Balanceamento de Carga em Servidores de Anúncios Eletrônicos Distribuídos

Rodrigo Pereira Bruno Diniz Flávia Ribeiro
Wagner Meira Jr. Virgílio Almeida

e-SPEED - Departamento de Ciência da Computação
Universidade Federal de Minas Gerais
Belo Horizonte — Brasil
{rpereira,diniz,flavia,meira,virgilio}@dcc.ufmg.br

Abril 2001

Abstract

An essential feature to enable the success of services provided by Internet is its scalability, that is, the ability of the servers to support the utilization growth rate of this services. Distributing the service is a strategy of success to increase scalability. An example of Internet service, where the need of scalability is considerable, is in eletronic advertisement (*banners*). In this paper, we discuss the distribution of eletronic advertisement services and some load balance approaches in a distributed implementation, that are evaluated using a simulation of the proposed architecture and a workload of a real eletronic advertisement server.

Resumo

Uma característica fundamental para o sucesso de serviços providos pela Internet é a sua escalabilidade, ou seja, a capacidade dos servidores em suportar as taxas de crescimento de utilização desses serviços. Uma estratégia de sucesso para aumentar a escalabilidade de serviços é a sua distribuição. Um exemplo de serviço Internet, onde a necessidade de escalabilidade é grande, é o de publicidade eletrônica (*banner*). Neste artigo discutimos a distribuição de serviços de publicidade eletrônica e mecanismos para balanceamento de carga em uma implementação distribuída, os quais são avaliados utilizando uma simulação da arquitetura proposta e uma carga de trabalho de um servidor de anúncios eletrônicos real.

Keywords: banner, sistemas distribuídos, balanceamento de carga, análise de desmpenho, Internet.

1 Introdução

O crescimento da Internet pode ser verificado sob várias dimensões, além do número de usuários e máquinas conectadas à rede. As possibilidades de interação oferecidas pela

rede mundial de computadores são múltiplas e aumentam a cada dia. Um exemplo de aplicação que tem crescido muito e se tornado uma fonte real de recursos é o comércio eletrônico, que não apenas possibilitou a empresas tradicionais utilizarem a Web como meio de comercialização, como ainda permitiu modelos de interação antes impossíveis, como a a comercialização de produtos virtuais (p.ex., software) e aplicações como leilões não-presenciais.

Um aspecto fundamental para o sucesso das aplicações de comércio eletrônico é uma maior exigência em termos da qualidade de serviço que é oferecida aos clientes. Essa qualidade de serviço compreende não apenas a correção das transações executadas, a robustez e segurança dos servidores e dos dados que eles manipulam, como também a rapidez em atender às requisições submetidas por clientes. Entretanto, atingir esses níveis de qualidade de serviço não é uma tarefa trivial, tendo em vista a diversidade das aplicações, dos interesses dos clientes e da sua taxa de crescimento. Em particular, o problema da escalabilidade de servidores de comércio eletrônico, ou seja, a propriedade do servidor de comércio eletrônico conseguir suportar a demanda imposta pelos clientes, tem sido muito discutido tanto no meio acadêmico quanto no meio comercial, em razão do desafio tecnológico que representa e o impacto comercial que pode causar, como constatado em diversos serviços que tiveram problemas de escalabilidade.

Uma estratégia frequentemente utilizada para melhorar a escalabilidade de servidores de comércio eletrônico e servidores WWW em geral é a distribuição de serviços. Um exemplo de sucesso de distribuição de serviços na WWW são os proxy caches [13], que normalmente são utilizados para replicar conteúdo estático, cuja utilização reduziu significativamente o tráfego originado por requisições HTTP, principalmente em canais internacionais. Recentemente têm sido muito estudadas as redes de distribuição de conteúdo [2] como uma solução mais aprimorada para a distribuição de servidores de dados. Por outro lado, temos várias aplicações cujo conteúdo disponibilizado é gerado dinamicamente e que não pode ser replicado utilizando técnicas tradicionais. Um exemplo de tais trabalhos são o *active caching*, que permite a cache proxies executar programas Java localmente, reduzindo a carga imposta ao servidor original [5]. Outro exemplo são os *cache plugins*, que executam o processamento de máquinas de busca também nos proxy caches [14]. Embora sejam soluções efetivas para distribuir as aplicações às quais se destinam, o seu nível de especificidade limita o seu escopo de utilização. Finalmente, uma arquitetura distribuída muito difundida atualmente é a adotada pelo Napster [11]. Nessa arquitetura, chamada de ponto-a-ponto (*peer-to-peer*), os clientes também se comportam como servidores e a informação (no caso específico do Napster, arquivos no formato MP3) está espalhada entre eles e é transmitida através de conexões estabelecidas entre os próprios clientes.

No contexto de aplicações de comércio eletrônico, a necessidade da manutenção de propriedades transacionais e a existência de dados com alta volatilidade, como posições de estoque de produtos e valores correntes de produtos em leilão, torna essa distribuição mais complicada. Ela tem sido estudada sob várias perspectivas, seja através de protocolos específicos [12], ou abordagens baseadas em bancos de dados distribuídos.

Um exemplo de aplicação de comércio eletrônico que é inerente à Web é a comercialização de anúncios eletrônicos em páginas WWW. Neste cenário, há basicamente três entidades envolvidas: anunciantes, locatários e clientes. A comercialicação de anúncios consiste em alugar espaço em páginas WWW cedido pelos locatários para anunciantes,

de tal forma que clientes recebam o anúncio quando acessarem as referidas páginas. Obviamente que esse aluguel de espaço poderia ser contratado diretamente entre anunciante e locatário, mas ambas as partes, por questões de eficiência e praticidade normalmente iteragem através de um servidor de anúncios eletrônicos. Essa atividade de intermediação permitiu aperfeiçoar o serviço de publicidade eletrônica. Um primeiro aperfeiçoamento é a associação de anúncios a conceitos, ou seja, o anunciante pode requisitar, para fins de maior retorno do anúncio, que o mesmo seja veiculado para um determinado perfil de clientes, que pode, nos casos mais simples, ser determinado pelo site que o cliente está acessando. Mais ainda, anunciantes podem especificar não apenas a quantidade de anúncios a serem veiculados, como também a sua dimensão e posicionamento relativo na página onde são exibidos.

