Servidores Paralelos e Distribuídos de Comércio Eletrônico

T. Cançado [1], A. Pereira[1], B. Abrahão[1], R. Pereira[1], W. Meira Jr.[1], C. Amorim[2], A. Faustino[2], S. Dias[2]
[1] e-SPEED - Departamento de Ciência da Computação
Universidade Federal de Minas Gerais
Belo Horizonte — Brasil
{tassni,adrianoc,brut,rpereira,meira}@dcc.ufmg.br
[2] LCP - COPPE/Sistemas
Universidade Federal do Rio de Janeiro
Rio de Janeiro — Brasil
{amorim,faustino}@cos.ufrj.br

Resumo—

A popularização da Internet torna o comércio eletrônico uma de suas aplicações mais promissoras, o que explica o aumento significativo da carga observada nos sites e a consequente degradação do desempenho de seus servidores. Em paralelo, os usuários estão mais exigentes e criteriosos ao escolher os serviços utilizados e, portanto, escalabilidade tornou-se ponto chave para atender satisfatoriamente esses consumidores. Este artigo apresenta um mecanismo de distribuição de serviços e dados entre servidores de transação estendendo o trabalho anterior, onde apenas a replicação de dados em servidores de comércio eletrônico foi abordada.

Essa abordagem foi implementada e validada no âmbito de uma livraria virtual e *logs* de acesso reais e demonstra que o esquema proposto aumenta efetivamente a escalabilidade do servidor. Por outro lado, descobrimos que a demanda de comunicação entre os servidores distribuídos é crítica e requer análises futuras.

Palavras-chave— **comércio eletrônico, sistema paralelo e distribuído, escalabilidade**

Abstract—

The popularity of the Internet makes e-Commerce one of its most promising applications, explaining the significant increase in the load observed in the sites, and the consequent performance degradation. Further, users are requiring quality of service from their service providers, and scalability became a key issue for reaching those customers. In this paper we extend previous work on replicating data in e-commerce servers by distributing both services and data among a cluster of servers.

This approach was implemented and validated using a virtual bookstore and actual usage logs, and shows that the schemes proposed increase the scalability of the server. On the other hand, we also found that communication among distributed servers becomes a big issue, requiring further investigation.

Keywords— **ecommerce, parallel and distributed system, scalability**

I. Introdução

O comércio eletrônico tem se firmado como uma aplicação que tem não apenas facilitado o acesso a bens e serviços, como revolucionado a economia como um todo. Esses serviços de comércio eletrônico são implementados por servidores WWW, que devem, entre outros requisitos, suportar a grande variabilidade de carga imposta a esses servidores.

De uma forma abrangente, podemos definir comércio eletrônico como uso de recursos de rede e tecnologia da informação para facilitar processos centrais ao funcionamento de uma organização. Existe um consenso de que o alto volume de requisições e o baixo desempenho são alguns dos grandes empecilhos para um maior crescimento do comércio eletrônico.

Pesquisas revelam que a maioria dos consumidores, ao se deparar com um *site* cujo tempo de resposta não corresponde ao que eles consideram razoável, costumam desistir e procurar serviços em outros *sites* que atendam à sua necessidade com qualidade [MEN 99]. Logo, torna-se extremamente importante minimizar esse tempo, evitando a perda de clientes.

No entanto, minimizar o tempo de resposta a uma requisição efetuada por um cliente não é uma tarefa fácil, haja vista a grande quantidade de dados processados ao atender a mesma e a complexidade de um servidor de comércio eletrônico. O armazenamento desses dados, em geral, é feito em sistemas de gerência de banco de dados (SGBD), pois estes garantem três propriedades fundamentais: coerência, consistência e persistência. Por outro lado, o custo de acesso a SGBDs é normalmente alto, resultando em um aumento da latência observada pelos clientes [PAI 00].

Neste artigo apresentamos uma implementação do servidor de comércio eletrônico bastante aperfeiçoada em relação à sua versão preliminar descrita em [PER 00]. O primeiro aperfeiçoamento foi com relação à organização do cache e da estrutura de banco de dados, de forma a permitir uma maior escalabilidade para maior volume de dados armazenado e servidores sendo utilizados. O segundo está relacionado com o assinalamento dinâmico de requisições, integrando o servidor web e servidor de transações.

Este artigo está dividido em sete seções. Na próxima seção, descrevemos a arquitetura de servidores de comércio eletrônico. A implementação da nossa estratégia de

replicação e distribuição é descrita na Seção III, seguido de um estudo sobre modelos de replicação de dados nesse contexto. Na seção V, é analisada uma carga de trabalho real de um servidor de comércio eletrônico e discutidas as oportunidades de replicação. Na Seção VI, são apresentados os resultados obtidos em experimentos utilizando uma carga real. Por fim, a Seção VII, mostra as conclusões e os trabalhos futuros.

