Gustavo Machado Campagnani Gama

Distribuição de Serviços de Comércio Eletrônico

Dissertação apresentada ao Curso de Pós-Graduação em Ciência da Computação da Universidade Federal de Minas Gerais, como requisito parcial para a obtenção do grau de Mestre em Ciência da Computação.

Belo Horizonte

05 de Abril de 2002

UNIVERSIDADE FEDERAL DE MINAS GERAIS

FOLHA DE APROVAÇÃO

Distribuição de Serviços de Comércio Eletrônico

GUSTAVO MACHADO CAMPAGNANI GAMA

Dissertação defendida e aprovada pela banca examinadora constituída pelos Senhores:

Prof. WAGNER MEIRA JÚNIOR – Orientador
Departamento de Ciência da Computação – ICEx – UFMG

Profa. CRISTINA DUARTE MURTA
Departamento de Informática – UFPR

Dr. DORGIVAL OLAVO GUEDES NETO
Departamento de Ciência da Computação – ICEx – UFMG

Prof. MÁRCIO LUIZ BUNTE DE CARVALHO
Departamento de Ciência da Computação – ICEx – UFMG

Prof. VIRGÍLIO AUGUSTO FERNANDES ALMEIDA
Departamento de Ciência da Computação – ICEx – UFMG

Belo Horizonte, 05 de abril de 2002.

Resumo

O crescimento do comércio eletrônico nos últimos anos proporcionou um aumento considerável da demanda sobre os sites que proveêm tais serviços. Em consequência, os servidores frequentemente ficam sobrecarregados, diminuindo a qualidade dos serviços oferecidos e aumentando a latência percebida pelos usuários, o que, em último caso, pode comprometer o sucesso do negócio virtual devido à insatisfação dos mesmos.

As soluções tradicionais, como melhoria de equipamentos, são aplicáveis apenas a servidores centralizados e portanto não solucionam uma parte importante do problema que são os atrasos e sobrecarga impostos pelos canais de comunicação. Para contornar este problema, uma estratégia comum é a distribuição do serviço, utilizando-se para isso múltiplos servidores espalhados pela rede em pontos mais próximos aos clientes. O uso de servidores Proxy/Cache e redes de distribuição de conteúdo (CDN's - Content Distribution Network) são dois exemplos consagrados da eficiência da utilização de serviços Web distribuídos. Entretanto, em ambos os casos, os serviços oferecidos envolvem apenas informações estáticas - ou seja, que não são atualizadas com muita frequência. No caso das aplicações de comércio eletrônico, esta premissa é raramente válida, pois os dados manipulados nos servidores são essencialmente dinâmicos - como exemplo podemos citar a disponibilidade de um produto em uma loja virtual ou o preço de um item em um leilão.

A implementação de uma rede de distribuição de serviços que contorne estas limitações depende, entre outros fatores, da elaboração de estratégias de pré-alocação de recursos flexíveis o suficiente para se ajustar às cargas de trabalho extremamente dinâmicas típicas desta gama de aplicações. Nesta dissertação nós propomos e avaliamos duas abordagens fundamentais para esta questão. A primeira baseia-se em um modelo de otimização linear que utiliza tráfego de rede e número de *hops* entre clientes e réplicas como principais métricas de custo. A segunda utiliza heurísticas derivadas dos métodos aplicados a CDNs para resolver isoladamente os problemas de seleção de servidores e distribuição de recursos.

Os experimentos realizados mostraram que, apesar de prover uma solução ótima, o modelo linear apresenta severas restrições com relação ao tempo de execução, dependendo do volume dos dados de entrada. Os métodos aproximados não estão presos a estas limitações e as distribuições geradas apresentam desempenho razoável, com custos de solução variando entre 1,2 e 1,8 em relação aos valores apresentados pela solução ótima. Em compensação, as heurísticas têm maior sensibilidade àS VARIAções de carga, e são mais difíceis de calibrar devido ao grande número de parâmetros que elas utilizam.

Abstract

The growth of E-commerce in the last years has considerably increased the demand on the sites that provide such services. As a consequence, servers are frequently overloaded, reducing the quality of service and raising the latency observed by the users, which, in a worst case scenario, may compromise the success of the virtual business due to the custumers' insatisfaction.

Traditional solutions, such as hardware upgrade, apply just to centralized servers, and therefore do not target an important part of the problem that is the delay inherent to the network channels. A common strategy to work around this problem is to distribute the service, using multiple servers spread across the network at nodes closer to the clients. The use of Proxy/Cache servers and Content Distribution Networks (CDNs) are two well-known examples of efficient distributed Web services. However, in both cases, the services provided handle exclusively static information, i.e., information that is not frequently updated. For e-Commerce applications, this assumption is rarely valid, since data handled by the servers is essentialy dynamic – for instance, a product's availability in a virtual store or an item's retail value in an eletronic auction.

The implementation of a distributed service network that handles such limitations depends, among others, on the development of resource placement strategies that are flexible enough to adjust to the extremely dynamic workloads associated with such genre of applications. In this dissertation, we propose and evaluate two fundamental approaches to address this issue. The first is based on a linear optimization model that uses the network traffic and the number of hops between clients and distributed servers as the main metrics for the cost function. The second uses heuristics derived from the methods employed in CDNs to isolatedly solve the problems of server selection and resource distribution.

The experiments performed show that, although the solutions provided are optimal, the linear model presents severe restrictions concerning the execution time, depending on the volume of input data. The approximate methods are not limited to these constraints and the resource distributions generated show a reasonable performance, with total costs ranging from 20% to 80% worse than the optimal solution. On the other hand, the heuristics have greater sensibility to workload variations, and are harder to calibrate due to the large number of parameters they employ.

Agradecimentos

Primeiramente, quero agradecer a Deus, por ter me concedido a graça de viver uma experiência tão engrandecedora. Tenho certeza de que sem Sua força, esta conquista jamais seria alcançada.

Um obrigado especial ao professor Wagner Meira Jr, que foi para mim um grande orientador, não somente na escola da computação, mas principalmente na escola da vida. Obrigado pela oportunidade, pelos ensinamentos, e pelos sábios conselhos que me permitiram trilhar este caminho, que hoje posso afirmar com certeza ter sido a escolha correta.

Quero agradecer ainda aos outros mestres com quem tive o raro e gratificante privilégio de trabalhar e que se tornaram para mim exemplos de profissionalismo, seriedade e competência: Dorgival Guedes, Virgílio Almeida e Márcio Bunte. Sua participação foi, sem sombra de dúvida, fundamental para o mérito e relevância deste trabalho.

Agradeço também, de modo especial, aos meus familiares: ao meu pai Eduardo, pelo modelo de competência, dedicação, e amor; a minha mãe Fátima, por compartilhar os momentos alegres e difíceis, e sobretudo por me fazer acreditar que nenhum obstáculo é intransponível; e a meu irmão Bruno, pelo companheirismo, cumplicidade, amizade e crescimento conjunto ao longo desses anos.

Gostaria ainda de agradecer a todos os colegas de trabalho e amigos que me acompanharam (e aturaram, mesmo durante as crises) nesta jornada. Aos colegas da Smartprice, que vivenciaram de perto e muitas vezes compartilharam o suor, as aflições e as olheiras. Aos amigos da graduação, do e-Speed e do Sagrado por tornarem esse caminho bem mais divertido de ser percorrido. E ao EJC, pela acolhida, pelo crescimento, e pelos momentos únicos proporcionados.

Por fim, um muito obrigado a todos os familiares e amigos que tanto acreditaram e torceram por mim.

Sumário

Lista de Tabelas

Lista de Figuras

Capítulo 1

Introdução

O fim dos anos 90 foi marcado pela expansão e popularização da World Wide Web. Em particular, observou-se um aumento substancial no número de *websites* relacionados a empresas – desde as grandes e tradicionais corporações da velha economia até novas companhias "ponto-com", conhecidas como *startups*. Uma evidência clara dessa tendência foi o aumento considerável do número de domínios registrados com a extensão '.com', que passou de 18.3% em 1995 para 54.68% em 2000([41]).

Uma consequência imediata do domínio das entidades privadas sobre a Internet foi a expansão dos serviços que utilizavam a Web como interface de acesso. Anteriormente, a Web era utilizada essencialmente para divulgação de conteúdo, que em geral era gerenciado esporadicamente através de intervenção humana. Aos poucos, algumas empresas foram agregando valor aos serviços oferecidos, de modo que se pudesse explorar o largo alcance e o meio eletrônico provido pela Internet para efetuar negociações. Logo observou-se progressivamente o aparecimento e posterior proliferação de lojas eletrônicas, leilões, licitações, barganhas e diversos outros modelos de negócio voltados tanto para o mercado varejista quanto corporativo. Com o tempo, criou-se o conceito de Comércio Eletrônico para designar essa gama de aplicações e todas as propriedades e questões a elas associadas.

Agregada a esta expansão de provedores e serviços, houve naturalmente um significativo aumento da demanda pelos mesmos. A sobrecarga à que os servidores Web e canais de comunicação frequentemente se viam sujeitados logo evidenciou a necessidade de estratégias mais elaboradas para garantia de escalabilidade e qualidade no provimento dos serviços e recursos acessados pelos usuários. Diversas soluções para melhorar o desempenho dos servidores foram desenvolvidas, como a utilização de grupos de servidores cooperativos, ou o desenvolvimento de componentes de *hardware* específicos para determinadas funcionalidades. Essas soluções, entretanto, não tratam de um problema inerente ao modelo de aplicações cliente/servidor utilizado na Internet: as contenções associadas aos canais de interconexão. Para contornar este problema em específico, existem apenas duas soluções utilizadas hoje em dia: servidores Cache e redes de distribuição de conteúdo (CDNs).