Podemos dividir a comercialização de um anúncio em duas fases: contratação e provimento. Durante a contratação de anúncios, anunciantes e o servidor de anúncios definem a quantidade, a natureza e o público alvo dos anúncios, e o servidor se compromete a exibí-los. Uma vez que o servidor de anúncios tenha uma base de anúncios contratados, ele provê o serviço de anúncios propriamente dito, que pode ser dividido em três passos. O primeiro passo é a conceituação, onde características como o site de origem ou o perfil do cliente são associados a conceitos definidos anteriormente. A seguir, é feita a seleção do anúncio, quando se define qual o anúncio será exibido, e a referência ao mesmo é incluída na página carregada pelo cliente. Nessa fase são utilizados algoritmos de seleção com nível variado de sofisticação, que é função do conjunto de parâmetros da seleção (tamanho do anúncio, palavras-chave, etc). Finalmente, o cliente (normalmente utilizando um navegador) requisita o anúncio propriamente dito, em geral uma imagem, utilizando o protocolo HTTP padrão. A execução dessas tarefas por parte de um servidor e as formas de relacionamento entre clientes, locatários e servidores de anúncio já foi objeto de investigação anterior [8], assim como a utilização de arquiteturas multiprocessadas para fins de manutenção de qualidade e diferenciação de serviços (QoS) [9].

Neste artigo discutimos e avaliamos, através de simulação, uma arquitetura distribuída de servidores de anúncios eletrônicos. Esta seção serve como introdução do artigo. Na próxima seção apresentamos conceitos básicos das arquiteturas de servidores de anúncios e descrevemos as funcionalidades básicas que são providas por esses servidores. A Seção 3 discute a estratégia que adotamos para distribuição de serviços de anúncio eletrônico. Avaliamos essa estratégia de distribuição de serviços e balanceamento de carga entre servidores através de simulação, utilizando logs de acesso reais de um grande servidor cache. A caracterização desses logs é apresentada na Seção 4, e a arquitetura do simulador e os resultados são apresentados na Seção 5. A última seção apresenta as conclusões e trabalhos futuros.

2 Arquitetura de servidores de anúncios

Descreveremos nessa seção os componentes envolvidos na interação entre um cliente e um servidor de anúncios. Também será mostrado o fluxo de requisições quando um cliente requisita um anúncio. O modelo utilizado para a descrição foi retirado de um trabalho anterior [8] onde definimos alguns possíveis modelos para o provimento de anúncios.

Como discutido na Seção 1, o principal objetivo de um servidor de anúncios é prover condições para um anuciante selecionar e expor um anúncio de forma eficiente e inteligente, sempre que possível. Nesse contexto, dois pontos essenciais devem ser observados e levados em conta quando do provimento desse tipo de serviço:

- **Algoritmos de seleção**
  É altamente desejável que um servidor de anúncios implemente um algoritmo capaz de selecionar o anúncio mais apropriado dentro de um conjunto de anúncios possíveis. A escolha de tal anúncio deve ser baseada em critérios como adequação semântica ou cumprimento de metas contratadas. Esses algoritmos podem ser bastante sofisticados, considerando até o período do dia em que a requisição está sendo feita ou mesmo o número IP da máquina do cliente que requisitou o anúncio.

- **Desempenho**
  Apesar de algoritmos sofisticados serem desejáveis, nenhum cliente está disposto a esperar mais para obter anúncios mais elaborados para o seu perfil. Vale lembrar que uma página na Web é composta por uma parte textual e por outros componentes diversos (inclusive imagens) que são carregados do servidor após a parte textual [1]. Logo, a carga completa de determinada página Web somente terminará quando todas as imagens forem carregadas. Isso serve de motivação para que o servidor de anúncios tenha um desempenho aceitável, que não atrapalhe a veiculação da própria página.

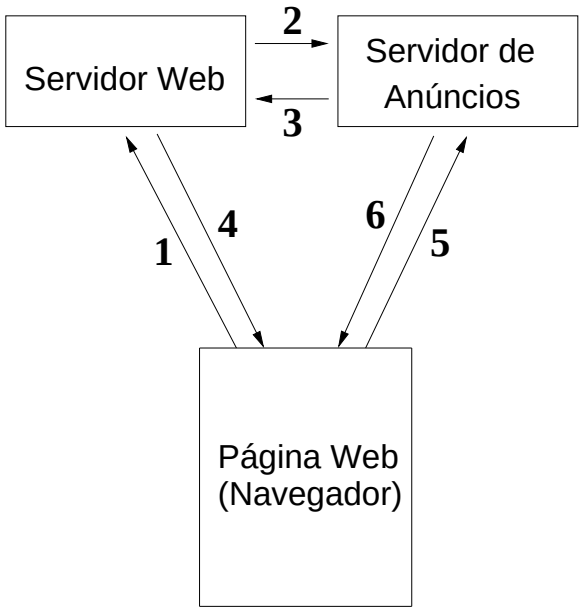

Figura 1: Fluxo de requesições em uma página Web que utiliza um servidor de anúncios

Na Figura 1 mostramos a sequência de seis eventos que ocorrem quando uma página é requisitada a algum servidor Web que esteja utilizando um servidor de anúncios para selecionar e veicular os anúncios que são incluídos nas páginas:

[1] No caso do protocolo HTTP versão 1.1, a mesma conexão é aproveitada para carregar os demais componentes de uma página. O que foi dito também se aplica nesse contexto pois a conexão somente será fechada quando o último componente for carregado, o que atrasa todo o processo de carga da página.