II. Servidores de Comércio Eletrônico

Servidores de comércio eletrônico têm como principal função prover processamento transacional para a comercialização de bens físicos e virtuais. Nesta seção, discutimos algumas características destes servidores e como elas afetam a implementação dos servidores. Descrevemos os componentes de um servidor de comércio eletrônico e como eles interagem entre si.

A. Arquitetura

A.1 Componentes

Um servidor de comércio eletrônico pode ser dividido em três componentes integrados (Figura 1):

1. **Servidor WWW**: É o gerente das tarefas, sendo responsável pela interface com os usuários (clientes), interagindo diretamente com eles, recebendo as solicitações de consultas ao banco de dados, disparando-as, e repassando os resultados. Provê a interface entre a ferramenta de acesso do cliente (normalmente um *browser*) e o servidor de comércio eletrônico.
2. **Servidor de Transações**: Realiza o processamento das requisições submetidas ao servidor de comércio eletrônico, como a adição/remoção de um novo produto à cesta de compras.
3. **Sistema de Gerência de Banco de Dados(SGBD)**: Armazena todas as informações da loja virtual, como descrição do produto e nível de estoque. Mais que um simples repositório, agrega uma série de funcionalidades que permitem o acesso padronizado, seguro e eficiente aos dados, através, por exemplo, da criação de índices e controle de acesso utilizando autenticação por usuário.

Figura 1. Arquitetura de 3 níveis

Em geral, as requisições são recebidas pelo servidor WWW que as repassa ao servidor de transações. Este, por sua vez, requisita dados quando necessário ao servidor de bancos de dados.

A.2 Desempenho

A qualidade do serviço prestado pelos servidores de comércio eletrônico é medida geralmente utilizando métricas como tempo de resposta e taxa de serviço [PAI 00].

Tempo de resposta é o intervalo de tempo entre o recebimento da requisição e o envio total da resposta. Esse tempo é geralmente medido no próprio servidor, uma vez que o tempo de resposta observado pelo cliente é muito difícil de ser medido pela variância das condições de rede e conectividade dos clientes.

A taxa de serviço é o número de requisições atendidas por unidade de tempo. Unidades de medição comuns de taxa de serviço são requisições por segundo e transações por segundo. Essa métrica é bastante importante do ponto de vista da eficiência do(s) servidor(es), uma vez que uma baixa taxa de serviço pode levar um comprador em potencial a desistir de suas compras.

Existem alguns fatores que podem prejudicar significativamente o desempenho dos servidores em razão de contenção por recursos. Dentre os quais, destacam-se:

- **Comunicação**: O tráfego na Internet está cada vez mais intenso, podendo elevar o tempo de execução de algumas operações. Outra fonte de contenção pode ser o grande número de requisições a serem tratadas por um servidor, que pode sobrecarregar a interface de rede do mesmo.
- **Processamento**: Tanto o tempo gasto em processamento de tarefas (CPU), quanto acesso a disco podem comprometer as operações entre servidor e cliente, uma vez que o número de requisições seja maior que a capacidade de execução do servidor.
- **Acesso a Dados**: A velocidade de acesso bem como a taxa de serviço dos servidores de bancos de dados utilizados são fatores críticos para o bom desempenho do servidor como um todo.

Neste artigo, avaliamos técnicas para a minimização do impacto desses fatores, através da utilização de vários servidores de transação e técnicas de replicação de dados e distribuição de operações.

B. Armazenamento de Dados

Discute-se, nesta seção, alguns dos fatores associados aos serviços oferecidos pelos servidores de comércio eletrônico e algumas características dos dados utilizados para descrição dos produtos comercializados. Podemos dividir os dados armazenados por servidores de comércio eletrônico em duas grandes categorias: produtos e sessões de usuários. Dados de produtos contém não apenas as especificações dos produtos, como o seu preço e disponibilidade. Dados de sessões de usuário contém toda a informação relativa às interações dos usuários com o servidor, constituindo o estado do servidor.

Um aspecto fundamental para o armazenamento e

distribuição de dados é sua volatilidade, ou seja, a frequência de atualização de um atributo de produto ou sessão de usuário. Com relação à volatilidade, podemos dividir os dados em três categorias:

1. **Estáticos:** São atributos que permanecem inalterados durante toda a execução de um servidor de comércio eletrônico, como por exemplo descrições de produtos e dados cadastrais dos usuários. No caso de uma livraria virtual, os títulos e autores dos livros são exemplos de dados estáticos.
2. **Pouco voláteis:** Incluem os atributos cujo conteúdo varia a uma taxa muito baixa (p.ex., a colocação de um livro no *ranking* de uma livraria virtual, ou as opiniões a respeito de um produto que foram postadas por outros consumidores). De modo geral, os dados pouco voláteis são modificados em função das ações tomadas pelos administradores do *site*, e não da interatividade entre os clientes e a loja.
3. **Muito voláteis:** Incluem os dados modificados com grande rapidez, em geral em função das interações entre os clientes e o servidor de comércio eletrônico. O melhor exemplo é provavelmente a disponibilidade dos produtos. Toda vez que um consumidor compra um produto, a informação sobre a disponibilidade do produto deve ser atualizada. A manipulação de dados muitos voláteis é normalmente mais complexa e em consequência exige maior atenção.

