Avaliação de Desempenho em Web Clusters para E-Commerce: algoritmos de escalonamento e disciplinas para filas

CAIO P. SABO[1]
RODRIGO F. DE MELLO[1]
REGINA H. C. SANTANA[2]
MARCOS J. SANTANA[2]

USP - Universidade de São Paulo – Instituto de Ciências Matemáticas e de Computação
[1]Departamento de Ciências de Computação, [2]Departamento de Sistemas de Computação
CEP 13560-970 – São Carlos – SP
caiosabo@gmail.com, (mello,rcs,mjs)@icmc.usp.br

Resumo. Sites de *e-commerce* têm adotado sistemas baseados em servidores Web distribuídos, contudo, em situações de sobrecarga, pode-se deixar de atender requisições de venda devido ao aumento de requisições de navegação. Neste contexto é proposto um modelo de Servidor Web para E-Commerce (*SWE-C*), validado por meio de simulações e uma carga sintética com diversas combinações de algoritmos de escalonamento e disciplinas de atendimentos para filas, dentre as quais se destaca uma política orientada à prioridades para CPU. Experimentos comprovam que essa política aumenta o *throughput* de transações de venda – onde a motivação é o fato de gerarem renda – com menores tempos de resposta em situações de sobrecarga. Além disso, aumenta o *throughput* de requisições genéricas ao longo do tempo.

Palavras-Chave: servidor web distribuído, avaliação de desempenho.

Performance Evaluation of E-commerce Web Clusters: Scheduling Algorithms and Queue Policies

Abstract. E-commerce web sites, under high load situations, have been using distributed web servers such as the Web Cluster architecture. In such situations, the site can define priorities among selling and navigation requests. In this context, the E-commerce web server model (*SWE-C*) is proposed, evaluated by simulations and a synthetic workload, combining scheduling algorithms and different queue policies, confirmed that the proposed policy, CPU-oriented, allows to increase the selling transaction throughput – which generates profit – with the lower response times in overloaded circumstances.

Keywords: distributed web servers, performance evaluation.

(Received November 21, 2006 / Accepted March 12, 2007)

1 Introdução

O crescimento da quantidade de serviços de acesso remoto oferecidos na Web têm contribuído para o aumento de sua popularidade. Esse crescimento pode ser notado com base em diversas análises recentes [1]. A alta demanda de usuários pode gerar gargalos, gerando aumento nos tempos de atendimento. Surge então a necessidade de servidores Web capazes de suportar um grande número de acessos com desempenho aceitável.

Uma das soluções mais simples e utilizadas para obtenção de escalabilidade em servidores Web baseia-se na atualização de hardware. Essa estratégia não resolve o problema de forma efetiva e não tem uma relação

custo-benefício satisfatória [1]. Outra opção é baseada em melhoramentos nos sistemas operacionais dos servidores, construção de servidores Web mais eficientes e implementação de políticas de escalonamento de requisições.

Essas soluções são consideradas para sites Web com aumento freqüente de carga, onde um único *host* com o servidor Web pode não atender requisitos de escalabilidade e desempenho necessários [3]. A arquitetura baseada em servidores Web distribuídos é uma alternativa efetiva pois tem menor custo e é escalável. Um servidor Web distribuído consiste de múltiplos *hosts* conectados que se beneficiam do poder computacional agregado para tratar requisições de clientes [11].

O tipo de arquitetura para servidor Web distribuído mais adotado é o *Web cluster* composto por computadores em rede local. Esses computadores aplicam políticas de escalonamento para definir qual elemento de processamento deve atender requisições. A política adotada tem impacto no desempenho e na escalabilidade do sistema.

Atualmente pode-se encontrar na literatura algoritmos e políticas para balanceamento de carga em *Web clusters*. Todavia, observa-se uma lacuna relativa a estudos comparativos utilizando cargas de trabalho com conteúdo estático[1] e dinâmico[2]. Essa limitação motivou o desenvolvimento deste trabalho que apresenta um estudo e avaliação de desempenho de alternativas para o balanceamento de carga gerada por páginas estáticas e dinâmicas em servidores Web distribuídos adotando diferentes políticas.

Este artigo é subdividido nas seguintes seções: A seção 2 aborda o problema e o modelo de carga sintética desenvolvido. Na Seção 3 é descrito o cenário real utilizado para a geração de uma carga de trabalho para *e-commerce*. A Seção 4 apresenta detalhes das modificações realizadas no software servidor Web Apache para coletar o consumo de CPU de requisições Web atendidas. A Seção 5 apresenta um modelo de servidor Web distribuído para *e-commerce*. A Seção 6 traz detalhes da validação do modelo. Na Seção 7 são apresentados resultados. Por fim, as conclusões e referências.

2 Problema Abordado e Modelagem

Sites de *e-commerce* são submetidos a altas taxas de requisições as quais justificam a utilização de uma estrutura com servidores Web para atender o tráfego de requisições com desempenho aceitável. Essa demanda motivou este trabalho que tem por objetivo avaliar o desempenho de políticas de atendimento de requisições em *clusters* de servidores Web. Para avaliar o desempenho são necessárias informações de carga de sites com tais características. Para isso reproduziu-se um cenário de comércio eletrônico e a geração de uma carga de trabalho sintética que retratasse o comportamento de consumidores.

Com base em trabalhos relacionados ao assunto [9], um modelo de carga de carga de trabalho para sites de comércio eletrônico foi desenvolvido. Esse modelo considera páginas das categorias: *Navegar* que envolve navegação e pesquisa, mas nenhuma atividade de pedido de produtos. As interações típicas que abrangem essa categoria são: Página principal, Listar categorias, Visualizar detalhes de produto e Pesquisar produto; *Pedir* que envolve somente atividades de pedido de produtos, e incluem as seguintes interações: Login, Local de entrega, Forma de pagamento e Confirmação de pedido.