Servidores Cache são agentes eletrônicos instalados estrategicamente próximos aos clientes, de forma que o tempo de resposta das requisições WWW seja bem menor. Eles baseiam-se na propriedade da localidade de referência entre os acessos de uma comunidade, intermediando os acessos de um conjunto de usuários e armazenando localmente os objetos mais frequentemente

acessados, de modo que futuras requisições por estes objetos possam ser respondidas sem a necessidade de acessar os servidores Web originais. Os servidores Proxy/Cache compartilham muitas propriedades com os modelos de hierarquia de memória utilizados nos computadores modernos e, consequentemente, diversas técnicas e algoritmos foram herdados e adaptados para este novo contexto. As implementações de servidores Proxy/Cache mais utilizadas hoje em dia são o Squid [43], um projeto de código livre desenvolvido originalmente pela Universidade do Colorado, e o MS-ISA [9], um conjunto de aplicativos para disponibilização e gerência de conteúdo Web desenvolvido pela Microsoft.

Uma vantagem óbvia desta estratégia é o seu baixo custo de implantação e manutenção em relação aos benefícios advindos tanto para os provedores de acesso (ISPs) quanto para os *websites*. Com os custos de armazenamento de dados em mídia magnética cada vez menores, e os custos com conectividade inflacionados devido à crescente demanda, os servidores Proxy/Cache constituem uma estratégia simples, eficiente e barata de diminuir o tráfego de saída dos *backbones* – e consequentemente os custos associados à largura de banda necessária para suprir a demanda de seus clientes. Adicionalmente, os provedores de conteúdo passam a dispor de um mecanismo distribuído para responder às requisições, o que reduz significativamente a carga de seus servidores, o que diminui os seus custos com equipamentos de *hardware* mais sofisticados e ao mesmo tempo melhora a velocidade de acesso e a robustez dos serviços oferecidos.

Uma desvantagem da utilização de servidores Cache, entretanto, é o fato de o controle da replicação do conteúdo ficar exclusivamente sob responsabilidade dos clientes. Com isso, a qualidade do acesso aos *websites* muitas vezes fica sujeita às políticas de armazenamento, invalidação e reposição de páginas adotadas pelos administradores dos provedores de acesso onde o servidor cache está instalado. Um exemplo de como esta restrição pode limitar a utilidade dos caches é a política, adotada por diversos ISPs, de rejeitar a replicação de objetos muito grandes (normalmente acima de um *megabyte*) porque a relação entre número de acessos e o espaço em disco ocupado por estes objetos não compensa a sua replicação. Outra desvantagem significativa da utilização de servidores Cache é a ausência do controle sobre o tráfego efetivo a que os servidores Web estão sujeitos, uma vez que boa parte das requisições passa a ser respondida pelos caches e não fica registrada em seus logs de acesso. Isso dificulta, para um provedor de conteúdo, estimar carga de trabalho real a que ele está sendo submetido.

As **CDNs**, – *Content Distribution Networks*, ou Redes de Distribuição de Conteúdo – surgiram alguns anos após a popularização dos servidores Cache com o propósito específico de contornar as limitações de controle e gerência que estes apresentam. Akamai [1], Digital Island [20] e Inktomi [8] foram algumas companhias pioneiras na disponibilização e comercialização de serviços de distribuição de conteúdo em suas redes proprietárias.

O princípio em que se baseiam as CDNs para melhorar a qualidade de serviço dos servidores Web é essencialmente o mesmo que o dos caches – replicação dos objetos em pontos da rede que estão mais próximos aos clientes – mas com uma diferença fundamental: o controle da replicação dos objetos passa para os provedores de conteúdo. No modelo dos servidores cache, os *websites* atuavam passivamente na distribuição de seu conteúdo, podendo, quando muito, alterar os campos de tempo de validade enviados nas respostas de seus objetos. Além disso, os nodos da rede de servidores cache atuam independentemente, e sua localização e disponibilidade é desconhecida para os servidores Web, o que inviabiliza o controle das instâncias replicadas de

cada objeto. Já nas redes que formam as CDNs, os nodos são posicionados segundo um planejamento prévio, de modo a prover a melhor relação entre o número de usuários atendidos e o tempo de resposta médio de acesso. Em algumas CDNs, chega-se a utilizar uma rede de interconexões privada para garantir a qualidade do serviço de distribuição de informação. Outra facilidade inerente às redes de conteúdo é que os *websites* passam a ter condições de escolher, de acordo com suas necessidades individuais, quantas réplicas serão utilizadas, onde elas estarão situadas, quais objetos serão armazenados em cada uma delas e, mais importante, quando será feita a atualização de cada objeto replicado.

Apesar de as CDNs suprirem diversas deficiências dos servidores Proxy/Cache, ambos apresentam uma limitação que restringe fortemente o âmbito de atuação das duas estratégias: todos os objetos replicados são tratados como unidades atômicas, indivisíveis e não atualizáveis. Em outras palavras, ambas trabalham exclusivamente com **objetos estáticos**. Quando a Web ainda estava em seus primeiros anos, as funcionalidades providas pelos servidores WWW e navegadores limitavam-se a enviar, receber e visualizar documentos no formato hipertexto e imagens. Entretanto, à medida que as necessidades dos usuários e os serviços oferecidos foram se sofisticando, as páginas e os objetos disponibilizados pelos provedores passaram a ser gerados em tempo real, de acordo com a interação entre os usuários e os *websites*. Tornou-se, então, cada vez mais frequente a utilização de linguagens e protocolos para construção de páginas dinamicamente, tanto do lado dos navegadores (através das *applets* Java ou linguagens interpretadas como JavaScript e VBScript) quanto do lado dos servidores (através de *plugins* como CGI e Servlets, ou de trechos de código embutidos no hipertexto, como ASP, JSP ou PHP). As aplicações de Comércio Eletrônico, em especial, alavancaram estas tecnologias por utilizar intensamente as funcionalidades de geração dinâmica de conteúdo para adicionar processamento e inteligência aos serviços oferecidos e ao mesmo tempo garantir as restrições de consistência e segurança de dados que são próprias desta classe de aplicações.

Surge, então, a demanda por um serviço de distribuição de conteúdo que ofereça suporte para replicação de objetos que não são necessariamente auto-contidos, mas constituem unidades fragmentadas de informação que serão utilizadas para construção da resposta final enviada ao usuário. Existem três grandes dificuldades para implementação de um serviço desta natureza. Primeiro, é preciso prover uma interface transparente para acesso aos objetos replicados que suporte as diversas tecnologias de geração de conteúdo utilizadas atualmente. Segundo, a rede de distribuição passa a ser responsável pelo controle de consistência dos objetos replicados, um papel que é normalmente desempenhado por uma ferramenta específica – como um sistema de gerência de banco de dados, por exemplo. E terceiro, a rede de distribuição precisa prover estratégias mais flexíveis e configuráveis para controle do conteúdo replicado, porque os aplicações baseadas em conteúdo dinâmico utilizam informações com menor granularidade e normalmente estão sujeitas a cargas de trabalho com maior variância. O primeiro problema pode, com o tempo, ser minimizado, à medida que os protocolos para padronização de estruturação e acesso a dados (como XML) forem sendo incorporados às tecnologias de geração dinâmica de conteúdo. O segundo problema exige o projeto, implementação e integração de agentes específicos para gerência de dados às funcionalidades oferecidas pelas redes de distribuição atuais. O terceiro problema, que envolve a elaboração e avaliação de métodos para distribuição dos dados utilizados pelas aplicações baseadas em dados dinâmicos, é o objeto de

estudo deste trabalho.

1.1 Trabalhos Correlatos

Apresentamos agora, os principais trabalhos relacionados ao tópicos abordados ao longo da dissertação. Podemos agrupá-los segundo quatro categorias principais: construção remota de respostas dinâmicas remotamente; estratégias para replicação de conteúdo em CDNs; técnicas para controle de consistência e coerência de informação distribuída; e análise e modelagem de topologias de Internet.

O problema de **construção remota de respostas dinâmicas** começou a ser estudado à medida que as ferramentas destinadas à criação de *websites* baseadas em *templates* e linguagens embutidas em hipertexto foram se popularizando – principalmente nas grandes corporações e instituições comerciais – e passaram a representar uma fração significativa dos documentos disponíveis na Web.

Numa primeira tentativa de replicar páginas HTML dinâmicas em servidores cache tradicionais de forma transparente, Cao, Zhang e Beach [7], propuseram converter os objetos dinâmicos em *applets* Java que, ao serem executadas, reconstruiriam o objeto original utilizando um conjunto de parâmetros que representavam as informações dinâmicas da página. Apesar dessa solução ser de aplicação geral, os servidores cache apresentavam uma sobrecarga de processamento quando o número de objetos dinâmicos requisitados aumentava, devido ao alto poder de processamento requerido pelas máquinas virtuais Java sendo executadas. Outro problema dessa solução é que o processo de conversão dos objetos dinâmicos em *applets* não é trivial e, a princípio, seria de responsabilidade dos *websites* repassar aos caches a informação necessária para tal. Essa premissa logicamente restringe fortemente a abrangência da aplicação dessa abordagem porque nem todos os provedores de conteúdo teriam disponibilidade ou mesmo interesse em gerar essas informações.

Outra solução apresentada posteriormente foi a utilização de listas de itens associadas a agentes independentes, instalados nos servidores Cache sob a forma de *plugins*, para gerar o conteúdo dinâmico. Na primeira proposta [31], as listas são utilizadas como banco de dados textual para construção de respostas de uma máquina de busca. Num segundo trabalho [25], já dentro do contexto de comércio eletrônico, as listas são utilizadas como parâmetros de *templates* HTML para responder requisições não críticas (como, por exemplo, busca, navegação e seleção) dos clientes de uma loja eletrônica. Essa abordagem apresenta, entretanto, dois problemas. Primeiramente, há a necessidade de que cada aplicação implemente o agente respectivo que irá construir as respostas. Além disso, apesar de se propor como uma solução voltada para conteúdo dinâmico, o controle de invalidação de dados – no caso, as listas – é feito pelo próprio servidor Cache, e consequentemente está sujeito às mesmas limitações que tornam os servidores cache uma solução inviável para gerenciamento de dados que requeiram suporte transacional ou com atualizações muito frequentes.