1) A página é requisitada ao servidor Web.

2) O servidor Web então verifica que a página sendo requisitada possui alguns espaços onde devem ser inseridos anúncios. Nesse momento, uma requisição é enviada ao servidor de anúncios para que este retorne os códigos dos anúncios que são mais indicados para preencher os espaços na página em questão.

3) Os códigos dos anúncios a serem expostos são enviados para o servidor Web, que então monta a página a ser enviada ao cliente com a URL de acesso aos anúncios.

4) A página montada é enviada ao cliente.

5) Como o navegador geralmente utilizado pelo cliente carrega as imagens separadamente, quando ele encontra no código HTML a URL que representa a imagem, ele abre uma conexão com o servidor de anúncios e requisita o anúncio a ser posicionado.

6) O anúncio é enviado pelo servidor de anúncios e o número de exposições do anúncio é atualizado, para garantir a correção do algoritmo de seleção.

Com base nessa descrição podemos identificar duas funcionalidades básicas a serem providas por servidores de anúncio: *seleção* e *provimento*.

A seleção consiste em selecionar os anúncios a serem incluídos na página a ser carregada pelo cliente. O algoritmo de seleção determina, entre todos os anúncios sendo gerenciados pelo servidor, o anúncio mais adequado.

A seleção é baseada em algumas características dos anúncios como o tamanho, o assunto do qual o anúncio trata, a categoria (negócios, educação, etc) na qual o anúncio está inserido, entre outras. Outras variáveis também podem ser consideradas como o momento em que a requisição foi feita, o IP da máquina do cliente que fez a requisição ou mesmo o assunto tratado pela página onde os anúncios retornados pela requisição serão expostos. Uma forma de simplificar a tarefa de seleção é empregando conceitos abstratos que caracterizam tanto a natureza dos clientes e suas requisições quanto os anúncios. Desta forma, a tarefa dos seletores é associar conceitos abstratos a requisições de clientes baseando-se nas suas características. Os anúncios, por sua vez, também são associados a conceitos por ocasião da sua contratação. Neste contexto, a tarefa de seleção se resume a determinar qual o par de conceitos mais similares e assinalar o anúncio correspondente à requisição.

O provimento é o envio do anúncio propriamente dito ao cliente. Essa tarefa é funcionalmente semelhante à executada por servidores WWW tradicionais, pois consiste apenas em ler o arquivo do anúncio do disco e enviá-lo ao cliente. Entretanto, há uma diferença importante que é a necessidade do registro do provimento do anúncio, que é a confirmação de que o cliente efetivamente vai visualizá-lo. Essa informação é importante para garantir que os servidores de anúncio cumprem as suas metas contratadas e é também utilizada pelo algoritmo de seleção.

Na próxima seção discutimos a distribuição desses servidores de anúncios eletrônicos, que é baseada na divisão de funcionalidades e sua replicação, criando os seletores e provedores.

3 Servidores de anúncios distribuídos

Como discutido, a escalabilidade de servidores de publicidade eletrônica é com certeza uma propriedade essencial para o seu sucesso e demanda o desenvolvimento de soluções que garantam o desempenho. Na Seção 1 foi mencionado que um exemplo de tal solução é a distribuição dos servidores ou dos serviços prestados pelos mesmos entre várias máquinas. Nesse cenário, um desafio é garantir que a quantidade de trabalho realizada pelos vários servidores é equivalente. No contexto de servidores WWW, algumas pesquisas já foram desenvolvidas no sentido atingir um nível aceitável de balanceamento de carga [3, 16]. Nas seções a seguir discutimos os dois aspectos fundamentais da distribuição de serviços, o mecanismo de distribuição de requisições e a estratégia de balanceamento de carga.

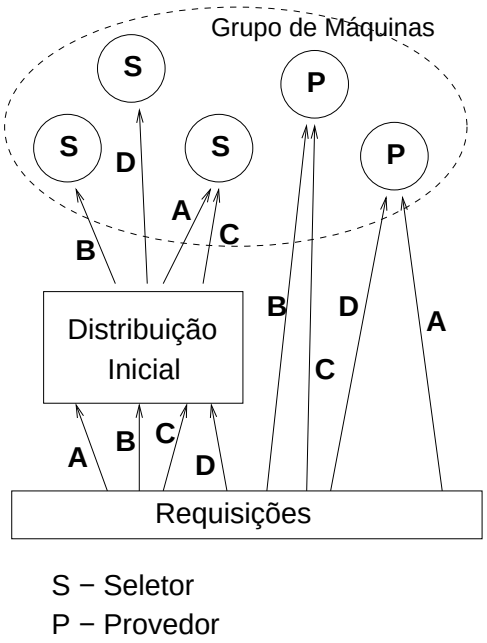

Figura 2: Diagrama dos serviços e fluxo de requisições em um servidor de anúncios distribuído

Na Figura 2 mostramos um típico servidor de anúncios distribuído onde os serviços estão divididos entre as várias máquinas disponíveis. Todas as requisições de seleção são interceptadas por um intermediário que impõe uma distribuição inicial às requisições. Podemos enumerar como possíveis intermediários servidores DNS [16], *switches* nível 4 [1, 10, 7], ou *switches* nível 7 [2]). A maneira como as requisições são repassadas varia e existem pesquisas que buscam tornar o redirecionamento transparente e eficiente [4, 15, 6]. Abaixo descrevemos em alto nível essas três formas de intermediação:

Servidor DNS A distribuição inicial pode ser feita configurando-se o servidor DNS para que, a cada vez que for consultado, ele indique um número IP diferente de uma das máquinas do grupo, o que é chamado de *Round Robin DNS*. Existe um problema grave que afeta bastante o balanceamento inicial que é o fato de os navegadores armazenarem localmente os números IP providos, de forma que requisições posteriores são sempre para o mesmo servidor.

