limite de MPL da camada 1 é igual a $L_1 = 9$, o modelo fechado apresentou o melhor resultado entre os três modelos, com um erro de 10%. O modelo aberto comportou-se bem para valores de λ menores do que 10 sessões/s ($U_1 = 80\%$), mas começou a superestimar o tempo de reposta para valores de λ superiores a 10. Isso acontece porque tal modelo não considera que existem limites de MPL por camada, sendo que para $\lambda > 10$, a probabilidade do sistema rejeitar requisições tornar-se significativa. No geral, o modelo aberto apresentou um erro absoluto de 48%. O modelo semi-aberto, assim como o modelo aberto, superestimam o tempo de reposta. Mas, nesse caso, é interessante observar que nosso modelo apresentou um resultado intermediário entre o fechado e o aberto para uma taxa de chegada de sessões (λ) maior do que 11 sessões/s. Por fim, nosso modelo apresentou um erro absoluto de 27%, que foi menor do que o erro do modelo aberto.

Ao elevar o limite de multiprogramação da camada L_1 de 9 para 20 ($L_1 = 20$, Figura 9-b), podemos notar que o modelo semi-aberto passou a apresentar os melhores resultados, estando ele novamente entre o modelo fechado e aberto. O modelo fechado subestimou o tempo de resposta para todos os valores de λ e o modelo aberto comportou-se bem para valores de λ menores do que 10,5 sessões/s, mas para valores maiores que esse, ele começou a superestimar os tempos de reposta. Com relação aos erros absolutos, ao elevar L_1 de 9 para 20, o erro do modelo aberto caiu de 48% para 36% (queda de 25%), do modelo semi-aberto caiu de 27% para 15% (queda de 44%) e o erro do modelo aberto subiu de 10% para 32% (alta de 220%). Podemos perceber através da tabela 2 que esse comportamento se mantém para os valores de L_1 iguais a 50 e 100 também. A partir disso, concluímos que o modelo semi-aberto no geral é a melhor opção. Entretanto para quando o limite de MPL é pequeno, o modelo fechado é uma boa escolha. Porém ele tende a subestimar os valores do tempo de reposta, e isso para o gerenciamento ou planejamento de capacidade levará a uma má qualidade de serviço para os usuários. Portanto, o modelo semi-aberto continua sendo mais interessante do que o fechado mesmo quando o limite de multiprogramação for pequeno.

Por fim, vamos analisar a precisão dos modelos analíticos em calcular a probabilidade do tempo de resposta ultrapassar o requisito do tempo de resposta do contrato SLA, que nesse caso foi definido como 0,5s. Na Figura 10 são mostrados os resultados para o modelo aberto, semi-aberto e fechado para quando o limite de MPL da primeira camada



(a) L_1 = 9 (b) L_1 = 20

Figura 9. Tempo de resposta x Taxa de chegada de sessões (L_1 de 9 e 20)

Tabela 2. Erros Absolutos em Relação ao Tempo de Reposta, variando λ

L_1	Aberto(%)	Semi-Aberto(%)	Fechado(%)
9	48	27	10
20	36	15	32
50	13	6	37
100	3	2	34

é $L_1 = 20$ e a tabela 3 mostra os erros absolutos para cada modelo em relação aos valores do simulador para os valores de L_1 iguais a 9, 20, 50 e 100. Com relação aos resultados mostrados na Figura 10, podemos perceber que o modelo aberto apresentou os melhores resultados com erro absoluto de 34%, seguido do modelo fechado com erro de 122% e do modelo semi-aberto com erro 201% para L_1=20. O modelo aberto foi melhor do que os outros pelo fato dele estimar $P(r \geq R^{SLA})$ com base nas propriedades da distribuição hipoexponencial enquanto que os modelos semi-aberto e fechado utilizam apenas a desigualdade de Markov, sendo que ela baseia-se apenas na média do tempo de resposta. O modelo fechado foi melhor do que o modelo semi-aberto, no cálculo de $P(r \geq R^{SLA})$, pelo fato dele subestimar o tempo de resposta. Por fim, podemos observar na tabela 3 que esse comportamento se mantém para os outros valores de L_1. Com isso concluímos que se for necessário estimar $P(r \geq R^{SLA})$ é melhor utilizar o modelo aberto ou desenvolver métodos que considerem os aspectos da distribuição do tempo de reposta, por exemplo, o desvio padrão, para os modelos fechado e semi-aberto.

5. Conclusão e trabalhos futuros

Neste artigo, propomos e desenvolvemos um modelo analítico de desempenho baseado em teoria de filas que explicitamente captura aspectos de cargas de trabalho semi-abertas para plataformas servidoras Web multi-camadas. Sabemos que as aplicações reais da Web executam serviços que exibem comportamento de sessões semi-abertas, não sendo nem puramente abertas e nem puramente fechadas [Schroeder et al. 2006]. Por-

Tabela 3. Erros Absolutos em Relação à $P(r \geq R^{SLA})$, variando λ

L_1	Aberto(%)	Semi-Aberto(%)	Fechado(%)
9	102	404	321
20	34	201	122
50	15	157	95
100	13	152	92

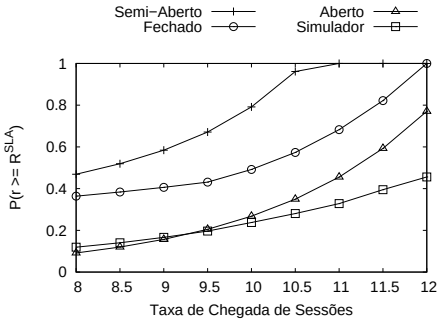


Figura 10. Probabilidade do tempo de resposta ser maior do que 0,5 s (L_1=20)

tanto, modelos tradicionais podem nem sempre ser precisos para o gerenciamento de capacidade efetivo para tais aplicações. Validamos nosso modelo híbrido por meio de simulação, comparando os resultados do mesmo com os puramente abertos e fechados.