III. Estudo de Caso: Loja Virtual

Nesta seção discutimos a implementação de um servidor de comércio eletrônico tradicional, uma livraria virtual.

A. Operações

Sem perda de generalidade, podemos caracterizar o funcionamento de uma livraria virtual por seis operações:

Página Principal (home)**:** É a página inicial do servidor de comércio eletrônico, a qual normalmente é estática e respondida apenas pela leitura de um arquivo armazenado no servidor de transações.

Navegação (browse)**:** Apresenta os produtos organizados por categorias, normalmente o resultado de um consulta ao banco de dados sobre o atributo categoria de cada produto.

Busca (search)**:** Lista os produtos que satisfazem o critério de busca, os quais são determinados por uma busca por padrões similares em todos os atributos da base de dados de produtos.

Seleção (select)**:** Exibe todas as informações disponíveis de um produto selecionado pelo usuário, tais como descrição completa e preço.

Adição à Cesta (add)**:** O usuário reserva um produto para posterior compra, ação registrada na sessão do próprio usuário. A quantidade de produtos disponíveis é atualizada para refletir a opção de compra.

Pagamento (pay)**:** O usuário conclue a aquisição de produtos confirmando a compra dos produtos que estão na sua cesta de compras. O nível de estoque é atualizado em definitivo e os dados do usuário consultados para despacho dos bens adquiridos.

B. Estruturas de Dados

Descrevemos aqui a estrutura de dados utilizadas para implementar as várias operações de uma loja virtual descritas na última seção. O banco de dados possui basicamente quatro tabelas:

Clientes: Contém os dados cadastrais de clientes.

Produtos: Contém os dados dos produtos, incluindo o seu preço e nível de estoque.

Termos: Contém todos os termos únicos das descrições de produtos.

Listas invertidas: Cada entrada desta tabela é formada por um termo e um produto, indicando a ocorrência do termo nos atributos do produto.

Sessão: Contém as opções de compra de um cliente.

Todas as operações à exceção da busca são implementadas com consultas ou atualizações simples à base de dados. A busca é implementada por uma consulta à tabela de lista invertida[BAE 99] que retorna os produtos que contêm o termo. No caso de buscas com vários termos, o resultado é a interseção entre as consultas por termos individuais.

IV. Servidores Paralelos e Distribuídos

Nesta seção discutimos como a escalabilidade de lojas virtuais pode ser aumentada através das técnicas de replicação de dados, paralelização e distribuição. O servidor de transações é o principal componente neste sistema de três níveis, sendo o melhor candidato para aplicação das diversas estratégias. Como visto em [PER 00], estratégias de replicação de dados e de paralelização do servidor de transação são formas eficazes de melhorar taxas de atendimento e evitar o acesso ao banco de dados, que é de alto custo. Torna-se necessário, então, analisar tais estratégias e encontrar soluções que possibilitem a escalabilidade do sistema em ambientes de bases de dados crescentes e grande volume de acessos. Nas seções a seguir discutiremos cada uma dessas estratégias.

A. Replicação de Dados

A replicação de dados no servidor de transação, implementada e discutida em [PER 00], torna o acesso aos dados mais rápido, dispensando o alto custo de acesso ao SGBD. Desta forma, os livros e termos de consultas com maior acesso (e consequente frequência de leitura) estão presentes

na *cache* do servidor, aumentando o número de requisições atendidas por unidade de tempo.

Nota-se que a localidade temporal das consultas é explorada, trazendo melhorias visíveis nos tempos de resposta para os casos de *hit*. Nessa perspectiva, as operações de leitura (mais frequentes) são fortemente privilegiadas.

Apesar do conceito de replicação de dados ser o mesmo, a implementação foi significativamente alterada em razão das mudanças na estrutura de dados. Mais ainda, as listas de produtos são armazenadas em memória de forma paginada, permitindo um melhor aproveitamento da memória, uma vez que, na maioria dos casos, apenas as primeiras páginas de uma busca são efetivamente consultadas.