***Switch* nível 4** Nesse caso, um hardware configurável realiza o mapeamento entre endereços IPs dos clientes e seletores. Tendo em vista a possibilidade de reconfigurações dinâmica, resulta em melhor balanceamento de carga que servidores DNS. Como desvantagens podemos mencionar o custo do equipamento e a possibilidade de se transformar em ponto de contenção, além de ser um ponto de falha.

***Switch* nível 7** É um componente com capacidade de processamento e que trata as requisições a nível de aplicação, ou seja, ele consegue verificar o conteúdo da requisição e tomar a ação apropriada. Nesse caso, o *switch* lê a requisição, verifica qual o conceito associado a ela e a envia diretamente ao servidor distribuído responsável por responder a requisições daquele conceito.

Os seletores devem ter informação a respeito de todos os provedores e quais anúncios estão sendo servidos por cada um deles. Essa informação é fundamental para decidir para qual provedor o seletor irá direcionar a requisição que recupera o anúncio. Essa decisão também considera a carga instantânea dos provedores.

Cada seletor, por sua vez, é responsável por responder às requisições associadas a um grupo de conceitos. Quando o intermediário repassa a requisição para um servidor de seleção, este pode não ser o responsável pelo conceito associado à requisição. Nesse caso, a requisição é novamente repassada para um seletor do grupo que será capaz de tratá-la. Logo, cada seletor mantem uma tabela indicando para todos os conceitos, quais os seletores responsáveis por eles. As estratégias para executar a retransmissão da requisição transparentemente e de forma segura não serão mencionados nem explicados nesse artigo [6, 15, 4].

3.1 Balanceamento dinâmico de carga nos servidores

Uma vez definida a arquitetura de um servidor de anúncios distribuído, é necessário definir as métricas e algoritmos para balancear a carga entre os servidores. No caso específico desse artigo o objetivo é que todos os servidores estejam sujeito a cargas tão equivalentes quanto possível.

Na Seção 3 foi mencionado que cada seletor recebe alguns conceitos de forma que uma requisição somente poderia ser tratada pelo(s) seletor(es) que detém o conceito ao qual a requisição está associada. Desta forma, a carga de um determinado servidor é função dos conceitos que ele está tratando e suas respectivas popularidades. A associação inicial de conceitos a servidores é feita aleatoriamente, de forma que no início a carga está completamente desbalanceada. Pode ocorrer, por exemplo, de um servidor receber os dois conceitos mais populares para tratar, fazendo com que sua carga seja mais alta do que a dos demais. A idéia do nosso algoritmo de balanceamento de carga dinâmico é permitir que situações como essa possam ser contornadas e que a carga dos seletores seja equilibrada. Para tal, é necessário que os seletores alterem as suas responsabilidades em termos de conceitos de forma a balancear a carga. Essas alterações são baseadas em três tipos de operação:

Migração: Nessa operação um seletor A requisita um conceito que está sendo tratado por seletor B. Nesse caso o seletor B, que atualmente é responsável pelo conceito, o cede

ao seletor A, que passa a atender as requisições para o conceito. Se o conceito que for requisitado estiver sendo tratado por mais de um seletor (isso acontece devido à operação de compartilhamento, explicada a seguir), todos os seletores responsáveis pelo conceito devem cedê-lo. A Figura 3 (a) ilustra uma migração entre os seletores B e C, onde o seletor B requisita o conceito "futebol" ao seletor C.

Compartilhamento: É uma operação semelhante à migração, exceto pelo fato de que o seletor para o qual o compartilhamento foi requisitado não deixa de ser responsável pelo conceito, ou seja, ambos passam a responder por aquele conceito. Neste caso, o assinalamento de requisições a seletores é feita de forma aleatória entre os seletores responsáveis pelo conceito. A Figura 3 (b) ilustra o compartilhamento do conceito "futebol" entre os seletores A e C, de forma que os dois passam a responder pelo refereido conceito.

Aglutinação: Representa a operação contrária à de compartilhamento. Aqui, caso mais de um seletor possua o mesmo conceito, um deles pode decidir por monopolizar o tratamento de requisições daquele conceito. Nesse caso ele solicita de volta os conceitos nos seletores que anteriormente executaram uma operação de compartilhamento. Logicamente é fundamental na aglutinação que mais de um seletor possua o conceito. A Figura 3 (c) ilustra uma aglutinação entre os seletores B e C para o conceito "escola". Nesse caso somente o seletor B continua respondendo por este conceito.

Todas as operações são implementadas através de mensagens entre os seletores. Esse custo e também o ganho em termos de balanceamento de carga de tais operações são mostrados e analisados na Seção 5.

Para que alguma dessas operações possa ser realizada, métricas devem ser definidas para indicar que um determinado seletor necessita executar uma delas. Essas métricas são definidas e explicadas posteriormente na Seção 5.

4 Caracterização da carga

Nesta seção descrevemos a carga de trabalho de um servidor de anúncios eletrônicos e discutimos algumas de suas peculiaridades. A carga utilizada como entrada para o simulador foi coletada nas máquinas de cache do POP-MG. Nós filtramos o log para que apenas as requisições a anúncios eletrônicos do servidor de anúncios *doubleclick* [2], tanto a filial brasileira quanto matriz internacional, fossem consideradas. Os dados foram coletados no período de 15 de dezembro de 2000 a 03 de janeiro de 2001.