Os primeiros trabalhos sobre **replicação de conteúdo em CDNs** surgiram há pouco tempo, à medida que a utilização destes serviços pelos grandes portais WWW se tornou mais frequente. As primeiras estratégias tratavam o problema de replicação completa do conteúdo dos servidores [23, 37] e eram baseados nas estratégias anteriormente propostas para colocação de servi-

dores Cache [28]. Pouco tempo depois, surgiram extensões destes primeiros trabalhos em diferentes direções. Em [38], os autores avaliam os algoritmos originalmente propostos utilizando topologias de redes mais elaboradas e de menor granularidade. Já em [11] e [27], são avaliadas estratégias para replicação e atualização dos objetos individualmente, e não mais do conteúdo completo dos *websites*. Os experimentos realizados, entretanto, utilizam apenas carga sintetizadas analiticamente. Paralelamente, [5] e [29] desenvolveram extensões do modelo tradicional das CDNs que oferecem suporte para distribuição de serviços inteligentes. O primeiro descreve os protocolos e infraestrutura interna necessários para provimento dos serviços, enquanto o segundo define um sistema de regras para configuração dos serviços na rede de distribuidores distribuídos. Ressaltamos a relevância destes dois últimos trabalhos, que são complementares à proposta desta dissertação, pois definem mecanismos que constituem soluções para os dois primeiros problemas relativos à implementação de uma rede de distribuição de serviços com suporte à inteligência e flexibilidade exigidas pelas aplicações de Comércio Eletrônico.

O problema de **controle de consistência e coerência de informação distribuída** foi particularmente explorado dentro do contexto dos sistemas gerentes de banco de dados (popularmente conhecidos como SGBDs ou DBMSs). Por se tratarem, na grande maioria, de aplicações comerciais de alto custo, boa parte dos algoritmos e estratégias utilizadas pelas principais implementações de sistemas de distribuição de dados não são publicados em detalhes. Em [22], entretanto, estão descritas algumas das técnicas utilizadas em uma das últimas versões do SGBD de um dos principais desenvolvedores de aplicações de armazenamento e manipulação de dados atualmente. Outra classe de algoritmos proposta recentemente, denominada de algoritmos epidêmicos, que realiza atualização de conteúdo baseados no compartilhamento dos registros de logs são descritos em [2]. Analiticamente, os algoritmos epidêmicos apresentam baixos custos de comunicação, mantendo todo o tempo a consistência entre todos os nodos do sistema sob o compromisso de um custo um pouco maior quando houver necessidade de cancelar uma transação. No entanto, tanto esta quanto as demais técnicas propostas para os SGBDs tradicionais não podem ser aplicadas ou devem ser adaptadas quando contempladas sob a ótica de um ambiente diversificado como a WWW, uma vez que o tempo de resposta e os canais de comunicação entre os nodos da Internet normalmente apresentam uma grande variabilidade em termos de desempenho e disponibilidade, se comparados com um ambiente controlado como o de um *cluster* de servidores de banco de dados.

Já considerando este novo ambiente, existem alguns trabalhos recentes que apresentam resultados promissores. Em [19], os autores descrevem os algoritmos existentes e propõem um novo método para localização de recursos em aplicações distribuídas em redes heterogêneas. Em [36], um esquema de distribuição baseado em localidade para grupos de servidores é implementado e analisado, alcançando altas taxas de acerto dos caches e um bom balanceamento de carga. Os trabalhos mais recentes no contexto de arquiteturas distribuídas para Internet, entretanto, envolve as aplicações ponto-a-ponto (P2P, ou *peer-to-peer*), como Napster [21] ou GNUtella [13]. A proposta desta nova arquitetura é permitir que qualquer nodo conectado à rede possa tanto recuperar quanto distribuir conteúdo, eliminando assim a diferenciação tradicionalmente feita entre nodos clientes e nodos servidores. Por ser um tópico muito recente e objeto de muitos estudos ainda não concluídos, não existe ainda um consenso sobre quais as melhores estratégias e algoritmos para interconexão de nodos e distribuição dos dados, mas já

existem algumas análises e estudos de caso que podem ser utilizados como referência [4].

Por último, temos os trabalhos acerca da **análise e modelagem de topologias de Internet**. Um dos pré-requisitos para avaliação das estratégias de distribuição propostas foi a geração um grafo representativo da estrutura da rede sobre a qual seria realizada a distribuição. A topologia criada seguiu o desenho utilizado pela equipe do CAIDA (*Cooperative Association for Internet Data Analisys* – Associação cooperativa para análise de dados de Internet) [12] através do projeto SPHIRE. Este projeto utilizou as informações coletadas e disponibilizadas pelo projeto RADB [34], um sumário das interconexões entre todos os sistemas autônomos acessíveis entre os períodos de março de 1998 e janeiro de 2000, para construir um grafo representativo da Internet no nível dos ASs. O CAIDA e o RADB proveêm ainda diversas ferramentas para consulta e análise estatística de informações relacionadas à modelagem topológica da Internet.

Seguindo outra direção, Faloutsos et. al. [10] definem três leis de potência que caracterizam as topologias entre domínios de Internet, através da observação e análise de trechos de registros de *log* de tabelas de roteamento. Como complemento deste trabalho, as causas das referidas leis de potência são estudadas em [30], associando cada uma delas a um conjunto de fatores (a saber, preferência de conectividade, crescimento incremental, distribuição espacial dos nodos e localidade das interconexões). Adicionalmente, é apresentada uma análise comparativa de alguns geradores de topologia de rede [30, 44], mostrando a relevância de cada um dos fatores supracitados para a similaridade e exatidão das topologias de rede sintéticas.

1.2 Objetivos

Utilizando como base o modelo das CDNs, e levando em consideração a demanda dos sistemas atualmente disponíveis por construção dinâmica de respostas e suporte à consistência e coerência de dados, o objetivo desta dissertação é propor e avaliar estratégias para o problema da distribuição de serviços, dentro do contexto das aplicações de Comércio Eletrônico. Assumindo o papel dos provedores de conteúdo, nosso foco é a estrutura de alto nível da rede, abstraindo assim dos problemas relacionados à implementação e manutenção, e ao mesmo tempo permitindo uma visualização mais clara dos elementos que efetivamente influenciam a distribuição de dados do ponto de vista dos servidores Web.

Nossa primeira abordagem foi a elaboração de um modelo de otimização linear baseado em um problema clássico de pesquisa operacional comumente conhecido como “problema de transporte”. Os experimentos realizados utilizando os registros de log de uma loja eletrônica real para representar a carga de trabalho demonstraram que o modelo apresenta um custo computacional excessivamente alto, restringindo sua aplicação apenas a versões restritas do problema.

Para contornar estas limitações, nós propusemos então, um conjunto de heurísticas baseadas na mesma estrutura de custos do modelo linear, que realizassem a distribuição de serviços a um custo computacional bem menor. A primeira desvantagem óbvia das heurísticas é que a solução de distribuição não é ótima, e consequentemente está sujeita a variações na carga de trabalho das diferentes aplicações. Uma outra dificuldade inerente a esta abordagem é o ajuste empírico dos parâmetros utilizados pelas heurísticas, que precisa ser feito individualmente para cada nova aplicação que será distribuída. Apesar disso, os resultados obtidos nos experimentos realizados utilizando a mesma carga de trabalho utilizada para avaliar o modelo linear mostraram que a

redução da complexidade dos algoritmos de solução não impactam tão fortemente no custo das configurações de distribuição propostas pelas heurísticas.

1.2.1 Contribuições

Dentre as contribuições apresentadas por este trabalho, podemos destacar os seguintes pontos:

- Histórico e contextualização dos mecanismos de replicação de dados utilizados atualmente na WWW, e a identificação da carência de suporte a serviços transacionais – como, por exemplo, os oferecidos pelas aplicações de comércio eletrônico – por parte das redes de distribuição de conteúdo convencionais;
- Análise das propriedades e compromissos associados à distribuição dos serviços de comércio eletrônico;
- Estudo qualitativo da carga de trabalho de uma aplicação de comércio eletrônico – especificamente, uma loja eletrônica – direcionado ao problema de distribuição;
- Proposição, implementação e avaliação de um modelo de otimização para as questões de distribuição de serviços relacionadas com a arquitetura da rede (publicado em [15] e [14]);
- Proposição e avaliação de uma série de métodos aproximados e computacionalmente mais eficientes que o modelo de otimização.

1.3 Organização

Esta dissertação está organizada como se segue. No capítulo 2, apresentamos os principais conceitos associados aos serviços de comércio eletrônico, bem como algumas propriedades obtidas através da caracterização da carga utilizada em nossas avaliações experimentais. O capítulo 3 descreve o modelo de otimização proposto como solução para o problema de distribuição de serviços, e apresenta e discute os resultados dos experimentos realizados para avaliação do modelo. O capítulo 4 então, propõe e avalia uma série de soluções aproximadas e computacionalmente mais viáveis para o mesmo problema de distribuição, analisando as vantagens e desvantagens de cada abordagem. Por fim, o capítulo 5 sumariza as contribuições e resultados da dissertação, acrescentando algumas direções para trabalhos futuros.

Capítulo 2

Serviços de Comércio Eletrônico

Neste capítulo apresentamos uma breve discussão acerca das principais características dos serviços de comércio eletrônico, a fim de prover um embasamento teórico que facilite a compreensão das propriedades e compromissos envolvidos nesta classe de aplicações. Em seguida, descrevemos a abordagem adotada para solucionar a questão de distribuição de serviços, justificando algumas das premissas e escolhas tomadas na elaboração das heurísticas e do modelo linear de otimização. Por fim, apresentamos um resumo da caracterização de carga realizada sobre os registros de log da livraria virtual utilizada como estudo de caso, de modo a corroborar e ajudar na compreensão dos resultados obtidos.