B. Paralelização

Uma vez que os dados manipulados em um servidor de comércio eletrônico são em sua maioria não-voláteis, é possível não somente diminuir os tempos de resposta a requisições, como também aumentar o número de clientes servidos se o servidor que os atende for paralelo. Em uma implementação mais eficiente, cada requisição é atendida por uma *thread*, permitindo que vários clientes sejam atendidos simultaneamente. As modificações a dados críticos se faz necessária e é implementada utilizando semáforos.

C. Distribuição

As estratégias de replicação descritas aproveitam-se dos acessos frequentes aos mesmos dados e do fato de serem eles utilizados geralmente para leitura somente. Entretanto, havendo uma base de dados maior (e potencialmente crescente) e o aumento do número de clientes a um *site*, não são suficientes apenas replicação e paralelismo. Inicialmente, porque o custo de acesso ao banco de dados é ainda maior em caso de *miss* na *cache*, já que a base existente é consideravelmente maior. Por outro lado, cada servidor paralelizado terá restrições no número de requisições atendidas de acordo com o número de *threads* que executar. Essa última é uma restrição forte, pois é de conhecimento geral que o ganho com a utilização de *threads* está vinculada obrigatoriamente ao número de processadores na máquina que comporta o servidor de transação.

Para minimizar estes problemas, propomos a utilização de um conjunto de servidores de transação atuando cooperativamente. Desta forma, explora-se os recursos de cada servidor:

- Quanto ao poder de processamento: os servidores paralelos em conjunto serão capazes de atender a um maior número de requisições simultaneamente, dispensando o alto custo em utilização de computadores multiprocessados.
- Quanto à memória: o volume de dados armazenados em *cache* é maior, diminuindo as taxas de atualização da mesma e permitindo que as taxas de acerto sejam ainda maiores.
- Quanto aos recursos de rede: mais requisições poderão ser repassadas do servidor *web* para os servidores de transação. Esse ponto merece maior cautela, uma vez que a comunicação entre os servidores poderá reverter este quadro, transformando-se em um ponto de contenção. Além disso, avalia-se o desempenho oferecido pelo servidor *web*, agora mais exigido em sua capacidade de repassar requisições e receber as respostas produzidas.

A partir da implementação de uma estratégia de distribuição, pretendemos verificar se os ganhos esperados serão realmente atingidos e de que forma este resultado afeta o desempenho do servidor de comércio eletrônico como um todo.

Para implementar a distribuição dos servidores de transação, foi necessário discutir duas questões distintas e complementares. Primeiramente, foi preciso determinar uma política de distribuição que indicasse quais requisições seriam atendidas por quais servidores, ou seja, como o servidor *web* redireciona as consultas recebidas entre os servidores de transação. Além disso, deve-se determinar de que forma os dados e operações serão distribuídos, como e quais dados obter dos outros servidores e de que maneira manter a consistência e a coerência das sessões dos usuários. A seguir serão mostrados detalhes de implementação de servidores distribuídos.

C.1 Distribuição de Requisições

Na arquitetura de servidores de comércio eletrônico distribuídos, é necessário definir a política de assinalamento de requisições a servidores de transações, a qual é implementada pelo servidor WWW. Neste caso, o servidor WWW deve garantir que as requisições de uma mesma sessão sejam sempre direcionadas ao mesmo servidor de transação, uma vez que os dados das sessões dos usuários são armazenados em um único servidor.

A estratégia de distribuição de requisições utilizada é *round-robin*. Por exemplo, se houver n servidores de transação, a primeira requisição da primeira sessão que chegar ao servidor WWW é enviada ao primeiro servidor de transações, a primeira requisição da segunda sessão é enviada ao segundo servidor e assim sucessivamente, até que a primeira requisição da sessão $n + 1$ é enviada ao primeiro servidor novamente, reiniciando o ciclo de distribuição.

Na figura 2, podemos ver o funcionamento de um servidor WWW que utiliza a técnica de *round-robin* para sua distribuição:

- As requisições da sessão *A* são direcionadas para o servidor de transação *LV1*;
- As requisições da sessão *B* são direcionadas para *LV2*;
- As requisições da sessão *C* são direcionadas para *LV3*;

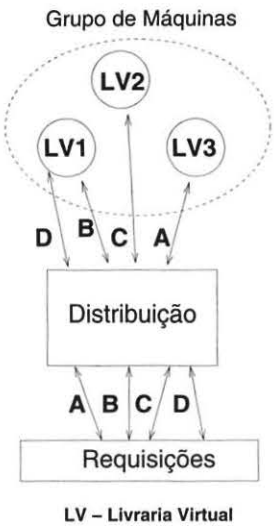

Figura 2. Diagrama das requisições entre os clientes e os servidores de aplicação distribuídos

- As requisições da sessão *D* são direcionadas para o servidor *LV1*.