Um modelo difundido para definir transições entre diferentes categorias de páginas Web é apresentado por Singhmar [6]. Esse modelo utiliza uma máquina de estados finitos, onde as probabilidades são estacionárias (fixadas estaticamente) e as transições de estados sem memória, ou seja, não necessitam de informações do estado atual para assumirem outro (Figura 1). Estes estados correspondem a tipos de páginas que caracterizam interações de um site de *e-commerce* e os arcos representam transições entre eles.

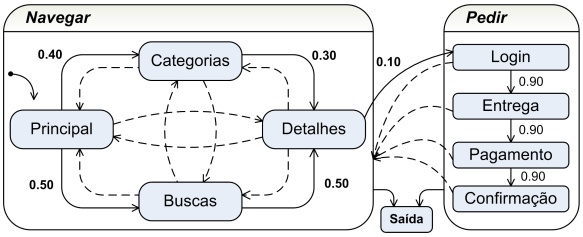

Figura 1: Máquina de estados finitos [6]

Sabe-se que a probabilidade máxima de transição de um estado para outro é 1, o que não acontece no modelo proposto por Singhmar [6], onde os arcos pontilhados são assumidas como probabilidade zero de transição. A Figura 2 ilustra uma nova abordagem para a máquina de estados finitos apresentada em [6], porém, com probabilidades diferentes de zero nos arcos pontilhados.

No modelo de carga de trabalho proposto, subtraiu-se a probabilidade máxima 1 da soma das probabilidades de transição de cada estado para os demais. Dessa maneira, pode-se calcular a probabilidade para eventos representados por arcos pontilhados.

[1]Páginas que não são processadas do lado do servidor são denominadas conteúdo estático.

[2]Páginas web que realizam processamento representam conteúdo dinâmico.

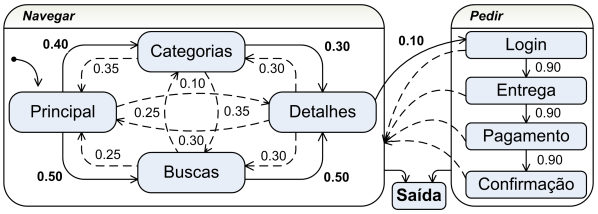

Figura 2: Máquina de estados proposta para modelar a carga de trabalho

As Tabelas 1 e 2 apresentam essas porcentagens extraídas da Figura 1 para diferentes tipos de interações de páginas.

Tabela 1: Porcentagens de acesso para interações do tipo *Navegar*

INTERAÇÕES DO TIPO NAVEGAR (90%)		
	Prob. Acesso	**Porcentagem**
Principal	0,9	23%
Detalhes	0,9	23%
Buscas	1,15	30%
Categorias	0,95	24%
Total	**3,9**	**100%**

Tabela 2: Porcentagens de acesso para interações do tipo *Pedir*

INTERAÇÕES DO TIPO PEDIR (10%)		
	Prob. Acesso	**Porcentagem**
Login	0,9	25%
Entrega	0,9	25%
Pagamento	0,9	25%
Confirmação	0,9	25%
Total	**3,6**	**100%**

Como a probabilidade de acesso para as interações do tipo *Pedir* tem peso $0,1$ e as interações do tipo *Navegar* peso $0,9$, as porcentagens para a carga de trabalho total foram recalculadas, gerando o comportamento de acesso de usuários apresentado nas Tabelas 3 e 4. Este trabalho utiliza esse comportamento de usuários para criar um modelo de carga de trabalho sintética com o objetivo de avaliar diferentes políticas de distribuição de requisições em *clusters* de servidores Web para um site de *e-commerce*.

Tabela 3: Probabilidades de acesso para interações de *Navegar* e *Pedir*

CATEGORIAS DE INTERAÇÃO		
	Prob. Acesso	**Porcentagem**
Navegar	0,9	90%
Pedir	0,1	10%
Total	**1**	**100%**

Tabela 4: Distribuição dos tipos de requisições da carga de trabalho.

CARGA DE TRABALHO	
Principal	21,00%
Detalhes	20,00%
Buscas	27,00%
Categorias	22,00%
Login	2,50%
Entrega	2,50%
Pagamento	2,50%
Confirmação	2,50%
Total	100,00%

3 Cenário Adotado

Com o modelo de carga de trabalho definido, decidiu-se por realizar experimentos e coletar tempos médios para cada categoria de página que recebe requisições Web[3]. Com esses experimentos foi obtido o comportamento de um sistema Web real para *e-commerce* e encontrados valores posteriormente utilizados na parametrização de um modelo de simulação com políticas de atendimento de requisições.

Para obtenção de valores médios de tempos de resposta para cada categoria de página que compõe o modelo de carga de trabalho definido, seria necessário analisar registros de servidores Web reais com tais informações. Devido a dificuldade de encontrar arquivos de registros do gênero, optou-se por reproduzir um cenário real. Esse cenário é composto por: 1) um software servidor Web; 2) uma aplicação de *e-commerce*; 3) e, computadores clientes em rede.

Foi utilizada nos experimentos a ferramenta de *e-commerce OSCommerce* – escrita na linguagem PHP e licenciada sob GNU/GPL – que utiliza o banco de dados MySQL e atende todos os requisitos de uma ferramenta de comércio eletrônico. O Apache foi adotado como servidor Web.

As interações relacionadas ao pedido de produtos são criptografadas por meio de conexões seguras usando SSL (*Secure Socket Layer*). Os clientes precisam se registrar no site antes de terem permissão de compra. Cada item à venda tem uma descrição textual e uma imagem miniatura associada com tamanho inferior a 5 KBytes.

Três computadores foram utilizados para realizar experimentos, um para gerar requisições Web (como cliente) por meio de um navegador, outro configurado como servidor Web (registrando tempos de atendimento) e um terceiro configurado como servidor de aplicações, responsável pelas operações de acesso a banco de dados.

[3]Essa é uma técnica para obter valores de ambientes reais e fazer com que a simulação resulte em informações precisas.