Serviços de comércio eletrônico são usualmente providos utilizando os mecanismos tradicionais da WWW, isto é, os clientes requisitam os serviços utilizando o protocolo HTTP e esperam pelas respostas do servidor. Do ponto de vista de implementação, os servidores de comércio eletrônico diferem dos servidores Web tradicionais por demandar três funcionalidades adicionais: suporte a transações; manutenção de estado; e armazenamento de dados persistente e confiável. O suporte transacional assegura que os serviços são executados corretamente, mantendo a validade e a consistência dos dados utilizados. Em geral, a maioria dos servidores de comércio eletrônico oferecem suporte às tradicionais propriedades ACID [26] – Atomicidade, Consistência, Isolamento e Durabilidade. O suporte transacional é importante, por exemplo, para evitar condições de corrida quando mais de uma requisição simultânea for atendida. Este tipo de problema não precisa ser tratado nos servidores tradicionais, uma vez que os clientes não realizam atualizações, apenas operações de leitura. A manutenção de estado, que introduz aos serviços de comércio eletrônico o conceito de sessão, permite que o servidor armazene temporariamente informações relacionadas à interação do usuário com o *site*, como o identificador do usuário após a autenticação, a sequência de páginas acessadas, ou o conjunto de produtos selecionados para compra. Por fim, armazenamento confiável de dados é usualmente provido por sistemas gerentes de banco de dados que executam próximos ao servidor Web. Em alguns casos, os servidores de comércio eletrônicos utilizam, além da robustez e confiabilidade dos SGBDs, todo o suporte transacional oferecido por esses sistemas, de modo a diminuir a alta carga computacional normalmente requerida pelos serviços de comércio eletrônico.

2.1 Propriedades dos Serviços de Comércio Eletrônico

Normalmente, as aplicações de comércio eletrônico definem um conjunto de serviços mais ou menos padronizados que são oferecidos aos usuários. As técnicas utilizadas para melhoria da escalabilidade de uma aplicação em específico estão fortemente ligadas às propriedades intrínsecas dos dados manipulados e das funcionalidades oferecidas pelos serviços que ela oferece. Assim sendo, discutimos a seguir algumas das propriedades mais comuns desses serviços, com o intuito de melhor compreender os compromissos que estão associados à replicação e distribuição dos mesmos.

Começamos por uma análise das características associadas às informações manipuladas pelas aplicações de comércio eletrônico. Foram criadas duas classificações para diferenciação dos dados envolvidos na construção das respostas. A primeira classifica os dados segundo a sua natureza, ou seja, de acordo com a entidade à qual ele está relacionado. Em uma análise generalista, podemos contemplar de imediato três classes segundo este critério:

1. Dados relacionados a produtos/serviços: compreendem não apenas as informações descritivas sobre os itens comercializados, mas também o estoque, preço e outras informações dinâmicas que são modificadas em consequência dos serviços requisitados pelos clientes;

2. Dados relacionados aos usuários: incluem desde nome, endereço, e-mail e demais informações cadastrais como o histórico de compras, a lista de tópicos de interesse e diversas outras informações comumente utilizadas para personalizar os serviços oferecidos;

3. Dados relacionados ao estado do servidor: compreendem as informações transientes, como registros de navegação, logs de transações não completadas, lista de produtos pré-alocados na cesta de compras e o conjunto de lances oferecidos a um produto em um leilão.

Essa classificação assume um papel especialmente importante no contexto da distribuição de serviços porque permite reduzir o conjunto de dados que precisa efetivamente ser replicado. Considere, por exemplo, o cenário em que os clientes de um *website*, cujos serviços foram distribuídos em uma rede de servidores geograficamente distantes, são assinalados unicamente ao servidor mais próximo. Os dados relativos ao estado do servidor são naturalmente associados a um único elemento da rede, e ainda que eventualmente sejam consultados remotamente para administração da rede, não precisam ser copiados para nenhum outro servidor para que os serviços sejam distribuídos. Por sua vez, os dados relativos aos usuários podem ser tratados de maneira idêntica, pois serão acessados exclusivamente pelo servidor ao qual cada cliente foi assinalado. Já os dados relativos aos serviços são naturalmente compartilhados pelos usuários situados em diferentes localidades. Sob este contexto, reduzimos o conteúdo que precisa ser efetivamente distribuído apenas aos dados diretamente associados aos serviços.

A segunda classificação divide os dados em três grupos de acordo com a sua taxa de atualização – daqui em diante também referida por volatilidade: estáticos, pouco voláteis e muito voláteis.

1. Estáticos: incluem o conjunto de todas as páginas estáticas mantidas pelo *site* (exemplos comuns são as páginas de auxílio ao usuário ou informações sobre a empresa/corporação),

e o conjunto de descrições estáticas sobre um item (por exemplo, o autor de um livro, o nome e a lista de músicas de um CD).

2. Baixa Volatilidade: incluem-se aqui os dados cujo conteúdo varia a taxas relativamente baixas, em geral como resultado da interação dos administradores do *site* e não das requisições dos usuários. Como exemplo, podemos citar a lista de produtos de uma loja, o conjunto de produtos em destaque que aparece na página principal, ou o conjunto de revisões enviadas pelos consumidores. A ausência temporária de coerência entre os servidores distribuídos pode ser seguramente tolerada no caso de dados pouco voláteis.

3. Alta Volatilidade: compreende os dados que são modificados muito frequentemente como resultado das interações entre os consumidores e a loja virtual. O exemplo mais comum é a disponibilidade de um item em uma loja virtual, que deve ser atualizada cada vez que uma compra é efetivada.

Esta classificação é especialmente relevante quando agregada a uma análise funcional dos serviços de comércio eletrônico.

A análise de operações ou funcionalidades, providas pelas aplicações de comércio eletrônico na medida que os diferentes serviços são responsáveis por cargas computacionais distintas sob cada um dos componentes estruturais dos servidores. Assim, passa a ser extremamente importante identificar e caracterizar as funcionalidades providas por cada serviço de modo a avaliar apropriadamente o impacto que uma potencial sobrecarga causaria sobre o sistema. Sem perda de generalidade, ilustraremos esta diferenciação de serviços utilizando como exemplo uma livraria virtual, que até o momento constitui o modelo *business-to-consumer* de maior sucesso e que será utilizada como estudo de caso neste trabalho. Cinco funções básicas são normalmente oferecidas:

- Navegação: Permite aos consumidores acessar as informações dos produtos da loja segundo um critério arbitrário previamente definido de filtragem e ordenação, como por exemplo, o autor do livro, a editora ou a sua categoria.

- Busca: Permite aos consumidores explorar e personalizar a seleção dos produtos sobre o qual ele deseja obter informação, através de consultas que podem especificar utilizando seus próprios termos, um conjunto de atributos pré-selecionados pelo *site*, como, por exemplo, o autor, o título ou as palavras chaves de um livro.

- Seleção: A navegação e pesquisa normalmente retornam uma lista de produtos que satisfazem a um determinado critério. Os consumidores podem então obter informações mais detalhadas a respeito de um determinado produto – como o resumo do livro, a opinião de outros leitores ou a listagem de outros livros relacionados – através da seleção deste.

- Adição à cesta: Uma vez que o consumidor decide comprar o produto, ele o insere em sua “cesta de compras”, de maneira que ele possa continuar a navegar pela loja. A manipulação da cesta de compras após a adição permite ainda a inclusão de outros produtos, a remoção dos correntemente incluídos e a alteração da quantidade de cada item.

- Pagamento: Após selecionar todos os produtos, o consumidor faz a confirmação da ordem de compra e envia (ou recebe) as informações necessárias para efetivação do pedido. Diversos métodos de pagamento eletrônico já estão sendo utilizados hoje em dia, mas os mais populares continuam sendo cartão de crédito e boletas bancárias.

Algumas observações importantes a respeito dessas funções são pertinentes. Em geral, as operações de navegação, pesquisa e seleção envolvem dados estáticos ou pouco voláteis, enquanto que adição à cesta de compras e pagamento– as chamadas operações críticas – atuam sobre dados muito voláteis, além de serem as únicas que realizam escrita sobre a massa de dados. É intuitivo, portanto, que as operações não-críticas sejam mais simples de ser distribuídas, como ilustrado em [16]. É interessante observar, ainda, que alguns serviços podem alterar instâncias de todos os tipos de dados. Por exemplo, sempre que um cliente adicionar um produto à sua cesta de compras, o estoque do produto adicionado é decrementado, a página acessada é adicionada ao histórico de navegação do usuário, e a informação de estado – neste caso, a própria cesta de compras – é atualizada.

2.2 Distribuição de Serviços

Apresentamos agora uma discussão sobre a metodologia adotada para abordagem do problema específico de distribuição dos serviços de comércio eletrônico, e como esta metodologia foi utilizada para elaboração das estratégias de solução descritas nos próximos capítulos.

De forma a estruturar e simplificar o problema de distribuição, nós adotamos a abordagem comumente conhecida como *divide-and-conquer* (dividir e conquistar), identificando quatro questões relativamente independentes como fundamentos para uma metodologia de distribuição de serviços de comércio eletrônico: (1) seleção de servidores; (2) distribuição de conteúdo; (3) localização de recursos e (4) balanceamento de carga. Como descrito a seguir, algumas soluções para estas questões já foram propostas sob diferentes contextos, mas não há conhecimento de nenhum trabalho que analise e integre os quatro problemas no ambiente específico dos servidores de comércio eletrônico que, como visto nas seções anteriores, apresenta algumas particularidades significativas em relação a funcionalidades, tipos de dados manipulados e a própria carga de trabalho a que estão sujeitos.

1. **Seleção de servidores**: o primeiro problema que surge ao se tratar de distribuição dos nodos dentro de uma rede extremamente ampla como a Internet atualmente é o da seleção dos nodos mais apropriados para instalação dos servidores replicados. Diversas variáveis entram em cena quando se propõe uma sistematização do processo. Quais os limites mínimo e máximo do número de servidores? Mais servidores levam a um maior grau de paralelismo de acesso, mas também a uma maior complexidade e custo à medida que eles precisam trocar informações com frequência para garantir sincronismo e consistência. Qual a métrica de seleção dos melhores servidores (latência, largura de banda, tráfego, número de conexões, custo, afinidade política)? Dependendo da aplicação, diferentes métricas podem ser utilizadas, em conjunto ou separadamente. Além disso, ao se trabalhar em um ambiente com a extensão e diversidade da Internet, é extremamente difícil obter ou mesmo estimar algumas destas métricas utilizando somente dados de domínio

público.