Utilizamos o servidor HTTP *Apache* para tratar as requisições HTTP da livraria virtual, entretanto, para efetuar o redirecionamento, foi criado um módulo que possui a lista de todas as máquinas, de forma a subsidiar a distribuição. Esta operação utiliza o número da sessão de cada requisição e o número de servidores de aplicação disponíveis para determinar o destino de cada requisição.

C.2 Distribuição de Dados e Operações

A política de distribuição dos dados e das operações foi formulada tomando-se como base os seguintes pressupostos:

1. Os servidores de transação recebem indistintamente quaisquer requisições, independente da operação solicitada.
2. A distribuição é configurada estaticamente, ou seja, os servidores que a compõe sabem *apriori* todos os participantes e como identificá-los.
3. Cada nó da distribuição é responsável por um conjunto de termos de consulta e um conjunto de livros. Tais conjuntos são disjuntos e determinados aplicando-se uma função sobre o termo ou livro, a partir da qual associa-se a consulta a um dos nós.
4. Identificado o nó responsável por responder a uma consulta, são trocadas mensagens entre os servidores, de forma a obter o resultado da operação solicitada.
5. O servidor que recebeu a requisição originalmente será aquele que retornará a resposta para o servidor *web*.

A Figura 3 ilustra o processo de distribuição. Recebida uma requisição, o termo de busca ou o identificador do livro selecionado é aplicado a uma função. Essa função retorna

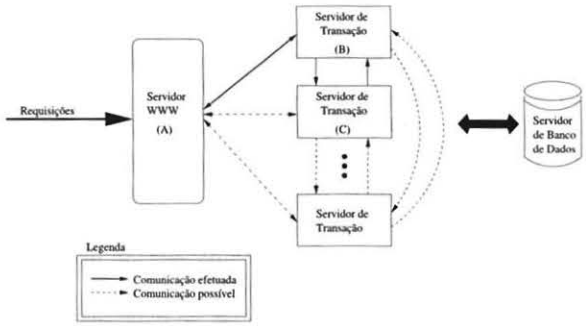

Figura 3. Estratégias de distribuição

um número correspondente ao identificador do servidor que é responsável por responder à esta consulta. Caso o identificador retornado corresponda ao servidor corrente (B), este efetua a operação, cria uma página de resposta e retorna a mesma para o servidor *web* (A). Caso contrário, a *thread* que no momento está atendendo à consulta envia uma mensagem via *socket* ao servidor identificado (C). Nessa mensagem estão contidos a operação a ser realizada e os seus parâmetros, o identificador de sessão do usuário e quaisquer outras informações relevantes à execução da operação. A partir desse ponto, o servidor C efetua a operação, monta a página de resposta e a envia ao servidor B. De posse da página de resposta, esta é finalmente retornada ao servidor A.

Através desse esquema, observa-se que não só todas as operações são distribuídas pelos servidores, como também a *cache*. Este fato é de extrema importância, pois evita-se que para uma mesma consulta sejam necessários vários acessos ao banco de dados. Além disso, o recurso de memória utilizado para comportar a cache, bem como de processamento para sua gerência e política de reposição, são melhor aproveitados, reduzindo a redundância dos mesmos produtos ou termos replicados em mais de um servidor. Por essa razão, pretende-se obter ganhos com a distribuição ao aumentar as taxas de acerto. Se isso não for alcançado, o custo de comunicação entre os servidores deverá resultar em maior tempo de resposta e pior qualidade de serviço oferecido.

Como o custo de comunicação entre os servidores tende a ser alto, as respostas de *search* e *browse* são limitadas a 25 produtos no máximo, como é usual em sites reais, dando a oportunidade de navegação pelas páginas subsequentes de resposta.

Como mencionado, outra medida adotada para a melhorar o tempo de resposta do banco de dados foi a criação de listas invertidas. Tais listas contituem-se de tabelas no próprio banco de dados que contém informações sobre as palavras nele existentes, em quais livros ocorre e a quantidade de ocorrências. A operação de *search* é de alto custo por se utilizar de operação de casamento de padrão (*LIKE*), e passa

a ser implementada por uma consulta pelas palavras e suas ocorrências. A consulta exata por palavras, substituindo a operação de busca por expressões *LIKE*, representa ganho significativo no tempo de resposta do SGBD. Essa forma de indexação é de alto custo para a criação inicial das tabelas de palavras e ocorrências, mas de manutenção trivial, já que os dados textuais sobre livros não mudam com frequência e a inserção de novos livros pode ser controlada por mecanismos de gatilhos (*triggers*) no próprio SGBD.

Ordenando a tabela de ocorrências de palavras em livros pelo número de ocorrências, privilegia-se as respostas sobre livros nos quais há maior ocorrência do termo. Além de melhorar o tempo de resposta da consulta, tem-se ainda um aperfeiçoamento da resposta em si, retornando informações com maior relevância para a busca solicitada.