Todos os computadores conectados por uma rede Gigabit Ethernet e com processadores AMD Athlon XP 3.2 GHz de freqüência, 1 GB de memória RAM e disco rígido de 7200 rotações por minuto com capacidade de armazenamento de 80 GB.

4 Instrumentação para Experimentos

Servidores Web registram em arquivos texto (*logs*) eventos de sucesso e erros associados às requisições de clientes. Cada registro traz as seguintes informações relacionadas à requisição atendida: endereço IP do cliente, data e hora de atendimento, método pelo qual realizou a solicitação, nome do arquivo, protocolo de comunicação e sua respectiva versão, código de status da requisição, quantidades de bytes enviados, tempo total de atendimento e tempo de consumo de CPU.

Uma informação necessária para modelar um sistema Web é o tempo total de consumo de CPU para cada requisição. O servidor Web Apache utilizado neste trabalho disponibiliza apenas o tempo total de atendimento de cada requisição e não a quantidade de processamento (CPU) consumida. Isso motivou a instrumentação do Apache, adicionando uma nova função para coletar o consumo de CPU para requisições Web. Essa função foi implementada utilizando a chamada *clock()* do sistema operacional Linux.

Na estrutura de uma requisição do Apache foi adicionado um novo campo para armazenar o tempo inicial de CPU da requisição. Na última etapa de atendimento de uma requisição é feito o registro do atendimento em arquivo texto. Essa tarefa é de responsabilidade do módulo *mod_log_config*. Nesse módulo foi implementada uma função para coletar o tempo atual de consumo de CPU, que em seguida é subtraído do tempo inicial de CPU . Esse cálculo resulta no tempo total de consumo de CPU de cada requisição em *ticks* do relógio do sistema. Dividindo esse valor pela macro *CLOCKS_PER_SEC* obtém-se o número de segundos consumidos para processar uma requisição (independente da carga do processador, pois é medido o tempo de CPU realmente consumido).

Dados de atendimentos de requisições Web foram coletados durante intervalos de tempo e, posteriormente, utilizados para modelar o comportamento de um sistema Web composto por vários servidores. Uma vez obtido o tempo total de atendimento de uma requisição e o de consumo de CPU pode-se calcular o tempo consumido em operações de entrada e saída (I/O). Esse tempo ($T_{r,io}$) é obtido através da equação 1 onde: r representa a requisição, $T_{r,total}$ o tempo total de atendimento de uma requisição r, e $T_{r,cpu}$ o tempo total de consumo de CPU.

$$T_{r,io} = T_{r,total} - T_{r,cpu} \quad (1)$$

O tempo de entrada e saída inclui o tempo de acesso a disco local e acesso a banco de dados. Caso o banco esteja em outro computador, o tempo de entrada e saída inclui, também, atrasos de rede. Operações de acesso a base de dados são caracterizadas por exigirem maior demanda de dispositivos de I/O. A Tabela 5 apresenta o tempo consumido de CPU e I/O para diferentes tipos de requisições. Nota-se que requisições para páginas dinâmicas consomem mais recursos de CPU.

Tabela 5: Tempos de resposta obtidos COM cache no cliente

	Tempos de Resposta					
	Cliente com cache			**Cliente sem cache**		
Tipos	**Total**	**CPU**	**I/O**	**Total**	**CPU**	**I/O**
Principal	0,088	0,070	0,017	0,091	0,072	0,018
Detalhes	0,086	0,063	0,022	0,087	0,073	0,013
Buscas	0,115	0,094	0,021	0,121	0,096	0,024
Categorias	0,096	0,076	0,019	0,098	0,082	0,016
Login	0,046	0,041	0,005	0,076	0,041	0,035
Entrega	0,056	0,048	0,007	0,125	0,083	0,042
Pagamento	0,105	0,083	0,021	0,140	0,072	0,068
Confirmação	0,112	0,063	0,049	0,144	0,058	0,085

Páginas Web dinâmicas têm maior consumo de recursos de CPU e não são armazenadas em memória *cache* dos servidores. Operações de acesso a base de dados são caracterizadas por exigirem maior demanda de dispositivos de I/O.

Como observado na Tabela 5 , o tempo de consumo de CPU das requisições é maior que o tempo de consumo de dispositivos de I/O. Isso se deve não somente ao fato da base de dados estar presente em um servidor de aplicações (separado do servidor Web), mas também pela presença do recurso de memória *cache* do sistema operacional onde executa o servidor Web que armazena arquivos estáticos acessados com maior freqüência (imagens, folhas de estilos, etc). Dessa forma são resgatados da memória e não de um dispositivo de armazenamento, minimizando o tempo consumido em I/O.

Para cada arquivo (página Web do servidor), foram geradas requisições contínuas com intervalos de 1 segundo até o tempo de resposta convergir em torno de uma média segundo o teorema do limite central [10]. Utilizando essa técnica pode-se adotar medidas estatísticas de resumo (neste caso médias) para representar a ocupação de recurso, posteriormente utilizada na modelagem do problema. Cada requisição atendida no servidor Web gera uma linha no arquivo de registros com as informações: nome do arquivo, tempo total de atendimento, de consumo de CPU e de consumo de recursos de I/O.

Caso o cliente Web utilize *cache* local (opcional e configurável no navegador), objetos que compõem as

páginas Web podem ou não ser requisitados para o servidor, o que gera economia de recursos.

Em requisições da categoria *Navegar*, o ganho de desempenho *com* o uso de *cache* no cliente se torna desprezível, pois apresenta melhoria inferior a 5%. Por outro lado, quando se trata de interações do tipo *Pedir*, os efeitos de usar ou não o recurso de *cache* no cliente apresenta diferença significativa no tempo total de atendimento da requisição no servidor, ocasionada pelo aumento no tempo de I/O da requisição. Isso se deve ao fato de requisições dessa categoria serem criptografadas.