2. **Distribuição de conteúdo:** assumindo que já está disponível um conjunto de servidores instalados em determinados nodos da rede, passamos ao problema de distribuir apropriadamente os itens que constituem o conteúdo do *site*, entre os nodos onde estão os servidores. Uma solução trivial seria a replicação total dos objetos em todos os nodos servidores, porém, se os dados replicados apresentarem alta volatilidade, o custo de atualização dos nodos replicados pode exceder os ganhos advindos da distribuição dos serviços. Uma segunda solução, um pouco mais elaborada, seria distribuir uniformemente os itens entre os nodos e utilizar um protocolo interno baseado em chamadas remotas para execução dos serviços que não puderem ser satisfeitos localmente. O grande problema deste método reside em uma característica da carga de trabalho dos servidores de comércio eletrônico que, como será visto na seção seguinte, apresenta uma distribuição de popularidade de cauda pesada, ou seja, uma pequena parcela dos produtos mais populares é responsável por uma grande parcela do total de requisições. Com isso, o protocolo baseado em distribuição uniforme está sujeito ao acúmulo de alguns itens muito populares em um dos servidores da rede, que seria logicamente sobrecarregado com um conjunto muito maior de requisições e rapidamente se tornaria o ponto de contenção do sistema. Mais ainda, a mesma distribuição de frequência indica que há um grande número de produtos com relativa popularidade, e que têm de ser apropriadamente distribuídos. Surgem então propostas mais sofisticadas para distribuição dos dados baseadas, por exemplo, na localidade de referência espacial ou temporal entre as requisições. No entanto, o sucesso de qualquer destas estratégias está vinculado à análise da carga de trabalho do *site*, e fica, consequentemente, atrelado ao comportamento dos usuários de cada sistema.

3. **Localização de recursos**: o próximo passo para distribuição dos serviços trata do processo de localização, seleção e propagação de atualizações dos nodos remotos a serem contactados, tanto por parte dos clientes quanto por parte dos servidores que precisarem executar um serviço remotamente. Em suma, quando um nodo qualquer da rede vai enviar uma requisição associada a um determinado item, como ele decide qual servidor contatar? Deve-se decidir baseado em informações dos recursos disponíveis em cada servidor ou simplesmente utilizar uma função simples de assinalamento baseada em topologia de rede, afinidade geográfica ou outro critério semelhante? As soluções Web atuais para este problema são em geral limitadas; no melhor caso, pode-se utilizar resposta de DNS dinâmico que obedeçam a uma função de distribuição segundo um critério arbitrário. Ao contemplarmos, porém, uma configuração de distribuição mais complexa, onde eventualmente tem-se mais de um servidor responsável por um determinado produto, e cada servidor apresenta latência, carga computacional e custo de atualização distintos em relação ao cliente, verificamos a necessidade de algoritmos mais sofisticados e flexíveis para esta tomada de decisão.

4. **Balanceamento de carga**: chegamos por fim ao processo de reajuste interno do sistema que consiste, basicamente, em rearranjar a distribuição dos serviços de acordo com a situação atual de cada um dos nodos. Isto envolve a automatização dos três primeiros passos, transformando-os em um processo interativo, de forma que em intervalos de tempo pré-

determinados, o sistema se rearranje e otimize a sua estrutura de distribuição interna. Por exemplo, ao perceber que um determinado item, inicialmente assinalado ao nodo i, teve, no último intervalo, um aumento muito grande de sua popularidade no nodo j, o sistema migra (ou divide) a responsabilidade do item para este último nodo, de modo a reduzir a quantidade de requisições de serviço remotas. Deve-se notar, entretanto, que o processo de automatização é por si só extremamente complexo, e está sujeito a degradar a performance do sistema se não estiver muito bem ajustado. Por exemplo, se o sistema tenta se reajustar com uma frequência muito alta, o custo de atualizar as tabelas com a distribuição de cada item pode causar um excesso de mensagens, fazendo com que os servidores passem mais tempo se reconfigurando do que respondendo às requisições dos clientes.

As quatro questões acima apresentadas, apesar de poderem ser definidas e analisadas independentemente, estão bastantes interligadas, e em geral as soluções tomadas no nível mais alto refletem ou limitam bastante as soluções disponíveis ao nível inferior. Além disso, alguns fatores como a utilização de diretório central ou a decisão de qual a métrica de desempenho será utilizada para definir a qualidade do serviço oferecido interferem em todos os quatro níveis e devem, portanto, ser estabelecidos antes mesmo de aplicar alguma metodologia.

É importante ainda notar que as duas primeiras questões estão associadas à estrutura e organização da rede de distribuição, enquanto as duas últimas estão vinculadas aos protocolos de controle e manutenção utilizados para implementação e gerenciamento da rede. Como a proposta deste trabalho é a elaboração de estratégias de distribuição voltadas para a arquitetura de alto nível da rede, nós utilizaremos apenas as questões de seleção de servidores e distribuição de conteúdo como fundamento para as estratégias de distribuição apresentadas nos capítulos 3 e 4.

2.3 Caracterização de Carga

Antes de passarmos às estratégias de distribuição de serviços propriamente ditas, apresentamos nesta seção alguns resultados preliminares obtidos através da caracterização da carga de trabalho empregada como estudo de caso para avaliação das estratégias. Esses resultados, assim como algumas referências e conclusões de trabalhos recentes relacionados com a caracterização de carga no contexto de comércio eletrônico, estão sendo antecipadamente apresentados porque proveêm um arcabouço para justificar algumas das opções adotadas na elaboração dos modelos e heurísticas de distribuição propostos.

Recentemente, uma das mais importantes constatações evidenciadas pela análise de cargas de trabalho de servidores WWW foi a da baixa relação existente entre visitas e transações de compra em um *site* de comércio eletrônico. Mais especificamente, menos de 5% das visitas a uma loja *on-line* são convertidas em compras [35]. Números semelhantes foram evidenciados a partir de outros modelos de negócio, como leilões. Isso demonstra que as operações críticas, reponsáveis no alto nível pelo sucesso do negócio virtual por envolver a efetivação de transações comerciais, são responsáveis por uma parcela muito baixa da carga do sistema. Somando-se a estes dados, os resultados apresentados em [3], onde estima-se que o percentual de requisições

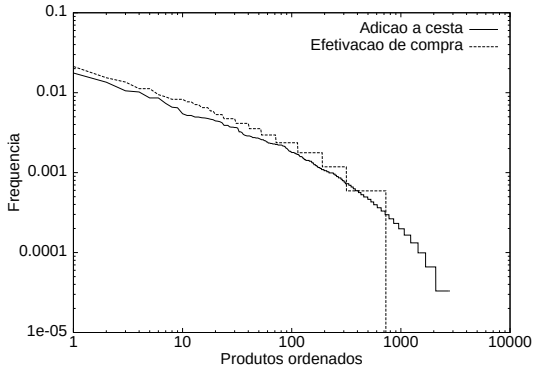

Figura 2.1: Distribuição de frequência de popularidade nas operações de adição à cesta e efetivação de compra

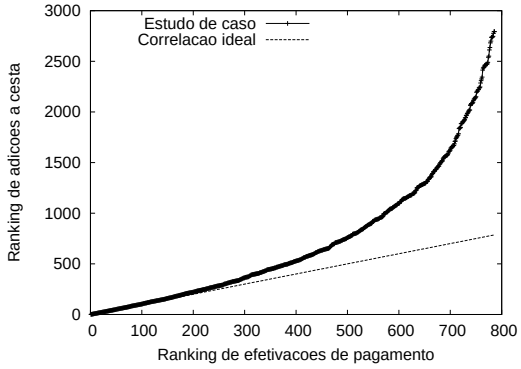

Figura 2.2: Ausência de correlação nas distribuição de frequência de popularidade

originadas por agentes eletrônicos que são enviadas aos servidores é em média 16%, observamos que uma parcela considerável da carga sequer tem potencial para gerar transações, pois não há efetivamente um cliente responsável por elas. Nesse contexto, já surgiram propostas de trabalho [32] que elaboram métodos para identificação do perfil de um usuário a partir do conjunto de requisições que compõem sua sessão, de modo a oferecer um serviço diferenciado e mais eficiente àqueles usuários que apresentarem maiores chances de efetivar negócios e consequentemente aumentar o faturamento do revendedor.

Analisando os registros de log utilizados para simular a carga do trabalho do estudo de caso, constatamos algumas interessantes propriedades relativas à preferência e padrões de acesso temporais e topológicos dos usuários que acessam uma livraria virtual. Nosso primeiro passo na caracterização foi verificar a popularidade dos produtos nas operações críticas, uma vez que constituem nosso foco de estudo. Na figura 2.1, observamos a distribuição de frequência das operações de alocação à cesta e efetivação de compra, ordenadas independentemente em ordem decrescente de acordo com a frequência de acesso da operação, isto é, os pontos mais à esquerda são relativos aos itens mais adicionados à cesta e mais comprados de acordo com a respectiva curva.

Nós imediatamente observamos que a Lei de Zipf [45] se aplica fortemente a ambas as curvas. A Lei de Zipf foi originalmente aplicada no relacionamento entre a posição de uma palavra em um *ranking* de popularidade e a sua frequência de uso. Ela afirma que, dado um

texto literário qualquer, se ordenarmos as palavras pelo número de vezes que cada uma ocorre (que semanticamente definimos como a popularidade da palavra, e denotamos por ρ) pelo valor percentual da sua frequência de uso (que semanticamente representa quantas vezes, em média, uma palavra aparece a cada cem palavras do texto de referência, e será denotado por P), então $P \sim 1/\rho$ (outras constatações da aplicação da Lei de Zipf sobre cargas de trabalho de Comércio Eletrônico podem ser encontradas em [33]). Esta alta variância em termos das operações de alocação e efetivação de compra impactam a estratégia de distribuição ao definir um grupo de itens muito populares, que são requisitados por uma fração significativa dos usuários, demandando uma estratégia de distribuição que divida a responsabilidade por esse grupo de itens, de forma a não sobrecarregar as réplicas que responderem por essas requisições.

Em seguida, nós avaliamos a correlação existente entre a popularidade dos recursos em relação a operações distintas. Para a carga de trabalho utilizada, observamos que a correlação entre as operações críticas é quase inexistente, ou seja, os itens mais adicionados à cesta não necessariamente são os mais comprados. Na figura 2.2 podemos visualizar a relação entre os *rankings* de popularidade das operações de adição de cesta e efetivação de pagamento. Para efeito de comparação, também está representada a curva correspondente à correlação perfeita – ou seja, ambos os *rankings* são idênticos. É notável como a popularidade da operação de adição à cesta difere substancialmente da popularidade da operação de efetivação de compra, especialmente para os produtos que não tão populares. Para efeito de distribuição de serviços, a alta variância em termos da relação alocações/compras confirma a demanda por uma estratégia flexível e consistente já que a frequência das desalocações de items distribuídos pode ser um diferencial no desempenho do sistema.