V. Caracterização da Carga de Trabalho

Nesta séção, descrevemos a estratégia utilizada para gerar a carga de trabalho utilizada nos testes. Essa carga de trabalho é baseada em registros de acesso de uma livraria virtual real descrita em [MEN 00, PER 00]. O nosso objetivo é escalar a carga de trabalho de forma a simular o servidor de comércio eletrônico, atendendo um maior número de usuários ou servindo uma base de produtos maior.

A. Geração da Carga de Trabalho

A geração da carga é baseada no método da Tranformada Inversa [ROS 97], que, a partir de uma dada distribuição de probabilidade, cria uma amostra randômica de cada variável aleatória que compõem a carga.

Assumindo que uma variável aleatória possa assumir n valores $x_1 \ldots x_n$, e uma distribuição de probabilidade $p_1 \ldots p_n \mid P(X = x_j) = p_j, \sum_j p_j = 1, 1 \leq j \leq n$, podemos gerar um conjunto X de eventos que sigam essa distribuição como:

$$X = \begin{cases} x_0 \; se & U < p_0 \\ x_1 \; se & p_0 \leq U < p_0 + p_1 \\ \vdots & \\ x_j \; se & \sum_{i=1}^{j-1} p_i \leq U < \sum_{i=1}^{j} p_i \\ \vdots & \end{cases}$$

Pelo fato de que, para $0 < a < b < 1$,

$$P\{a \leq U < b\} = b - a,$$

temos que

$$P\{X = x_j\} = P\left\{\sum_{i=1}^{j-1} p_i \leq U < \sum_{i=1}^{j} p_i\right\} = p_j$$

Logo, X tem a distribuição desejada.

Resumidamente, o método consiste em achar a probabilidade de cada elemento, resultando em uma distribuição de probabilidade do conjunto e posteriormente a distribuição acumulada dessa variável aleatória. Dado um número aleatório normalizado, devemos encontrar o intervalo, no qual a função $F(U)$ se encontra. A partir dessa operação, temos mais um elemento da amostra, dado pela função inversa $F^{-1}(U)$.

Podemos descrever o processo em forma de algoritmo como:

Gere um numero aleatório U;
Se $(U < p_0)$ $\{X = x_0$; Pare;$\}$
Se $(U < p_0 + p_1)$ $\{X = x_1$; Pare;$\}$
Se $(U < p_0 + p_1 + p_2)$ $\{X = x_2$; Pare;$\}$
$\vdots$

Algumas otimizações possíveis para o processo de geração de carga são descritas a seguir. Como o algoritmo percorre os intervalos de probabilidade, será mais eficiente se esses estiverem ordenados em ordem decrescente. Uma outra otimização que pode ser feita deve-se ao fato de que a probabilidade acumulada é sempre crescente($F(x) < F(x+1)$). Com essa ordenação, podemos reescrever o algoritmo acima, substituindo a pesquisa linear por pesquisa binária ou outro método mais eficiente.

B. Expansão do Banco de Dados

Para expandir o banco de dados, devemos criar novos nomes de produtos(livros), autores, descrições e categorias associadas que devem seguir o mesmo comportameto das informações originais, ou seja seguir a mesma distribuição de probabilidade em relação à frequência de palavras e tamanho do texto para cada um dos atributos da base de dados.

Para realizar a tarefa, fizemos para cada atributo da base de dados uma tabela de frequência de palavras e de tamanhos (em número de palavras). Tomando como elementos as palavras e tamanhos, podemos usar o método da Tranformada Inversa para obter o valor de cada atributo e assim um banco de dados expandido com a mesma característica do original.

C. Geração de Requisições

A lógica para geração de requisições é semelhante à usada para a expansão do banco de dados. O processo baseia-se na alteração do *log* da livraria virtual original, visando a adaptação para a nova base de dados. A frequência de operações e sessões de usuário permanecem intactas, bastando somente a mudança de parâmetros das operações. A alteração de cada uma das operações é realizada como a seguir:

Busca: Uma vez que a base de dados expandida utiliza um espaço de palavras igual ao original, o número de itens retornados nas pesquisas aumentará proporcionalmente. Assim não é necessario fazer alterações nessas

operações.

Navegação: Também não há necessidade de alteração, uma vez os novos produtos criados sinteticamente estão distribuídos entre as categorias existentes na base de dados original, segundo a mesma distribuição dos produtos dessa base.

Seleção: Gera-se uma sequência de acessos ao novo conjunto de produtos que segue a distribuição de probabilidades das operações de seleção originais. Para cada operação desse tipo presente no *log*, trocamos o parâmetro original com um elemento da nova sequência, escolhido de acordo com uma função de popularidade semelhante à original.

Adição à Cesta: Idem à operação de seleção, usando a distribuição original de operações de adição.