Sem o recurso de *cache* no cliente, a chave simétrica de criptografia entre cliente (navegador) e servidor Web deverá ser gerada pelo cliente a cada requisição. Dessa forma, o servidor executará novamente o procedimento de *handshaking* (troca de certificados, negociação de compressão e encriptação, configuração de identificador de sessão). Esse procedimento é realizado durante o atendimento de uma requisição e pode aumentar o tempo de I/O, conforme a largura de banda da rede que interliga o cliente e o servidor [2].

5 O Modelo

Teixeira [8] considera que servidores Web tratam requisições segundo uma disciplina *FIFO*, ou seja, é mantida uma única fila de espera onde cada requisição aguarda o momento de ser atendida, de acordo com sua ordem de chegada. Embora diferentes esquemas de controle de concorrência possam ser implementados com o objetivo de agilizar o atendimento de requisições, tal atendimento, em geral, não considera particularidades e a urgência diferentes tipos de requisição.

Esta seção descreve o modelo proposto neste trabalho para Servidor Web de E-Commerce (*SWE-C*) [5]. Esse modelo representa uma arquitetura de servidor Web cujo objetivo é atender requisições da categoria *Pedir* com prioridade superior sobre a categoria *Navegar*. É importante priorizar, pois a cada etapa que o visitante avança no processo de compra, maior a probabilidade de confirmação de pedido.

O modelo (Figura 3) segue as características apresentadas na Seção 2 e considera uma arquitetura com dois componentes: um escalonador e um *cluster* de Servidores Web. O Escalonador é responsável por admitir requisições e atribuí-las a servidores Web do *cluster*, segundo um algoritmo de balanceamento de carga. Cada servidor é modelado para mensurar o consumo de CPU e I/O, conforme valores extraídos da instrumentação do servidor Apache (Seção 4).

As requisições Web admitidas no sistema são inseridas na fila de CPU de um servidor selecionado pela

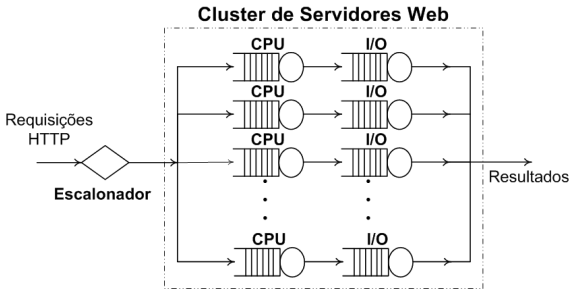

Figura 3: Modelo do Servidor Web para E-Commerce

algoritmo de balanceamento de carga e atendidas conforme a disciplina de fila vigente (este trabalho considera três disciplinas: *FIFO*, *LIFO* e *LIFO-Pri*) que são apresentadas na Seção 6.2. O tempo de atendimento de CPU é estabelecido segundo uma distribuição exponencial com médias baseadas na Tabela 5, considerando ou não o uso de cache local nos clientes.

Em seguida, a requisição passa por uma fila de I/O que tem seus tempos de atendimento baseados na Tabela 5. Após essas etapas resultados são retornados ao cliente.

6 Metodologia

A abordagem de avaliação do modelo foi baseada em simulação utilizando redes de filas [4]. O modelo do servidor *SWE-C* foi implementado para um ambiente com um *cluster* de oito servidores Web e um escalonador. Foram modeladas CPU e dispositivo de I/O de cada servidor Web. Assume-se que todos os servidores do cluster têm acesso aos mesmos arquivos (páginas Web)[4].

Uma carga de trabalho sintética foi gerada utilizando os dados da Tabela 4. Para tanto, foi desenvolvido um simulador que distribui o tráfego entre os tipos de requisições apresentadas na Seção 2. Antes mesmo de uma requisição chegar ao sistema, são definida duas informações: o tipo (Principal, Detalhes etc.) que define o tempo de ocupação de CPU e I/O e a taxa de acerto em *cache* no cliente.

Antes de processar cada requisição, o simulador gera um número aleatório que é comparado à probabilidade de acerto de cache. São adotados os seguintes valores para acertos: 30% para as páginas Principal, Detalhes, Buscas, Categorias, 0% para Login e 100% para Entrega, Pagamento e Confirmação.

A taxa de acerto em *cache* para requisições da categoria *Navegar* é fixada em 30% [7]. Requisições da

[4]Pode-se adotar um sistema de arquivos distribuídos para garantir acesso às páginas Web tal como NFS – *Network File System*

categoria *Pedir* têm taxas de acerto inferiores a requisições da categoria *Navegar*. Isso se deve ao fato de requisições dessa categoria serem criptografadas por meio de conexões SSL e apresentarem particularidades em relação ao uso do recurso de memória *cache* no cliente.

Neste trabalho considera-se que o procedimento de *handshaking* (troca de certificados, negociação de compressão e encriptação e configuração de identificador de sessão) é executado apenas em requisições do tipo Login e que a chave gerada permanece armazenada no *cache* local do cliente. Isso significa que, para requisições do tipo Login são utilizados os tempo de consumo de CPU e I/O sem cache da Tabela 5, uma vez que não existe probabilidade do cliente ter uma chave de sessão válida que o permite utilizar objetos embutidos armazenados em *cache* local.

Tendo em vista que as requisições dos tipos: Entrega, Pagamento e Confirmação não repetirão o procedimento de *handshaking* (uma vez que considera-se que o cliente tem uma chave de sessão em *cache* local), elas utilizam tempos de consumo de CPU e I/O com cache da Tabela 5, pois não repetirão o procedimento. Assume-se uma taxa de acerto em *cache* de 100% das requisições de tipos supra citadas.