Passando agora a um estudo dos padrões de acesso dos usuários de nossa carga de trabalho, procuramos a existência de uma relação entre os itens que foram acessados nas operações de adição à cesta utilizando a hora da requisição como referencial, na tentativa de estabelecer um padrão de comportamento sob a perspectiva temporal. Para isso, selecionamos os 50 produtos mais populares segundo três escalas de tempo diferentes – o primeiro dia, os primeiros sete dias e os primeiros quinze dias do registro de log– e traçamos a distribuição de frequência para cada uma, ordenada pela distribuição obtida ao fim do décimo quinto dia (figura 2.3). Observamos, novamente, que não é encontrado um relacionamento claro entre a distribuição das diferentes janelas de tempo, havendo casos em que a frequência do produto dobra no intervalo de quinze dias. Este resultado indica que há também uma alta variância na popularidade dos produtos dentre períodos de tempo relativamente curtos.

Finalmente, nós caracterizamos os padrões de acesso dos usuários em relação a topologia de rede. Para esta análise, foram utilizados alguns dos procedimentos desenvolvidos para construção do grafo usado para avaliação dos resultados do modelo linear, conforme será descrito mais detalhadamente na seção 3. Para esta caracterização, nós agrupamos as requisições em sessões segundo os critérios definidos em [32] de forma a isolar as interações independentes dos diferentes usuários do *site*, e em seguida classificamos as sessões segundo o seu nodo de origem, que neste caso representa o sistema autônomo (AS) onde está situado o usuário. Por fim, calculamos a distribuição de frequência por nodo e ordenamos em ordem decrescente. A representação gráfica dessa distribuição pode ser observada na figura 2.4. Novamente, é marcante a aproximação da curva a uma lei de potência, o que indica a alta variância entre o

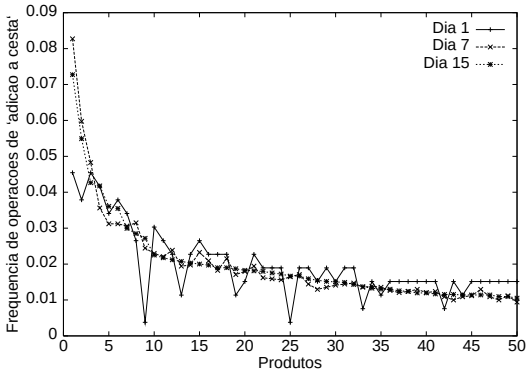

Figura 2.3: Variação temporal da operação de adição à cesta para os 50 produtos mais populares

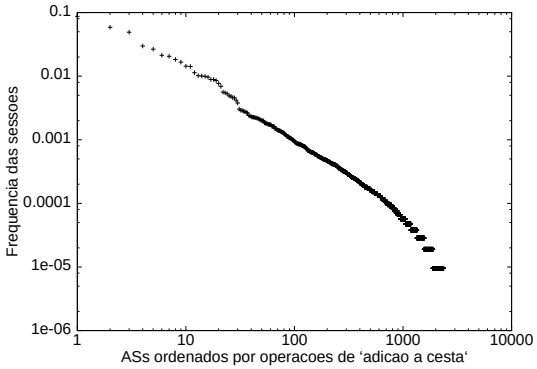

Figura 2.4: Distribuição de frequência do número de sessões originadas por cada AS

número de requisições originadas por cada AS, e define um conjunto de sistemas autônomos – os primeiros da lista ordenada – que são responsáveis por uma fração significativa da carga de trabalho total aplicada ao *website*.

Em síntese, o comportamento dos usuários é muito dinâmico e difícil de prever, tanto em termos da popularidade dos recursos a serem distribuídos, quanto dos padrões de acesso relativos ao tempo e à topologia da rede. Essa ausência de um comportamento bem definido e facilmente modelável evidencia a necessidade da elaboração de estratégias de distribuição que sejam ao mesmo tempo flexíveis, para suportar os diferentes parâmetros de entrada que podem influenciar a distribuição, e dinâmicas, de forma que os recursos possam ser redistribuídos e reajustados à medida que o perfil dos usuários se modifica. No capítulo a seguir, apresentamos um modelo de otimização que utiliza as informações de carga e organização topológica de uma determinada aplicação para determinar uma distribuição ótima de recursos segundo alguns parâmetros de custo.

Capítulo 3

Modelo de Otimização Linear Inteira

Neste capítulo, apresentamos nossa primeira abordagem para desenvolvimento de uma solução de distribuição de serviços que leva em consideração as questões levantadas e descritas no capítulo anterior. Nossa primeira solução é baseada no problema clássico de pesquisa operacional comumente conhecido como problema de transporte ou problema de distribuição de rede. Como este problema não constitui nosso foco de estudo principal, começaremos com uma breve descrição da origem e modelagem tradicional do problema de transporte. Em seguida apresentamos as adaptações e extensões que se fizeram necessárias para adequar o modelo ao contexto da distribuição de serviços de comércio eletrônico. Por fim, apresentamos os resultados obtidos através da avaliação do modelo utilizando a carga de trabalho caracterizada na seção 2.3.

3.1 Problema Clássico de Transporte

O problema de transporte (ou de distribuição) foi um exemplo precoce de otimização de redes lineares e é hoje uma aplicação-padrão em firmas industriais que têm várias fábricas, depósitos, zonas de vendas e vias de distribuição. A utilidade primária do modelo é para planejamento e tomada de decisões estratégicas que envolvem selecionar rotas de transporte de modo a distribuir a produção de várias fábricas a vários depósitos ou pontos terminais.

Na interpretação padrão do modelo, há m pontos de fornecimento com itens disponíveis a serem remetidos a n pontos de demanda. Especificamente, a Fábrica i pode remeter, no máximo, S_i itens, e o Ponto de Demanda j necessita, pelo menos, D_j itens. Os S_i e D_j são fixados com referência a um intervalo de tempo definido ou horizonte de planejamento. O custo de remeter cada unidade da Fábrica i ao Ponto de Demanda j é c_{ij}. O objetivo é escolher, para a duração do horizonte, um plano de rotas que minimize os custos totais de transporte. A descrição matemática do problema clássico de transporte é:

$$\text{minimize} \quad \sum_{i=1}^{m} \sum_{j=1}^{n} c_{ij} x_{ij} \tag{3.1}$$

sujeito a

$$\sum_{j=1}^{n} x_{ij} \leq S_i \quad \text{para } i = 1, 2, ..., m \tag{3.2}$$

$$\sum_{i=1}^{m} x_{ij} \geq Di \quad \text{para } j = 1, 2, ..., n \tag{3.3}$$

$$x_{ij} \geq 0 \quad \text{para todos os } i \text{ e } j \tag{3.4}$$

Na formulação acima, a quantidade não-negativa x_{ij} representa a quantidade de mercadorias remetidas da Fábrica i para o Ponto de Demanda j. Observe que a soma das remessas da Fábrica i a todos os Pontos de Demanda não pode exceder o fornecimento disponível S_i. Do mesmo modo, a soma das remessas ao Ponto de Demanda j vindas de todas as Fábricas deve igualar pelo menos a necessidade de demanda D_j.

Se o custo unitário de produzir um item diferir de fábrica para fábrica, então esse custo é incluído na determinação de c_{ij}. Se, por razões físicas ou econômicas, uma certa fábrica for inacessível a um ponto de demanda particular, então o x_{ij} associado é eliminado, ou, se for mais conveniente, ao c_{ij} correspondente atribui-se um valor arbitrariamente grande. Para simplificar a discussão, supomos $c_{ij} >= 0$ e reescrevemos a restrição 3.3 com igualdades.

Para o modelo possuir uma solução viável é certamente necessário que o fornecimento total seja pelo menos tão grande quanto a demanda total, $\sum_{i=1}^{n} S_i \geq \sum_{j=1}^{n} D_j$. Há um grande número de aplicações nas quais você esperaria que o fornecimento total excedesse a demanda total. Por exemplo, S_i representa algumas vezes a capacidade de produção da Fábrica i durante o horizonte de planejamento, em vez de uma quantidade de produto realmente fabricado para distribuição no início do período. Analisando um modelo de transporte padrão e ideando um algoritmo de otimização, contudo, é conveniente supor que o fornecimento total é igual à demanda total: $\sum_{i=1}^{n} S_i = \sum_{j=1}^{n} D_j$

A seguir, para simplificar ainda mais a descrição matemática, empregamos um dispositivo formal simples que não diminui a generalidade da solução: criamos um destino fictício com uma necessidade de $\sum_i S_i - \sum_j D_j$, e rotulamos esse destino de n-ésimo. Fazemos, então, $c_{in} = 0$ de modo que a interpretação de x_{in} seja a "capacidade de folga na Fábrica i". A soma das capacidades agora é igual à soma das necessidades, e a equação 3.2 pode também ser escrita como uma igualdade. Reescrevemos, então, o problema de transporte como:

$$\text{minimize} \quad \sum_{i=1}^{m} \sum_{j=1}^{n} c_{ij} x_{ij} \tag{3.5}$$

sujeito a

$$\sum_{j=1}^{n} x_{ij} = S_i \quad \text{para } i = 1, 2, ..., m \tag{3.6}$$

$$\sum_{i=1}^{m} x_{ij} = Di \quad \text{para } j = 1, 2, ..., n \tag{3.7}$$

$$x_{ij} \geq 0 \quad \text{para todos os } i \text{ e } j \tag{3.8}$$

3.2 Modelo Linear para Distribuição de Serviços

Passamos agora à descrição do modelo de otimização linear elaborado com o intuito de resolver especificamente as duas primeiras questões apresentadas e discutidas na seção 2.2.

Começaremos pela identificação das entidades que constam no modelo e sua associação ao modelo clássico de transporte acima descrito que serviu como fundamento para nossa extensão.