Os resultados mostraram que, para limites pequenos de MPL, o modelo semi-aberto é consideravelmente mais preciso que o modelo aberto, aproximando-se do modelo fechado. No entanto, esse último subestima o tempo de resposta, ao contrário do modelo semi-aberto, que se apresenta mais adequado para o gerenciamento de capacidade. Para maiores limites de MPL, o modelo semi-aberto apresenta os menores erros, sendo melhor que o modelo aberto em todos os casos. Porém, o modelo aberto apresenta a melhor previsão para a probabilidade de uma requisição demorar mais tempo que o definido no contrato SLA, $P(r \geq R^{SLA})$. Julgamos que isso se deve ao fato que o modelo aberto obtém suas estimativas com base nas propriedades da distribuição hipoexponencial enquanto que os modelos semi-aberto e fechado utilizam apenas a desigualdade de Markov.

Para decidir qual modelo de desempenho utilizar em um sistema real deve ser feito uma caracterização da sua carga. Com isso será possível definir se a carga do sistema em questão pode ser modelada como aberta, fechada ou semi-aberta. Após feito a caracterização, escolha o modelo de desempenho que melhor adequa-se à sua carga. Em [Schroeder et al. 2006] foi apresentado as principais características dos modelos de carga fechado, aberto e semi-aberto, isso irá ajudá-lo a definir qual modelo sua carga ajusta-se melhor.

As direções para trabalhos futuros incluem: (a) avaliar os modelos de carga aberto, fechado e semi-aberto em um protótipo real, como por exemplo o RUBiS [Amza et al. 2002], pois trata-se de um *benchmark* renomado. (b) Acoplar nossa solução de modelo de carga semi-aberto no contexto de gerenciamento de capacidade, comparando os resultados com os outros modelos. (c) Obter uma melhor previsão para a probabilidade do tempo de resposta atender ao que foi definido no contrato SLA para os modelos fechado e semi-aberto.

Agradecimento

Esta pesquisa foi desenvolvido em colaboração com a HP Brasil R&D e é parcialmente financiada pelo Instituto Nacional de Ciência e Tecnologia para a Web - INCTWeb (MCT/CNPq 573871/2008-6) e pelo Projeto REBU (CTInfo/CNPq 55.0995/2007-2).

Referências

Amza, C., Cecchet, E., Chanda, A., Cox, A., Elnikety, S., Gil, R., Marguerite, J., Rajamani, K., and Zwaenepoel, W. (2002). Specification and implementation of dynamic web site benchmarks. In *Proc. IEEE 5th Workshop on Workload Characterization*.

Chen, Y., Das, A., Qin, W., Sivasubramaniam, A., Wang, Q., and Gautam, N. (2005). Managing Server Energy and Operational Costs in Hosting Centers. In *Proc. SIGMETRICS*, Banff, Alberta, Canada.

Cunha, I., Almeida, J., Almeida, V., and Santos, M. (2007). Self-Adaptive Capacity Management for Multi-Tier Virtualized Environments. In *Proc. 10th IEEE/IFIP IM*, Munich, Germany.

Cunha, I., Viana, I., João Palotti, J. A., and Almeida, V. (2008). Analyzing Security and Energy Tradeoffs in Autonomic Capacity Management. In *IEEE/IFIP NOMS*, Salvador, Brazil.

Kleinrock, L. (1975). *Theory, Volume 1, Queueing Systems*. Wiley-Interscience.

Lazowska, E. D., Zahorjan, J., Graham, G. S., and Sevcik, K. C. (1984). *Quantitative system performance: computer system analysis using queueing network models*. Prentice-Hall, Inc., Upper Saddle River, NJ, USA.

Linden, G. (2006). Make Data Useful. Stanford University, http://glinden.blogspot.com/.

Menascé, D. and Bennani, M. (2006a). Autonomic Virtualized Environments. In *Proc. 2nd IEEE ICAS*, Silicon Valley, CA.

Menascé, D. A. and Almeida, V. (2001). *Capacity Planning for Web Services: metrics, models, and methods*. Prentice Hall PTR, Upper Saddle River, NJ, USA.

Menascé, D. A., Almeida, V. A. F., and Dowdy, L. W. (2004). *Performance by Design: Computer Capacity Planning by Example*. Prentice Hall.

Menascé, D. A. and Bennani, M. N. (2006b). Analytic performance models for single class and multiple class multithreaded software servers. In *Int. CMG Conference*, pages 475–482. Computer Measurement Group.

Schroeder, B., Wierman, A., and Harchol-Balter, M. (2006). Open versus closed: a cautionary tale. In *NSDI'06: Proceedings of the 3rd conference on 3rd Symposium on Networked Systems Design & Implementation*, pages 18–18, Berkeley, CA, USA. USENIX Association.

Trivedi, K. S. (2002). *Probability and statistics with reliability, queuing and computer science applications*. John Wiley and Sons Ltd., Chichester, UK, UK.

Urgaonkar, B., Pacifici, G., Shenoy, P., Spreitzer, M., and Tantawi, A. (2007). Analytic modeling of multitier internet applications. *ACM Trans. Web*, 1(1):2.

Zhang, Q., Cherkasova, L., and Smirni, E. (2007). A regression-based analytic model for dynamic resource provisioning of multi-tier applications. In *Proc. ICAC*, Washington, DC, USA. IEEE.