6.1 Algoritmos de Balanceamento de Carga

Dois algoritmos de balanceamento de carga foram utilizados para avaliar o modelo proposto. Esses algoritmos são executados pelo módulo escalonador e a carga (requisições) distribuída sobre computadores que compõem o *cluster*. Os seguintes algoritmos foram considerados devido à experiência anterior do grupo: *Round-Robin* (*RR*) que aloca requisições aos servidores segundo uma lista circular; *SQF* que aloca requisições para o servidor com o menor número de processos em fila. Neste trabalho é considerado o número de processos em fila de CPU[5].

6.2 Disciplinas para Fila de Atendimento de Requisições

As requisições que chegam aos servidores Web são adicionadas em filas. A ordem de atendimento de processos em filas seguem disciplinas específicas. Atualmente, servidores Web utilizam a disciplina *FIFO*, atendendo requisições segundo a ordem de chegada. Visando aumentar o desempenho de sistemas Web, outras disciplinas de fila foram avaliadas: *FIFO* (*First In First Out*) onde requisições são atendidas na ordem de chegada; *LIFO* (*Last In First Out*) onde a última requisição a chegar é a primeira a ser atendida; *LIFO-Pri* (*Last In First Out with Priority*) na qual as últimas requisições a chegar são avaliadas e atendidas segundo suas respectivas prioridades.

Embora grande parte dos visitantes não tenha intenção de comprar, é importante ressaltar que quando exercerem tal intenção e iniciarem o processo de compra devem ser orientados a completar a transação sem estourar um tempo limite (normalmente estabelecido por medidas de segurança). Isto é especialmente importante durante condições de sobrecarga, quando os servidores Web apresentam aumento no tempo de atendimento. Caso o tempo de resposta do sistema ultrapasse o tempo limite para completar uma transação, nenhuma transação poderá ser concluída com sucesso até que o tráfego do sistema diminua.

No processo de compra, as requisições geradas pelos clientes pertencem a categoria *Pedir*. Dessa forma, optou-se por atribuir maiores prioridades para as requisições dessa categoria (Login, Local de Entrega, Forma de Pagamento e Confirmação) em relação às da categoria *Navegar* (Principal, Detalhes, Buscas e Categorias). A utilização de prioridades foi adotada nas filas de CPU de cada *host* servidor Web do *cluster*.

6.3 Critérios Adotados

Para a definir quais tipos de requisições devem ter maior prioridade sobre as demais, faz-se necessário adotar critérios que justifiquem tais prioridades. Dois critérios foram propostos para definir prioridades na diferenciação do atendimento de requisições: um orientado ao negócio e outro ao consumo de recursos do sistema.

Em [6] Singhmar propõe um mecanismo de prioridades orientado ao negócio. As prioridades foram estabelecidas segundo a relevância que cada tipo de requisição representa na lógica do negócio.

A atribuição de prioridades adotada neste trabalho é orientada ao consumo de recursos, especificamente CPU, e se faz presente na implementação da disciplina *LIFO-Pri*. As seguintes prioridades foram definidas: Principal (3), Detalhes (4), Buscas (1), Categorias (2), Login (5), Entrega (6), Pagamento (7) e Confirmação (8). Assumiu-se que qualquer tipo de requisição da categoria *Pedir* tem maior prioridade do que as requisições da categoria *Navegar*, pois tendem a gerar lucro para o negócio.

7 Resultados Experimentais

Foi implementado um software para simular o atendimento de requisições Web usando o modelo *SWE-C* sob diferentes intensidades de carga por um período de uma

[5] A manutenção de informações sobre a fila pode gerar sobrecarga no sistema e prejudicar o balanceamento de carga [8]

hora (3.600 segundos). Combinações de algoritmos de escalonamento e disciplinas para filas de atendimento foram aplicadas respectivamente ao componente escalonador e às filas de CPU de cada *host* servidor Web do *cluster*. Os resultados apresentados foram obtidos executando experimentos até obter um desvio padrão inferior a 5% em torno da média [10][6].

Para Vallamsetty [9], a baixa utilização de recursos é típica em sites de *e-commerce*. Sistemas são projetados para sustentarem uma carga baixa durante a maior parte do tempo e resistirem a períodos de sobrecarga com desempenho aceitável.

Tabela 6: Utilização média de CPU para diferentes intensidades de carga.

Utilização Média de CPU		
Taxa de Chegada	**Intensidade**	**Utilização**
70 req/seg	Baixa	70%
100 req/seg	Média	98%
105 req/seg	Alta	100%

Optou-se por submeter o sistema a três diferentes intensidades de carga: baixa, média e alta, afim de representar situações reais as quais sites de e-commerce são submetidos. Observando gráficos de utilização média de CPU alcançada nos *hosts* servidores Web do cluster ao longo de uma hora de tráfego, pode-se comprovar os dados da Tabela 6.

Medidas de desempenho foram utilizadas para modelar e avaliar o desempenho de algoritmos de balanceamento de carga e disciplinas de fila, dentre as quais se destacam o tempo médio de resposta e *throughput* de requisições.

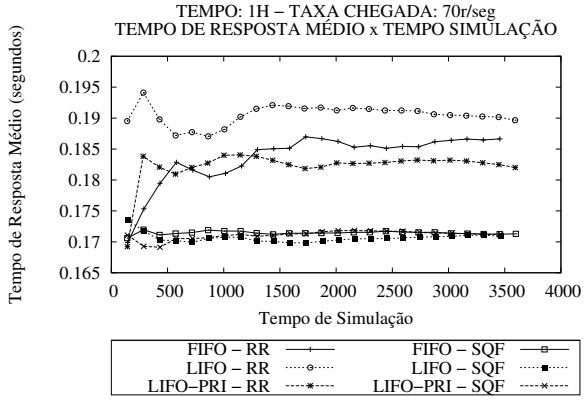

Figura 4: Tempo médio de resposta (intensidade de carga baixa – 70 req/seg).

[6]Essa técnica segue o conceito do teorema do limite central, permitindo a aplicação confiável de medidas estatísticas de resumo.