Para facilitar a elaboração e descrição do problema, iniciaremos com uma versão simplificada em que se considera a distribuição de um único serviço. Essa versão do problema envolve, essencialmente, um conjunto de nodos clientes C que geram as requisições pelo serviço, um conjunto de nodos servidores S que respondem a essas requisições e um mapeamento $c_{ij} \subseteq [C \times S]$, que define o custo de atendimento de uma única requisição para um determinado par de nodos cliente-servidor. Adicionalmente, consideraremos que, para uma janela de tempo pré-definida, temos um vetor de demandas $d[C]$ que define a quantidade de serviço demandada por cada um dos nodos clientes. A associação destas entidades ao modelo de transporte tradicional é trivial: o conjunto de fornecedores é associado pelo conjunto de nodos servidores; o conjunto de consumidores é associado ao conjunto de nodos clientes que originam as requisições; o custo de remessa de uma unidade é associado ao custo de atendimento de uma requisição; e a demanda de cada fornecedor é associada à demanda por serviço de cada nodo cliente.

Nota-se que não houve associação para a capacidade de produção dos fornecedores, que é um parâmetro importante para definição de uma das restrições do modelo convencional. Esta, na verdade, constitui a primeira adaptação do problema tradicional de transporte para o contexto da distribuição de serviços. Essa restrição poderia ser modelada considerando-se que cada nodo servidor tem uma capacidade finita de atendimento de requisições. Sob o ponto de vista minimalista, esta premissa aproximaria-se bastante da prática, uma vez que um servidor de comércio eletrônico apresenta, obviamente contenções de *hardware* e *software* que limitam seu poder de processamento e determinam um limiar de saturação. No capítulo 4 apresentamos uma avaliação de algumas heurísticas que utilizam restrições quanto à capacidade de processamento dos nodos servidores.

Para desenvolvimento do modelo de otimização, entretanto, considerou-se uma outra abordagem. Relembrando a discussão do capítulo 1, a proposta deste trabalho é analisar os compromissos da distribuição dos serviços sob a perspectiva dos clientes, dos contratantes de uma rede de distribuição. Sob esta perspectiva, é razoável considerarmos que a capacidade de processamento de cada nodo servidor não influencia a decisão do cliente, pois na maior parte dos casos ele sequer tem acesso a esta informação. Em geral, a capacidade dos nodos servidores é da preocupação e responsabilidade dos mantenedores da rede de distribuição, e cabe a estes ajustar a infraestrutura da rede de acordo com as demandas impostas por seus clientes.

De forma a adaptar o modelo linear a essa perspectiva, optou-se por incorporar os custos relativos ao reajuste e expansão da rede de distribuição à função de custo. Consideramos que, para o provedor de conteúdo, o custo financeiro associado ao direito de uso de uma rede de distribuição é dependente de dois fatores: quantos (e, eventualmente, quais) servidores serão utilizados, e a carga de trabalho aplicada a cada servidor. A estes dois fatores associaremos, portanto, dois parâmetros de entrada que serão utilizados no modelo: o custo de armazenamento, que independe do serviço replicado e apenas reflete os custos da utilização de mais um nodo da rede de distribuição; e o custo de processamento, que depende da carga total aplicada a cada servidor e representa uma abstração da redução do poder de processamento do total de servidores que compõem a rede de distribuição. Como o outro custo c utilizado na função objetivo

tem uma semântica bem diferente – poderíamos, por exemplo, interpretar como uma medida da qualidade de serviço oferecida ao clientes do provedor de conteúdo, já que representa os custos relativos ao atendimento das requisições – faz-se necessária a utilização de dois fatores que definam uma relação proporcional entre os três custos considerados. E, como se tratam de grandezas cuja importância é dependente de cada provedor de conteúdo (que pode, por exemplo, dar prioridade à qualidade do serviço oferecido em troca de um maior custo financeiro da solução, ou vice-versa), esses fatores precisam ser calibrados empiricamente a cada aplicação do modelo. Adicionamos, então, outros dois parâmetros de entrada, indexados pelo conjunto de nodos servidores, que definem os fatores de proporcionalidade entre o custo de atendimento de uma requisição e os custos de armazenamento (α_S) e os custos de processamento (β_S).

Uma outra pequena observação deve ser ainda ressaltada quanto à adaptação do modelo de otimização. No problema clássico de transporte, o conjunto de nodos fornecedores e o conjunto nodos de demanda são disjuntos, ou seja, nodos fornecedores não geram demanda e nodo de demanda não são capazes de produzir insumo. Dentro do contexto de distribuição de serviços, essa propriedade deve ser descartada, pois, dependendo da granularidade e representação semântica dos nodos que compõem a rede, os nodos que armazenam servidores podem ser geradores de demanda. Uma maneira relativamente simples de incorporar no modelo essa nova propriedade é definir o conjunto de nodos servidores como um subconjunto dos nodos de demanda, e modificar apropriadamente os custos relativos ao atendimento de uma requisição em um mesmo nodo (c_{ii}).

Feitas as devidas considerações e adaptações necessárias para aplicação do modelo sob este novo contexto, podemos passar então, à modelagem matemática do problema. Assim como no problema de transporte, nosso objetivo é minimizar a função de custo total da solução de distribuição, sujeito às restrições de demanda e suprimento.

$$\text{minimize} \quad \sum_{j=1}^{S}\sum_{i=1}^{N} c_{ij} * r_{ij} + \alpha_j * r_{ij} + \beta_j * s_j \tag{3.9}$$

sujeito a:

$$\sum_{j=1}^{S} r_{ij} = d_i \quad \text{para } i = 1, 2, ..., C \tag{3.10}$$

$$\sum_{i=1}^{N} r_{ij} = p_j \quad \text{para } j = 1, 2, ..., S \tag{3.11}$$

$$s_j = (p_j > 0)?1 : 0 \tag{3.12}$$

Além dos parâmetros de entrada previamente discutidos, o modelo acima equacionado apresenta três variáveis: r, p e s. Efetivamente, as variáveis p e s estão relacionadas a r, e não podem ser calculadas dinamicamente; porém, para melhor clareza e mais fácil compreensão da modelagem, optou-se por representá-las explicitamente porque definem papéis semânticos importantes. A matriz r_{ij} é diretamente associada à matriz x_{ij} presente na modelagem do problema de transporte, e representa a solução efetiva do problema de otimização, ou seja, quantas requisições originadas pelo nodo i foram respondidas pelo nodo j. O vetor p_j, obtido a partir da matriz r, representa o total de requisições atendidas pelo nodo servidor j, e a variável binária s, obtida a partir de p, representa a utilização ou não do servidor situado no nodo (s será igual

a um quando pelo menos uma requisição for atendida por aquele nodo). A inclusão de uma variável binária (s) ao modelo merece destaque devido ao aumento de custo computacional que este fator representa. Com o acréscimo de uma variável binária, o modelo linear torna-se um modelo linear inteiro, cuja ordem de complexidade é significativamente maior. Note ainda, que a restrição relativa à limitação das capacidades das fábricas não está presente no modelo; ao invés disso, temos as equações que definem as variáveis p e s, e a função objetivo é acrescida de duas parcelas que representam as extensões do modelo de custo discutidas anteriormente.

Esta versão do modelo é importante porque sua associação com o modelo de otimização utilizado para o problema de transporte é bem direta. Entretanto, para uma solução de distribuição completa, falta acrescentarmos um eixo no espaço de variáveis: os serviços distribuídos. Como visto na seção 2.3, uma das propriedades das cargas de trabalho dos servidores de comércio eletrônico é a independência entre os serviços requisitados. É pouco provável que a solução obtida para um determinado serviço possa ser reaplicada sem uma degradação considerável em termos de desempenho e custo. Uma primeira alternativa seria resolver o modelo independentemente para cada serviço a ser distribuído. Essa alternativa simplista no entanto apresenta duas desvantagens: em primeiro lugar, há redundância nos dados de entrada – pois as informações relativas aos conjuntos de nodos servidores e clientes é a mesma para todos os serviços – o que aumenta o tempo total de obtenção da solução porque a ferramenta que resolverá o modelo de otimização precisaria reprocessar todos estes dados. Em segundo lugar, perde-se a informação das variáveis independentes do serviço distribuído (como s por exemplo) entre as execuções para cada serviço, e consequentemente a solução do conjunto de execuções independentes passa a não ser ótima.

A melhor alternativa é incorporar o conjunto de serviços/recursos R a serem distribuídos ao modelo de otimização. Felizmente, a incorporação dos serviços não modifica a base de nenhuma das relações matemáticas presentes, apenas acrescenta uma dimensão em algumas variáveis e parâmetros utilizados. Por exemplo, o vetor de demandas passa a ter duas dimensões, $i \in C$ e $k \in R$, que respectivamente representam o nodo cliente e o serviço demandado. Similarmente, a matriz r passa a ter três dimensões ($i \in C$, $j \in S$ e $k \in R$), pois agora a semântica de um elemento da matriz passa a ser o número de requisições pelo serviço k originadas no nodo i e respondidas pelo servidor j. Dependendo das propriedades de cada serviço – por exemplo, se considerarmos que os serviços distribuídos são muito diversificados em relação à quantidade de dados trafegada ou aos recursos utilizados para processamento – pode-se optar por adicionar mais uma dimensão aos fatores de custo que dependem do serviço, c e α. O custo de armazenamento fica inalterado porque depende exclusivamente da utilização ou não de um determinado nodo servidor. Reescrevendo as equações após estas alterações, temos então:

$$\text{minimize} \quad \sum_{k=1}^{R}\sum_{j=1}^{S}\sum_{i=1}^{C} c_{ijk} * r_{ijk} + \alpha_{jk} * r_{ijk} + \beta_j * s_j \tag{3.13}$$

sujeito a:

$$\sum_{j=1}^{S} r_{ijk} = \quad d_{ik} \quad \text{para } i = 1, 2, ..., C \text{ e } k = 1, 2, ..., R \tag{3.14}$$

$$\sum_{k=1}^{R}\sum_{i=1}^{N} r_{ijk} = \quad p_j \quad \text{para } j = 1, 2, ..., S \tag{3.15}$$

$$s_j = (p_j > 0)?1 : 0 \tag{3.16}$$

Kangasharju et. al, em [27], desenvolveram um modelo semelhante de distribuição de objetos, também a partir do problema de transporte. Um dos resultados apresentados foi a demonstração de que essa modelagem pode ser reduzida ao problema da mochila [17], conhecido na literatura como pertencente à classe de problemas NP-completo. Este resultado, em conjunto com os experimentos realizados a partir de uma implementação de nosso modelo linear sugere fortemente que a complexidade da solução do modelo no caso médio é exponencial. Consequentemente, para aplicações cujos dados de entrada são muito grandes, o modelo linear pode ser uma solução impraticável, como será discutido no capítulo 4.