O log coletado possui 332.564 requisições. 4% do log foi desconsiderado por se tratarem de requisições para as quais não foi possível relacionar as de seleção com as de provimento através da URL somente (por exemplo, *Flash*). $13,45\%$ das requisições eram para anúncios, classificadas como requisições de provimento. Esse pequeno número de requisições de provimento pode ser explicado por dois fatores. O primeiro fator é a existência de servidores proxy cache, que replicam as imagens dos anúncios, satisfazendo eventuais requisições de usuários. O segundo fator são potenciais robôs, que requisitam apenas a

[2] www.doubleclick.com

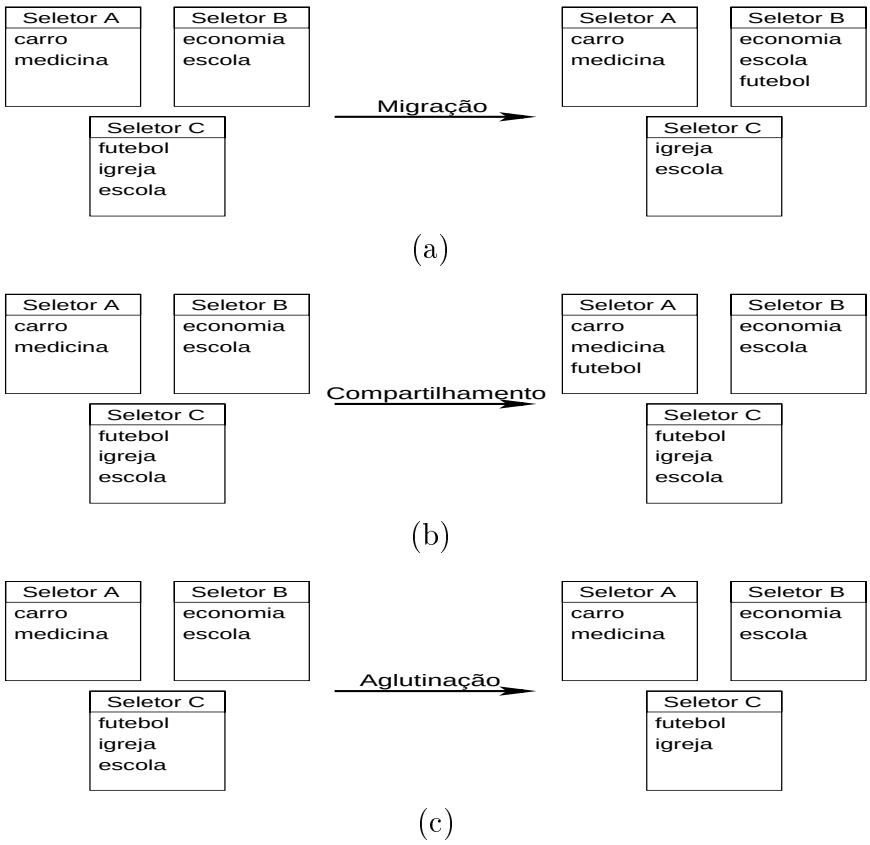

Figura 3: Possíveis operações no servidor de anúncios distribuído. a) Migração, b) Compartilhamento, c) Aglutinação

porção HTML das páginas, não requisitando quaisquer imagens embutidas. 82,54% das requisições são requisições de seleção, requisições cuja resposta contem o identificador de um anúncio a ser eventualmente provido. A *doubleclick* possui três tipos de requisições de seleção, que distinguem as três modalidades de provimento consideradas. 36,76% são requisições conjugadas, ou seja, requisições cuja resposta é o anúncio desejado e que não precisam ir a um provedor de anúncios. 25,34% são requisições por código *javascript* necessário para requisitar o anúncio e 37,90% são requisições para código html a ser incluído na página, o qual contem a requisição do anúncio.

A Figura 4 mostra a distribuição do número de requisições por dia. O número médio de requisições por dia de semana é 28.572, na primeira semana. A segunda semana foi descartada nesse cálculo por se tratar de uma semana especial, a semana entre o Natal e o Ano Novo. Podemos observar claramente no gráfico os finais de semana, quando o número de requisições é significativamente mais baixo. Dois destes finais de semana são prolongados, possuindo um terceiro dia com um número de requisições mais baixo, correspondente aos feriados.

Tendo em vista que toda a estratégia de balanceamento de carga é baseada na alteração da responsabilidade dos conceitos, é importante que a sua popularidade seja estudada, de forma a verificar o seu impacto nas estratégias de balanceamento de carga.

A lei de Zipf [17] foi originalmente usada em relacionamentos entre a popularidade de palavras em termos de *ranking* e sua freqüência. Ela diz que um *ranking* de popularidade

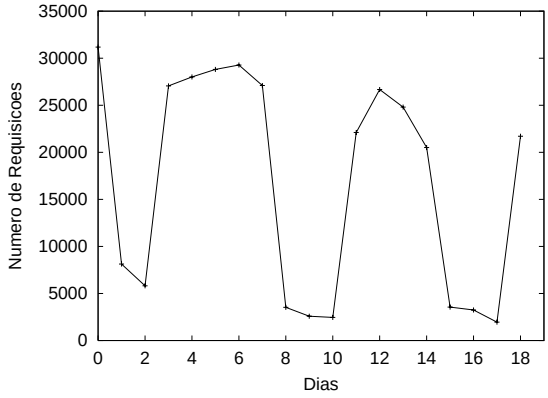

Figura 4: Distribuição do número de requisições por dia

de palavras usadas em um dado texto (ρ) é inversamente proporcional a sua frequência de uso (P), então: $P \sim 1/\rho$.

A Figura 5 mostra que parte da curva que representa o *ranking* dos conceitos se aproxima da lei de Zipf. A primeira parte da curva, até o ranking 20, não segue a lei de Zipf. Isso ocorre porque as 20 primeiras ocorrências do *ranking* possuem uma frequência muito próximas, variando de 0.21 a 0.01. A parte mais reta da curva é formada pelas posições de ranking de 21 a 300 e as suas frequências variam de 0.066 a 0.017. Esta caracterização mostra a necessidade de estratégias de balanceamento de carga dinâmicas, em razão da concentração de acessos a um pequeno grupo de conceitos.