Inicialmente, o sistema foi submetido a uma carga de tráfego de baixa intensidade, afim de analisar se o uso da disciplina *LIFO-Pri* é ineficiente em períodos de não sobrecarga do sistema, quando comparado às disciplinas *FIFO* e *LIFO* (que não utilizam mecanismos de prioridades). A Figura 4 ilustra o tempo médio de resposta do sistema quando submetido a uma carga de baixa intensidade (70 req/seg).

Nota-se que a utilização da combinação *SQF/LIFO-Pri* apresenta um ganho aproximado de $4,1\%$ e $2,5\%$ no tempo médio de resposta do sistema em relação às combinações *SQF/FIFO* e *SQF/LIFO* respectivamente.

Por outro lado, a utilização da combinação *RR/LIFO-Pri* apresenta apenas um desempenho equivalente às combinações *RR/FIFO* e *RR/LIFO*. Comparando as combinações *SQF/LIFO-Pri* e *RR/LIFO-Pri*, constata-se que a primeira disciplina apresenta um ganho de $6,24\%$ no tempo médio de resposta do sistema.

Conclui-se que, ao aplicar o mecanismo de prioridades proposto juntamente com o algoritmo de balanceamento de carga *SQF*, pode-se obter melhores tempos de resposta em situações de baixa intensidade de carga. Observa-se também que a porcentagem de requisições atendidas em relação a quantidade de requisições que chegam é igual ou superior a $99,9\%$. Isso faz com que a diferença de *throughput* entre as diversas combinações torne-se desprezível.

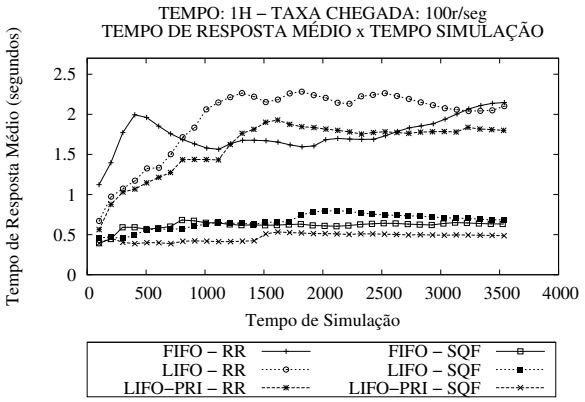

Figura 5: Tempo médio de resposta (intensidade de carga média – 100 req/seg)

Com o objetivo de analisar o comportamento do sistema em situações de alto tráfego, sem ultrapassar a capacidade total de atendimento, o sistema foi submetido a uma intensidade de carga média. A taxa de utilização média de CPU do sistema ficou em torno de 98%. Com uma taxa de chegada de aproximadamente 100 requisições por segundo, a porcentagem no atendimento de requisições com sucesso em relação a quantidade de

requisições que chegam é igual ou superior a 99%.

Embora o tempo de resposta médio geral combinando *RR/LIFO-Pri* ($1,797917$ seg.) não retrate tão bem os tempos médios individuais de cada tipo de requisição quanto das combinações *RR/FIFO* ($2,170538$ seg.) e *RR/LIFO* ($2,101788$ seg.), observa-se ainda aumento de desempenho aproximado de 17% utilizando *RR/LIFO-Pri* em relação a *RR/FIFO* e *RR/LIFO*. Na combinação *RR/LIFO-Pri*, nota-se que o tempo médio de resposta para as requisições de menor prioridade (*Buscas* - $5,985908$ seg.) está muito acima dos tempos médios de resposta dos demais tipos de requisições, apresentando influência direta no tempo de resposta médio.

Pode-se observar que o tempo médio de resposta para as requisições de menor prioridade (*Buscas*) está muito acima dos tempos médios de resposta dos demais tipos de requisições, apresentando influência diretamente no tempo de resposta médio geral. Embora o tempo de resposta médio geral da combinação *RR/LIFO-Pri* ($1,797917$ seg.) não retrate tão bem os tempos médios individuais de cada tipo de requisição quanto das combinações *RR/FIFO* ($2,170538$ seg.) e *RR/LIFO* ($2,101788$ seg.), observa-se ainda um ganho de desempenho aproximado de 17% da combinação *RR/LIFO-Pri* em relação a *RR/FIFO* e *RR/LIFO*.

A combinação *SQF/LIFO-Pri* (com tempo médio geral de resposta de $0,483794$ seg.) apresenta melhor tempo de resposta médio geral do sistema de aproximadamente 30% em relação às combinações *SQF/FIFO* ($0,674924$ seg.) e *SQF/LIFO* ($0,685020$ seg.). Além disso, observa-se que durante os 3.600 segundos de tráfego submetidos ao sistema, o aproveitamento na quantidade de requisições atendidas em relação a quantidade de requisições que chegaram para *SQF/LIFO-Pri* é de $99,99\%$, superior a combinação *SQF/FIFO* com $99,93\%$ e a combinação *SQF/LIFO* com $99,95\%$.

Na Figura 5, observa-se uma melhoria expressiva na média geral dos tempos de resposta do sistema com a utilização das combinações baseadas em *SQF* em relação as combinações baseadas em *RR*. Constata-se que, sob intensidade de carga média, o sistema também apresentou comportamento semelhante a uma intensidade de carga baixa.

Finalmente o sistema fora submetido a uma intensidade de carga total, cuja taxa de utilização média de CPU atingiu 100%. Nesse foi possível avaliar a eficiência do mecanismo de prioridades proposto para o atendimento de requisições Web em sites de *e-commerce*.

Em sobrecarga, nota-se que a disciplina *FIFO* combinada tanto com o algoritmo *SQF* quanto com o algoritmo *RR* se mostram ineficientes, com tempos médios de resposta de $46,077846$ e $52,784698$ segundos, respectivamente. A seguir são analisados apenas os resultados das combinações das disciplinas *LIFO* e *LIFO-Pri* com os algoritmos *RR* e *SQF*.