Pagamento e Página Principal: Essas informações permanecem inalteradas, pois não utilizam quaisquer parâmetros.

VI. Resultados

A. Ambiente de Experimental

Nessa seção, descrevemos em detalhes o ambiente experimental utilizado nos testes. Nossa plataforma de Hardware é composta por um *cluster* com onze processadores *Pentium III* distribuídos em máquinas uniprocessadas de 650Mhz e 512MB de memória. Cada máquina executando sistema operacional Linux, *kernel 2.2*. As unidades estão conectadas por uma rede *Fast Ethernet* de 100 Megabits/seg. O *cluster* foi organizado de forma que oito máquinas foram utilizadas com a finalidade de distribuição da aplicação, e as demais são alocadas para o servidor WWW, o servidor de banco de dados e o gerador de requisições. Como mencionado acima, o servidor WWW utilizado é o Apache 1.3.20, executando no modo *standalone*, o que significa que o servidor é iniciado apenas uma vez e serve todas as conexões subsequentes. O SGBD é o MySQL 3.22.25. Para gerar a carga WWW, utilizamos o Webstone (versão 2.0) [ALM 96, MIN], o qual é uma ferramenta de benchmark padrão de indústria para gerar requisições HTTP. Webstone pode ser configurado para gerar padrões de requisições baseado em um arquivo de sessões e requisições. A partir desse arquivo, ele gera um grande número de acessos que seguem o padrão definido e mede o desempenho do servidor a partir do que o cliente observa. A servidor de transações da livraria virtual foi implementado na linguagem C, por se tratar de um ambiente experimental de avaliação de desempenho.

B. Análise de Resultados

A fim de analisar a implementação, descrita na Seção III, instrumentamos a aplicação e coletamos dados dos experimentos nas seguintes configurações:

- Um servidor de transação
- Dois servidores de transação
- Quatro servidores de transação
- Oito servidores de transação

Para cada uma dessas configurações foram utilizados 12 clientes simulados pelo Webstone, sendo que cada servidor de transação dispunha de um número fixo de *threads*, valor escolhido a partir de amostragens de testes feitos anteriormente. Para esses testes, variamos o número de *threads* no servidor de transação e observamos a latência e o *throughput* (requisições/segundo). Esses experimentos caracterizaram o seguinte efeito: o número de requisições do cliente atendidas diminuía a partir da configuração do servidor de transação com 6 *threads*. A princípio isso não era esperado, o que demandou uma análise mais detalhada da implementação realizada. Essa análise demonstrou que a distribuição dos servidores de transação resulta em um *overhead* na sua comunicação, que se torna mais crítico a partir do aumento do número de *threads*.

Por conseguinte, nossos experimentos finais foram executados com 12 clientes, variando o número de servidores de transação, cada um deles executando 6 *threads*. Cada servidor teve sua cache aquecida com consultas de 2000 termos mais populares antes de ser submetido à carga. Em cada experimento realizado, foram coletados informações sobre taxas de acerto e erro das caches dos servidores, latência no tempo de resposta das requisições e número de requisições atendidas por segundo.

Número de Lojas	Home	Search	Select	Browse
1	450,00	62,79	0,65	21,97
2	443,00	87,54	0,62	28,81
4	435,00	20,11	0,66	23,54

Tabela I
TAXAS DE REQUISIÇÕES POR SEGUNDO POR OPERAÇÃO

A Tabela I apresenta o *throughput* das operações de *home*, *search*, *select* e *browse*. Os resultados obtidos nos experimentos com 8 servidores de transação não foram satisfatórios, o que pode ser explicado pela saturação dos mesmos devido ao *overhead* de comunicação [VOI 01]. Os dados demonstram que a aplicação distribuída resulta em ganhos na configuração com 2 servidores de transação. Entretanto, não observa-se escalabilidade contínua, facilmente visto a partir dos resultados obtidos com 4 servidores. Isso serve de motivação para que se aperfeiçoe a estratégia de distribuição no que diz respeito ao mecanismo de comunicação e sincronização dos servidores, que é um trabalho em curso.

A Tabela II fornece a comparação do tempo médio de resposta observado nos casos de *miss* e *hit* para as operações

Número de Lojas	Search	Select	Browse
1	2,08	292,00	106,85
2	2,03	333,89	102,43
4	6,27	293,72	103,64
8	5,45	276,11	100,69

Tabela II
RAZÃO DO TEMPO DE RESPOSTA DAS TAXAS DE MISS PELAS TAXAS DE HIT POR OPERAÇÃO

mais críticas do servidor de comércio eletrônico. Observa-se que para a operação de *search*, a razão entre *miss* e *hit* é relativamente pequena, o que pode ser explicado pela eficiência da implementação corrente dessa operação na loja. Por outro lado, observa-se que a latência média de uma requisição de *search* que corresponde a *miss* aumenta significativamente com 4 e 8 servidores, causando crescimento da razão entre *miss* e *hit*. Isso justifica a necessidade de investigar melhores meios de prover a comunicação entre as lojas.