A necessidade do balanceamento de carga para um servidor distribuído é confirmada pela Figura 6, que mostra a popularidade dos IPs em relação ao número de requisições de provimento e de seleção. Foram considerados como IPs únicos os IPs de classe C. A curva que representa as requisições de provimento se aproxima da função de Zipf com α (inclinação) igual a $1,3$, enquanto a curva das requisições de seleção se aproxima de Zipf com α igual a $1,98$. Estes resultados mostram uma grande variabilidade entre a quantidade de requisições submetidas por diferentes grupos de clientes.

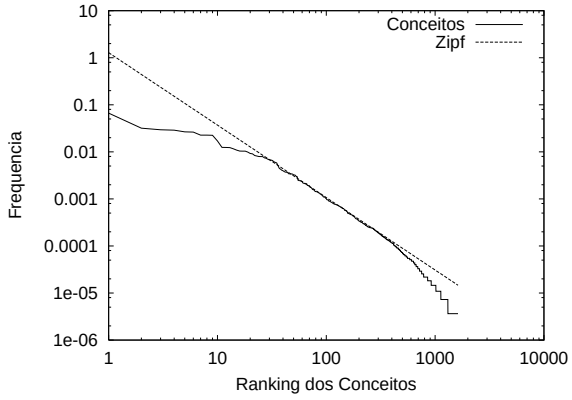

Figura 5: Popularidade dos conceitos

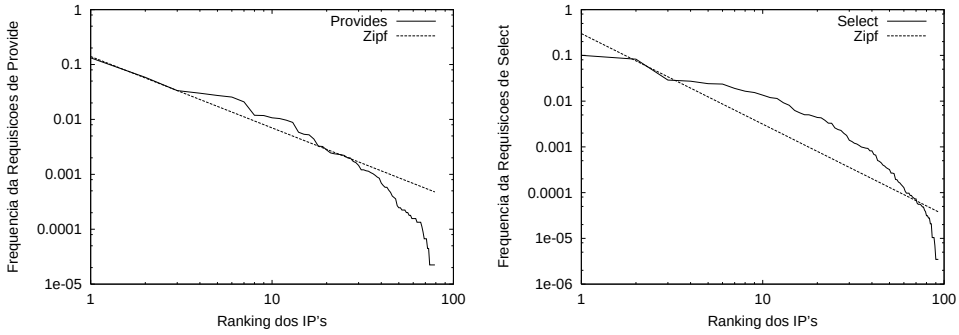

Figura 6: Popularidade das requisições do tipo *provimento* e *seleção*

5 Estudo de caso

Nesta seção apresentamos a arquitetura do simulador de servidores de anúncio distribuído que implementamos e discutimos os resultados obtidos com a simulação.

5.1 Arquitetura do simulador

O servidor distribuído é simulado por um programa baseado em *threads*. Há uma *thread* representando cada seletor e cada provedor que compõem o servidor de anúncios. Como mencionado, a comunicação entre os vários componentes é baseadas em mensagens, cuja transmissão é simulada através de *buffers* de mensagens implementados em memória compartilhada entre as *threads*. O simulador implementa completamente o funcionamento do servidor de anúncios, inclusive os protocolos de migração, aglutinação e combinação. As métricas de carga coletadas são independentes de arquitetura, ou seja, número e natureza de mensagens trocadas e operações executadas. A decisão por essas métricas aconteceu em virtude da dificuldade de modelarmos a latência entre máquinas arbitrárias na Internet, como seria o caso de um servidor geograficamente distribuído. Embora a validação dessa modelagem esteja sujeita a variáveis como latência de rede e capacidade de processamento das máquinas utilizadas, o nível de abstração empregado nos permite estudar os compromissos inerentes às estratégias de balanceamento de carga e sua interação com a carga de trabalho à qual um servidor distribuído seria submetido.

Desta forma, certas premissas foram adotadas no desenvolvimento do simulador. A primeira premissa é que os provedores possuem todos os anúncios a serem veiculados. Esta premissa permite um balanceamento de carga trivial entre os provedores e não compromete a sua implementação, pois as imagens associadas aos anúncios são pequenas (em torno de 7.2 kbytes) e o custo de armazenamento é muito baixo. A segunda premissa é que cada conceito é responsabilidade de um ou mais seletores, e que os seletores de um dado conceito devem manter, entre si, a contabilização dos anúncios gerados por eles de forma a satisfazer as demandas contratadas por anunciantes.

O simulador consiste basicamente em quatro rotinas: a primeira é responsável pelas inicializações, a segunda é uma *thread* que assinala as requisições e as envia para os seletores, a terceira rotina implementa os seletores e a quarta os processos provedores. A segunda rotina executa o papel de intermediário para a distribuição de requisições que chegam ao servidor de anúncios, empregando uma das estratégias discutidas na Seção 3.

A terceira rotina implementa, além do protocolo de seleção de anúncios, as operações descritas na Seção 3.1: migração, compartilhamento e aglutinação. A quarta rotina atende a requisições de provimento, contabiliza os dados necessários, e informa sua carga e a notificação da entrega de anúncios aos seletores.

A execução das operações de balanceamento de carga depende da satisfação de condições que são baseadas em três métricas coletadas nos seletores:

Requisições Repassadas São requisições para conceitos de responsabilidade do seletor que as recebeu, mas que foram repassadas por outro seletor, que por força da distribuição inicial, as recebeu. Uma vez recebidas, a seleção associada a essas requisições é executada normalmente.

Requisições Tranferidas São requisições para conceitos que não são de responsabilidade do seletor recipiente. Essas requisições são transferidas para um seletor apropriado (que seja responsável pelo conceito), sendo contabilizadas como "Repassadas" no receptor.