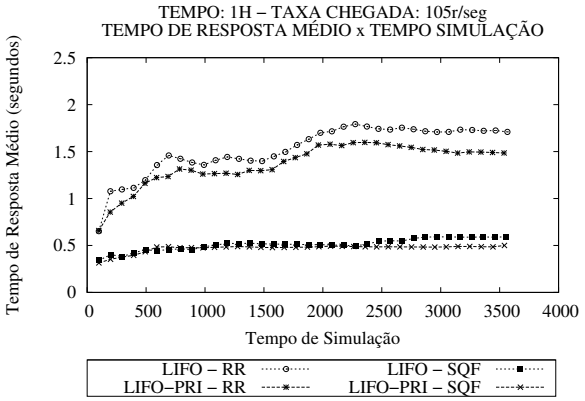

Figura 6: Tempo médio de resposta (intensidade de carga alta – 105 req/seg).

Tabela 7: Combinação *RR/LIFO-Pri* (intensidade de carga alta - 105 req/seg).

Tipo	Cheg.	Atend.	Tempo(seg)	Pri
Principal	21,04%	99,99%	0,25	3
Detalhes	19,98%	100,00%	0,20	4
Buscas	**26,90%**	**92,64%**	**4,99**	**1**
Categorias	21,99%	99,97%	0,41	2
Login	2,02%	100,00%	0,17	5
Entrega	3,02%	99,99%	0,16	6
Pagamento	2,00%	100,00%	0,18	7
Confirmação	3,05%	99,99%	0,20	8
Total	100%	98,01%		
Média			1,47	

Analisando os tempos médios de resposta do sistema por tipo de requisição para a combinação *RR/LIFO*, nota-se que os mesmos estão próximos da média ($1,703390$ seg.). Porém, observa-se que a porcentagem de requisição atendidas apresenta valores próximos a 97% em todos os tipos de requisições. Parte das requisições que chegaram ao sistema não foram atendidas no intervalo de uma hora e entre essas se encontram requisições da categoria *Pedir*. Essa combinação de algoritmo/disciplina poderia trazer prejuízos, uma vez que esta deixa de atender requisições com alta probabilidade de geração de renda.

Os resultados da combinação *RR/LIFO-Pri* são apresentados na Tabela 7. Como observado anteriormente, a disciplina *LIFO-Pri* tem a característica de aumentar o tempo médio de resposta das requisições de menor prioridade (*Buscas*). Dessa vez, nota-se que a porcenta-

gem de requisições atendidas do tipo *Busca* é inferior a porcentagem das demais categorias, o que pode ocasionar uma situação de *starvation*, ou seja, em algum momento de carga alta a porcentagem de atendimento de requisições de menor prioridade será nula. Deve-se ressaltar que a porcentagem de atendimento de requisições da categoria *Pedir* alcança sua totalidade, indicando fortes indícios de que a combinação *RR/LIFO-Pri* se aplica em sites de *e-commerce*.

Comparando o tempo médio geral de resposta de *RR/LIFO-Pri* com *RR/LIFO*, observa-se um ganho de desempenho de $15,3\%$. Se excluídos os tempos médios das requisições de *Buscas*, observa-se um ganho de desempenho aproximado de 650% no tempo médio de resposta. Porém, constata-se uma perda considerável no desempenho do tempo médio de atendimento de requisições de *Buscas*, cuja proporção chega a 190% do tempo médio da combinação *RR/LIFO*.

Tabela 8: Combinação *SQF/LIFO-Pri* (intensidade de carga alta - 105 req/seg).

Tipo	Cheg.	Atend.	Tempo(seg)	Pri
Principal	21,01%	99,99%	0,21	3
Detalhes	20,10%	99,99%	0,19	4
Buscas	**26,97%**	**92,85%**	**1,40**	**1**
Categorias	21,97%	99,99%	0,29	2
Login	2,02%	99,99%	0,17	5
Entrega	2,98%	100,00%	0,16	6
Pagamento	2,00%	100,00%	0,18	7
Confirmação	2,96%	99,99%	0,20	8
Total	100%	98,06%		
Média			0,52	

Para a combinação *SQF/LIFO*, observa-se que a porcentagem de requisição atendidas apresenta valores em um patamar próximo a 97% para todos os tipos de requisições, incluindo requisições da categoria *Pedir*. Utilizando a combinação *SQF/LIFO-Pri* (Tabela 8), todas as requisições à páginas da categoria *Pedir* foram atendidas com sucesso e que as requisições de *Buscas* tiveram o tempo médio de resposta e a porcentagem no aproveitamento comprometidos, em relação a quantidade de requisições que chegaram. Se comparados o tempo médio de resposta da combinação *SQF/LIFO-Pri* com o da *SQF/LIFO*, observa-se um ganho de desempenho de $12,2\%$. Se excluídos os tempos médios das requisições de *Buscas*, observa-se um ganho de desempenho aproximado de 196% no tempo médio de resposta. Porém, constata-se uma perda considerável no desempenho do tempo médio de atendimento de requisições de *Buscas*, cuja proporção chega a 120% do tempo médio da combinação *RR/LIFO*.

Utilizando o tempo médio de resposta do sistema como medida de desempenho, observa-se na Figura 6 que as combinações baseadas em *SQF* apresentam tempos médios de resposta menores em relação as combinações baseadas em *RR*. Considerando que a média dos tempos de resposta de *RR/LIFO* e *RR/LIFO-Pri* é $1,590199$ segundos e que a média dos tempos das combinações *SQF/LIFO* e *SQF/LIFO-Pri* é $0,560684$ segundos, constata-se um ganho de desempenho em combinações baseadas em *SQF* de 183% em relação as combinações baseadas em *RR*.