Já com relação à operação *select*, percebe-se que ganhos ainda maiores na ocorrência de um *hit*, uma vez que essa operação envolve manipulação de uma gama bem maior de informações. A análise isolada das diferentes configurações dos experimentos facilita o entendimento dessa operação. Quando dispomos de um servidor de transação, já verifica-se um ganho substancial no caso de *hit*. Isso retrata o alto custo para acesso ao SGBD para essa operação. À medida que o custo de comunicação se efetiva, os ganhos proporcionados por um *hit* são limitados pela latência observada na comunicação entre lojas.

Por fim, constata-se que a operação de *browse*, por ser mais previsível e de baixo custo, mantêm comportamento constante entre suas taxas de *hit* e *miss*. Não houve ganho na distribuição dessa operação na implementação apresentada, visto que o custo de comunicação adicionado a execução da operação supera os ganhos pretendidos pela estratégia aplicada.

VII. Conclusões e Trabalhos Futuros

Neste artigo, discutiu-se o modelo, a implementação e a validação de uma estratégia de distribuição a fim de aperfeiçoar a escalabilidade de servidores de comércio eletrônico. Essa estratégia, aliada a replicação de dados e paralelização do servidor de comércio eletrônico validadas anteriormente [PER 00], pode representar uma alternativa viável nesse contexto.

Os resultados apresentados demonstram que a distribuição tende a atingir resultados bastante promissores, porém torna-se necessário aprimorar o processo de comunicação inerente a ela. Os experimentos comprovaram que a comunicação entre os servidores de transação tornou-se algo bastante crítico à medida que se aumenta o número de servidores na distribuição. Ainda assim, bons resultados foram observados em configurações com 2 lojas.

Outra inovação, em relação ao trabalho anteriormente apresentado [PER 00], foi o modo de organização dos dados no banco de dados, que proporcionou melhorias bastante significativas para a operação de *search*, anteriormente a mais crítica da aplicação. Isso pôde ser alcançado através da criação de tabelas seguindo o modelo de listas invertidas [BAE 99].

As conclusões adquiridas do trabalho serviram de base para validação do mecanismo de distribuição, comprovando que bons resultados podem ser alcançados. Os trabalhos futuros visam solucionar os desafios identificados, ou seja, aperfeiçoamento do mecanismo de comunicação entre servidores de transação e avaliação de estratégias alternativas de distribuição. Além disso, pode-se adotar diferentes estratégias de distribuição por operação, que em conjunto poderão proporcionar desempenho superior.

Referências

[ALM 96] J. M. Almeida, V. Almeida, and D. J. Yates. Measuring the behavior of a world wide web server. Technical report, Boston University, 1996.

[BAE 99] R. Baeza-Yates and B. Ribeiro-Neto. *Modern Information Retrieval*. Addison Wesley, 1999.

[MEN 98] Daniel A. Menascé and Virgílio A. F. Almeida. *Capacity Planning for Web Performance: metrics, models and methods*. Prentice Hall, 1998.

[MEN 99] D. A. Menascé, V. Almeida, R. Fonseca, and M. A. Mendes. A methodology for workload characterization of e-commerce sites. In *Proc. 1999 ACM Conference on Electronic Commerce*, November 1999.

[MEN 00] D. Menascé, V. Almeida, R. Riedi, F. Peligrinelli, R. Fonseca, and W. Meira Jr. In search of invariants for e-business workloads. In *In Proceedings of the 2nd ACM e-Commerce Conference*, pages 56–65, Minneapolis, MN, October 2000.

[MIN] MindCraft. Webstone. http://www.mindcraft.com/.

[PER 00] A. Pereira, B. Gontijo, T. Cançado, and W. Meira Jr. Replicação de dados em servidores paralelos de comércio eletrônicos. In *Anais do I Workshop em Sistemas Computacionais de Alto Desempenho*, pages 45–50, São Pedro - SP, Outubro 2000.

[PAI 00] G. Paixão, W. Meira Jr., V. Almeida, D. Menascé, and A. Pereira. Design and implementation of a tool for measuring the performance of complex e-commerce sites. In *Proceedings of the 11th International Conference on Modelling Tools and Techniques for Computer and Communication System Performance Evaluation (Performance Tools 2000)*, March 2000.

[ROS 97] Sheldon M. Ross. *Simulation, Statitical Modeling and Decision Science*. Academic Press, 1997.

[VOI 01] T. Voigt and Per Gunningberg. Kernel based control of persistent web server connections. In *Proc. of Performance and Architecture of Web Servers 2001*, MIT Cambridge, MA, USA, June 2001.