Requisições Respondidas É o número total de requisições recebidas pelo seletor e que são de sua responsabilidade, sejam elas assinaladas pelo mecanismo de distribuição inicial, ou repassadas por outro servidor.

Cada uma das operações de balanceamento de carga é executada quando o respectivo critério é satisfeito:

Migração: um seletor requisita a responsabilidade por um conceito quando a razão entre o número de requisições transferidas e o total de requisições respondidas para o conceito estiver acima do limiar de migração.

Compartilhamento: um seletor requisita a co-responsabilidade por um conceito baseado na mesma razão que define a migração de um conceito, entretanto o limiar de compartilhamento é normalmente menor.

Aglutinação: um seletor solicita a suspensão da co-responsabilidade por um conceito quando a razão entre o número de requisições repassadas e recebidas localmente ultrapassa o limiar de aglutinação.

Os limiares que regem a execução das operações de balanceamento de carga são parâmetros da execução do simulador.

5.2 Resultados

Os testes foram feitos utilizando 10 threads de seletores e 10 threads de provedores em uma máquina Sun Ultra-4 com 128Mb de RAM com o sistema operacional Solaris 5. O *log* utilizado foi o caracterizado na Seção 4. Para cada configuração apresentamos a média e o desvio padrão das métricas de desempenho dos vários seletores. A média da quantidade dos vários tipos de mensagem quantifica a carga global do sistema, enquanto o desvio padrão é uma medida de balanceamento de carga entre os seletores. Os limiares

	Mensagens	Respondidas	Repassadas	Transferidas
Sem nenhuma operação				
Média	61081.20	27471.60	24665.00	24668.80
Desvio Padrão	48162.65	24634.02	22569.65	40506.83
Com operação de migração				
Média	59642.10	27475.40	23218.80	23218.80
Desvio Padrão	49903.45	17424.98	13577.71	34790.78
Com operação de compartilhamento/aglutinação				
Média	64751.30	27473.90	23656.90	23658.40
Desvio Padrão	60711.49	22796.61	20444.82	37192.06
Com todas operações				
Média	64753.50	27475.20	23659.00	23659.20
Desvio Padrão	60715.10	22793.57	20442.27	37194.46

Tabela 1: Resultados para Switch nível 4

das métricas foram obtidos através de testes sucessivos, até ser encontrado um valor ótimo para os mesmos.

Na tabela 1 apresentamos os resultados para o esquema de distribuição *switch* nível 4. As medidas do simulador sendo executado sem nenhuma operação (migração, compartilhamento ou aglutinação) possui valores de média e desvio padrão que podem ser tomados como base para a distribuição de cargas do servidor de anúncios eletrônicos. Quando é executada a operação de *migração*, que acontece em pequena escala, os valores de desvios padrão de requisições respondidas e repassadas diminuem enquanto as de requisições transferidas não apresentam a mesma taxa de redução, indicando que efetivamente ocorreu alguma distribuição de carga. Para operações do tipo de *compartilhamento/aglutinação*, há uma pequena diminuição em relação ao caso base, entretanto o número de mensagens aumenta muito, pois aconteceram 4 aglutinações e 5 compartilhamentos e este segundo gera muitas mensgens de sincronização. Os valores de desvios padrão, quando são aplicadas todas as operações, se aproximam dos valores do caso anterior, pois o número de compartilhamentos e aglutinações ultrapassa muito o número de migrações. Em suma, a única operação que reduziu a carga global do sistema e propiciou balanceamento de carga foi a de migração, enquanto as outras configurações não foram efetivas.

Os resultados utilizando *Round-Robin DNS* (Tabela 2) demonstram que a aleatoriedade da distribuição de requisições inviabiliza qualquer estratégia de balanceamento de carga, uma vez que os valores das medidas são muito próximos, independente da configuração. É interessante comparar as Tabelas 1 e 2, e verificar como a localidade de referência associada aos IPs de origem e até a sua distribuição de frequência influenciam a eficácia da estratégia de migração.

O switch de nível 7, como funciona no nível da aplicação, já envia a requisição para o servidor responsável pelo conceito, logo não existem requisições transferidas e repassadas (como pode ser visto na Tabela 3), e como os limiares das operações dependem destes valores, não existe nenhuma operação. As mensagens existentes correspondem às requisições de seletores e as requisições de informação de carga dos provedores e notificações de pro-

	Mensagens	Respondidas	Repassadas	Transferidas
Sem nenhuma operação				
Média	61149.30	27474.80	24733.10	24733.70
Desvio Padrão	29901.54	24636.19	22183.29	2472.70
Com operação de migração				
Média	61148.90	27475.20	24733.50	24733.70
Desvio Padrão	29902.59	24636.61	22183.70	2472.70
Com operação de compartilhamento/aglutinação				
Média	63010.30	27475.40	24735.20	24735.20
Desvio Padrão	29270.79	23256.44	20950.54	2325.44
Com todas operações				
Média	63495.00	27475.30	24735.50	24735.60
Desvio Padrão	29079.52	23179.27	20879.51	2319.24

Tabela 2: Resultados para Round-Robin DNS

	Mensagens	Respondidas	Repassadas	Transferidas
Simulador em qualquer ocasião				
Média	36416.30	27475.40	0.00	0.00
Desvio Padrão	32358.46	24636.55	0.00	0.00

Tabela 3: Resultados para Switch nível 7

vimento dos mesmos. Neste caso, a métrica de balanceamento de carga deveria considerar somente a razão entre a carga de cada servidor e a carga média global, entretanto não desenvolvemos métricas deste tipo, pois, de acordo com [2], softwares front-ends baseados em conteúdo sofrem limitações de escalabilidade.