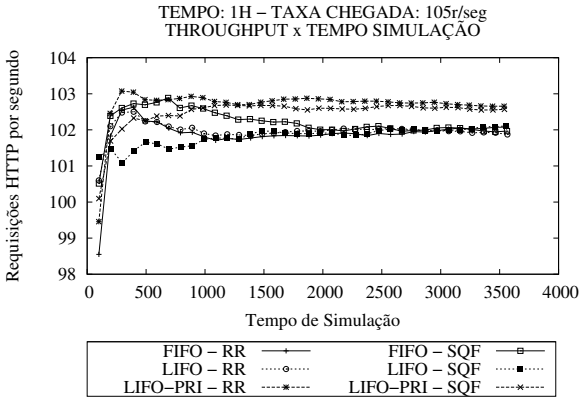

Figura 7: Throughput de requisições (intensidade de carga alta - 105 req/seg).

A Figura 7 apresenta o quantidade de requisições atendidas com sucesso pelo sistema. Nota-se que, exceto para *SQF/FIFO*, os valores estabilizam após 1.000 segundos e se mantém praticamente constantes até o final da simulação. Os valores da combinação *SQF/FIFO* diminuem consideravelmente entre 800 e 2.000 segundos, se estabilizam na seqüência até o término do experimento. Ao final de 3.600 segundos, observa-se que a quantidade média de requisições atendidas por segundo na combinação *RR/LIFO-Pri* é a maior, com $102,663$ req/seg, seguida de *SQF/LIFO-Pri* com $102,587$ req/seg. Em um patamar um pouco abaixo se encontram as combinações *SQF/LIFO* e *RR/FIFO* praticamente empatadas com $102,088$ req/seg e em seguida as combinações *SQF/FIFO* e *RR/LIFO* com $101,927$ e $101,868$ respectivamente.

Para observar o aumento no *throughput* do sistema ao longo do tempo, ele foi submetido a uma intensidade de carga muito alta, com uma taxa de chegada de 110 req/seg. A não utilização do mecanismo de prioridades proposto, independentemente do algoritmo de escalonamento adotado, mantem o *throughput* do sistema num patamar próximo de 102 req/seg. Quando utilizado o mecanismo de prioridades orientado ao consumo de , observa-se um aumento no *throughput*, que

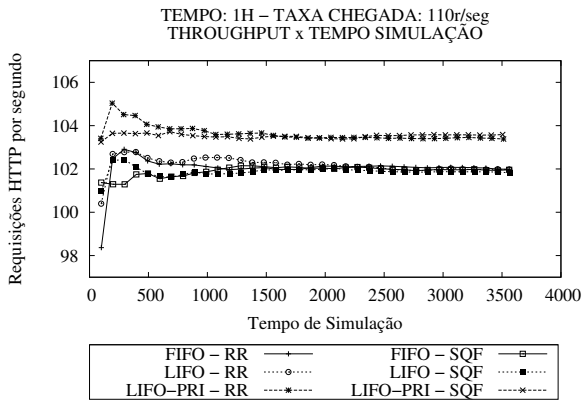

Figura 8: Throughput de requisições (intensidade de carga extremamente alta - 110 req/seg).

atinge o patamar de 103 req/seg. Essa diferença na vazão, se analisada no intervalo de 1 segundo, aparenta ser mínima. Porém, quando analisada ao longo do tempo, ou seja, até o final de uma hora de tráfego (3.600 segundos), torna-se importante. Com o mecanismo proposto, ao final de uma hora, obtem-se um aumento de 3600 requisições atendidas com sucesso. Tendo em vista que sites de *e-commerce* diariamente se deparam com situações semelhantes de tráfego, constata-se que, ao longo do período de um mês, o aumento na quantidade de requisições atendidas seria extremamente alto e que um aumento na renda do site cresceria na mesma proporção.

8 Conclusões

Neste artigo é proposto um mecanismo de prioridades orientado ao consumo de CPU para requisições Web. Ele foi avaliado com um modelo de carga de trabalho para sites de *e-commerce*.

Conclui-se que a utilização do mecanismo de prioridades proposto em filas de CPU dos *hosts* servidores Web do *cluster* é mais adequada para atender as necessidades de um site de *e-commerce*, pois apresenta melhor aproveitamento no atendimento de requisições da categoria *Pedir*. Essa técnica gera menores tempos médios de resposta e maior vazão no atendimento de requisições ao longo do tempo.

Referências

[1] CyberAtlas. Cyberatlas: Internet statistics and market research for web marketers, September 2002.

[2] Goldberg, A., Buff, R., and Schmitt, A. Secure web server performance dramatically improved by caching ssl session keys. *Workshop on Internet Server Performance*, Jun. 1998.

[3] Hirata, R. Otimizando servidores web de alta demanda. Master's thesis, Instituto de Computação– Universidade Estadual de Campinas, Abril 2002.

[4] Jain, R. *The Art of Computer Systems Performance Analysis: Techniques for Experimental Design, Measurements, Simulation and Modeling*. John Wiley & Sons, 1991.

[5] Peres, C. Avaliação de desempenho com algoritmos de escalonamento em clusters de servidores web. Master's thesis, Universidade de São Paulo – Instituto de Ciências Matemáticas e de Computação, Junho 2006.

[6] Singhmar, N., Mathur, V., Apte, V., and D.Manjunath. A combined lifo-priority scheme for overload control of e-commerce web servers. In *International Infrastructure Survivability Workshop (IISW'2004)*. IEEE Computer Society, 2004.

[7] SPECweb2005. Specweb2005 e-commerce workload design document, 2005.

[8] Texeira, M. A. M. *Suporte a Serviços Diferenciados em Servidores Web: Modelos e Algoritmos*. PhD thesis, ICMC-USP, São Carlos, maio 2004.

[9] Vallamsetty, U., Kant, K., and Mohapatra, P. Characterization of e-commerce traffic. *Eletronic Commerce Research*, pages 167–192, Mar. 2003.

[10] W.C.Shefler. *Statistics: Concepts and Applications*. The Benjamin/Cummings, 1988.

[11] Zhuo, L., Wang, C.-L., and Lau, F. C. Document replication and distribution in extensible geographically distributed web server, 2002.