Uma outra questão que investigamos foi a ineficácia das operações de compartilhamento e aglutinação. Uma hipótese é baseada na popularidade dos conceitos. Como foi discutido, a popularidade dos conceitos mais frequentes não segue a lei de Zipf e os valores das popularidades são muito semelhantes. Como o número de servidores utilizados é menor que o número de conceitos com popularidade equivalente, não há demanda para compartilhamento de conceitos, mas apenas por migração. Para investigar essa situação, aumentamos o numero de seletores e verificamos a variação das várias métricas. Os resultados são apresentados na Tabela 4, onde podemos perceber uma diminuição global do número de requisições transferidas, demonstrando uma maior efetividade das operações de compartilhamento/aglutinação.

6 Conclusões e trabalhos futuros

Neste artigo discutimos a distribuição de serviços de comércio eletrônico e as estratégias de balanceamento de carga para esses servidores distribuídos. As nossas propostas foram avaliadas por meio de simulação e demonstraram que a migração de responsabilidades

	Mensagens	Respondidas	Repassadas	Transferidas
Simulador com 15 Seletores				
Média	40341.66	18315.26	16058.53	16060.20
Desvio Padrão	41623.37	13780.67	9745.94	27690.31
Simulador com 20 Seletores				
Média	30470.80	13737.70	12258.45	12258.45
Desvio Padrão	36659.27	11809.68	8559.54	25438.63
Simulador com 25 Seletores				
Média	24458.00	10990.16	9888.08	9888.08
Desvio Padrão	33553.98	10493.52	7430.55	23305.97

Tabela 4: Resultados com variação do número de seletores

entre seletores de anúncios reduz a carga do sistema e permite que a carga imposta a cada um dos seletores seja mais balanceada. Já estratégias de compartilhamento de conceitos (responsabilidades) não foram efetivas em consequência da popularidade dos conceitos, que se apresentava bastante semelhante.

Este trabalho é uma primeira prospecção em um tema muito rico e tem várias formas de continuidade possíveis. Dentro do contexto do que foi apresentado, é necessário realizar um estudo sistematizado dos números ótimos de seletores e provedores, além dos limiares para as estratégias de balanceamento de carga. O baixo número de operações realizadas indica que talvez o log utilizado ainda seja pequeno, demandando uma coleta mais extensa.

A contabilização de custos de operação e a otimização dos protocolos, minimizando mensagens através de agregação e envio periódico são também atividades a serem realizadas, juntamente com a verificação do comportamento de outros servidores de anúncio que não a *doubleclick*, além de uma caracterização mais criteriosa da variação temporal das demandas por anúncios eletrônicos. Finalmente, pretendemos investigar a distribuição e problemas de balanceamento de carga em outras aplicações Internet e de comércio eletrônico.

Referências

[1] E. Anderson, D. Patterson, and E. Brewer. The magicrouter: An application of fast packet interposing, 1996.

[2] M. Aron, D. Sanders, P. Druschel, and W. Zwaenepoel. Scalable content-aware request distribution in cluster-based network servers. In *In Annual Usenix Technical Conference, 2000. San Diego, CA.*, 2000.

[3] L. Aversa and A. Bestavros. Load balancing a cluster of web servers using distributed packet rewriting. In *Proceedings of the 2000 IEEE International Performance, Computing, and Communications Conference*, pages 24 – 29, February 2000.

[4] A. Bestavros, M. Crovella, J. Liu, and D. Martin. Distributed packet rewriting and its application to scalable server architectures. In *Proceedings of the International Conference on Network Protocols*, October 1998.

[5] P. Cao, J. Zhang, and Kevin Beach. Active cache: Caching dynamic contents on the web. In *Proc. of IFIP International Conference on Distributed Systems Platforms and Open Distributed Processing*, pages 373–388, 1998.

[6] A. Cohen, S. Rangarajan, and H. Slye. the performance of tcp splicing for url-aware redirection. In *Proceedings of the 2nd USENIX Symposium on Internet Technologies and Systems*, October 1999.

[7] IBM Corporation. Ibm interactive network dispatcher. *http://www.ibm.com/software/network/dispatcher*.

[8] B. Diniz, W. Meira Jr., and V. Almeida. Análise de desempenho da terceirização de serviços de comércio eletrônico. In *Anais do XVIII Simpósio Brasileiro de Redes de Computadores*, Belo Horizonte, MG, Maio 2000. SBC.

[9] B. Diniz, R. Pereira, W. Meira Jr., and V. Almeida. Qos in parallelized e-commerce systems. In *Proceedings of the XII SBAC-PAD - Simpósio Brasileiro de Arquitetura de Computadores e Processamento de Alto Desempenho*, São Pedro, SP, Maio 2000. SBC.

[10] Cisco Systems Inc. Localdirector. *http://www.cisco.com*.

[11] Napster Inc. Napster home page. *http://www.napster.com*.

[12] W. Meira Jr., D. Menascé, V. Almeida, and R. Fonseca. E-representative: a scalability scheme for e-commerce. In *Proceedings of the Second International Workshop on Advanced Issues of E-Commerce and Web-based Information Systems (WECWIS'00)*, June 2000.

[13] A. Luotonen. *Web Proxy Servers*. Prentice Hall, Upper Saddle River, New Jersey, 1998.

[14] W. Meira Jr., M. Cesário, R. Fonseca, and N. Ziviani. Integrating www caches and search engines. Global Internet 1999, Rio de Janeiro, RJ, December, 1999.

[15] V. Pai, M. Aron, G. Banga, M. Svendsen, P. Druschel, W. Zwaenepoel, and E. Nahum. Locality -aware request distribution in cluster-based network servers. In *ASPLOS-VIII. Eighth International Conference on Architectural Support for Programming Languages and Operating Systems*, pages p. 205–16. 3–7, October 1998.

[16] R. Schemers. lbmnamed: A load balancing name server in perl. In *Proc. of the 9th Systems Administration Conf., Monterey*, September 1995.

[17] G. Zipf. *Human Behavior and the Principle of Least Effort*. Addison-Wesley, Cambridge, MA, 1949.