3.3 Estudo de Caso

Uma vez elaborado e formulado matematicamente, passamos à aplicação do modelo a um estudo de caso, de forma a verificar e avaliar a correção e eficácia do mesmo. Nosso estudo de caso, conforme mencionado, trata da distribuição de serviços críticos de uma loja virtual, utilizando para isso a carga de trabalho cuja caracterização apresentamos na seção 2.3. A aplicação de um modelo linear consiste essencialmente em definir e estruturar os dados que servirão como parâmetros de entrada para o mesmo. Descreveremos, portanto, o processo de adaptação e aplicação do estudo de caso através da descrição de como foram obtidos cada um dos 6 parâmetros de entrada utilizados por nosso modelo linear: (1) o conjunto de nodos clientes C; (2) o conjunto de nodos servidores potenciais S; (3) o conjunto de recursos R; (4) a matriz de demanda por requisições d; (5) a matriz de custo de transporte por requisição c; e (6) os fatores de proporção entre custos α e β.

3.3.1 Nodos Clientes

O primeiro dado de entrada do modelo constitui o conjunto de nodos geradores de requisições. A definição deste parâmetro, na verdade, pode ser semanticamente interpretada como a elaboração de uma estrutura de dados que represente a infraestrutura da rede onde será realizada a distribuição de serviços. Como nossa aplicação utiliza a Internet como meio para transmissão de dados, precisamos gerar um grafo bidirecional que represente o mais precisamente possível o conjunto de nodos presentes na Internet na época em que os registros de log da aplicação foram coletados, e preferencialmente, as interconexões existentes entre estes nodos, pois, considerando a estrutura tradicionalmente hierárquica da Internet, é bastante provável que as conexões entre os nodos sejam futuramente utilizadas como um importante fator para elaboração da matriz de custos de transporte.

Ao pesquisar, na literatura, métodos para representação topológica da Internet, nós divisamos inicialmente três técnicas que se adaptavam mais apropriadamente aos nossos propósitos: a utilização de ferramentas geradoras de topologia, como BRITE [30] e GT-ITM [44]; sintetização de topologias analiticamente utilizando relações e propriedades matemáticas observadas na configuração topológica real da Internet em diversos períodos distintos [10]; ou, a utilização de informações reais coletadas por agentes eletrônicos do projeto RADB durante os

anos de 1997 a 2000 – o que inclui o período em que foram obtidos os registros de log. Como boa parte dos dados de entrada do estudo de caso derivam de dados de uma aplicação real e os dados coletados pelos agentes certamente representam a real topologia com maior exatidão, optamos pela última alternativa.

Além de definirmos a metodologia que utilizaríamos para gerar o grafo, foi necessário definir a granularidade com que seriam representadas os nodos da rede. Intuitivamente, a opção seria utilizarmos o endereçamento IP para identificação dos nodos, pois o mesmo está diretamente associado à condição necessária à inclusão de um nodo qualquer à Internet e ao mesmo tempo provê mecanismos relativamente simples para identificação das conexões entre nodos – utilizando, por exemplo, as tabelas de rotas dos nodos intermediários ou de ferramentas específicas como *traceroute*. Entretanto, esta alternativa apresenta dois grandes problemas: primeiramente, o número de nodos considerados seria excessivamente grande (potencialmente teríamos até $2^{32} = 4294967296$ endereços IP que, tomados 2 a 2 correspondem a $\frac{2^{32}!}{2!\times(2^{32}-2)!}$ conexões). E, além disso, temos o problema da alta variação de assinalamentos de endereços IP's e rotas de interconexão a que está sujeita uma boa parte da Internet, o que aumenta muito o nível de imprecisão da topologia construída.

A alternativa pela qual optamos foi, então, a utilização dos sistemas autônomos (ou como são comumente referidos, ASs, do *Autonomous Systems*) como unidades representantes dos nodos de rede. Os sistemas autônomos podem ser definidos como unidades administrativas independentes que representam uma comunidade ou um grupo de redes e *hosts* que fazem parte da Internet. A utilização dos sistemas autônomos como entidades fundamentais do nosso grafo topológico apresenta diversas vantagens, tais como a abstração da estrutura de rede interna dos ASs que em geral é responsável por grande parte do mencionado dinamismo de assinalamentos de endereços e rotas, uma vez que as interconexões estabelecidas entre ASs tende a ser muito mais estável devido aos custos e contratos firmados entre as entidades administrativas; a diminuição significativa do número de nodos e interconexões a serem considerados; a facilidade de obtenção das interconexões entre os nodos, pois o projeto RADB já provê tabelas contendo as ligações conhecidas entres os sistemas autônomos em cada período monitorado; e a simplificação dos métodos utilizados para obtenção das rotas utilizadas para transmissão das requisições, uma vez que o roteamento de pacotes entre ASs utiliza tabelas estáticas. Obviamente, a utilização dos ASs apresenta também algumas desvantagens. Perde-se, por exemplo, as diferenças entre os atrasos impostos pelos ASs quando uma requisição é roteada por seus nodos internos, pois toda a complexidade da infraestrutura interna de cada AS é despresada. Além disso, o mapeamento entre os endereços IPs e os respectivos ASs administradores (necessário para ajuste dos registros de log) é realizado com base nos mapas atuais (pois não existe um registro histórico desses mapas, apenas um registro do mapeamento corrente), o que gera algumas requisições inválidas (quando o AS correspondente ao endereço IP do cliente não é encontrado) ou deslocadas (quando o AS responsável por um determinado endereço IP foi modificado).

3.3.2 Nodos servidores

O próximo parâmetro a ser adaptado é o conjunto de nodos candidatos a servidores. Como optou-se por utilizar um nível hierárquico alto para representação dos nodos do grafo topológico,

é bastante razoável supor que os nodos escolhidos como potenciais servidores sejam também geradores de requisições, pois o número e a diversidade de comunidades interconectadas a um determinado sistema autônomo varia bastante. Assim, consideraremos que o conjunto de candidatos a servidores deve ser um subconjunto do conjunto de nodos clientes, ou seja, $S \subseteq C$. Num cenário ótimo em que não houvessem restrições com relação à escolha dos nodos candidatos, poderíamos considerar que, como o modelo linear irá encontrar a melhor solução dentre todas as possibilidades, devemos adicionar o máximo de nodos possíveis ao conjunto S de forma a permitir que o modelo analise o maior número de possibilidades possíveis, o que intuitivamente leva ao caso extremo em que todos os nodos do grafo seriam adicionados ao conjunto de servidores e consequentemente teríamos $S = C$. Entretanto, este caso extremo acarreta custos computacionais excessivamente altos para execução do modelo linear, pois, ainda que a escolha do nível de sistemas autônomos para representar os nodos tenha reduzido substancialmente o número de nodos, temos um total de sistemas autônomos considerável, chegando a quase 6500 AS's. Adicionalmente, devemos considerar que os modelos comerciais de CDNs utilizados hoje em dia disponibilizam um pequeno conjunto de *sites* onde já existe uma infraestrutura de *hardware* pré-instalada e contratos de conectividade acertados para que os clientes escolham onde será replicado seu conteúdo. Sendo assim, a limitação do conjunto de candidatos a apenas algumas dezenas permite, não somente executar o modelo linear em tempo real, como também uma maior similaridade com os modelos contratuais utilizados hoje em dia. Resta, ainda definir o critério de seleção dos nodos que farão parte do conjunto de servidores. Como os provedores dos serviços de CDNs atuais não disponibilizam a lista de *sites* que os mesmos disponibilizam, ou utilizam uma rede proprietária cuja estrutura topológica não é externada, foi necessário definir um critério quantitativo que restringisse os nodos segundo algum princípio semântico. Optamos, então, por utilizar um *ranking* baseado na conectividade decrescente dos nodos e um índice arbitrário e variável usado como limite inferior do ranking para definir o conjunto de candidatos. O raciocínio por detrás desse critério é de que, quanto mais conectado for um nodo, maior o número de requisições que irão passar por ele, e por consequência, sua utilização como intermediário de requisições levaria a uma maior redução dos custos de transporte.

3.3.3 Recursos

Passamos então à definição do conjunto de serviços que serão distribuídos. Nosso estudo de caso trata das operações críticas presentes em uma loja eletrônica, a saber, adição à cesta e confirmação de pagamento. Como a operação de pagamento apresenta níveis de recorrência relativamente baixos (na ordem de centenas em períodos de aproximadamente 15 dias, como visto na seção 2.3), optamos por desconsiderá-la em nosso estudo de caso porque não seria possível realizar análise estatística com as pequenas taxas de amostragem obtidas. Utilizamos portanto a operação de adição à cesta, que é parametrizada pelo item alocado pelo consumidor para futura compra. Como foi observado em nossa caracterização, a variabilidade entre a preferência por diferentes itens é muito alta. Consideraremos, portanto, que cada serviço distribuído será identificado pelo item sendo adicionado à cesta do cliente que executa a requisição. Do ponto de vista do *site* que está distribuindo suas operações, podemos visualizar esta discriminação do

serviço de adição à cesta pelo item adicionado como a distribuição do estoque disponibilizado pela loja virtual, uma vez que a operação de adição à cesta está relacionada à disponibilidade dos produtos que estão sendo oferecidos. Entretanto, assim como acontece com o conjunto de nodos candidatos a servidores, o conjunto de items disponibilizado por uma loja virtual (que tem a mesma cardinalidade do conjunto de serviços a serem distribuídos) pode ser muito grande, sendo frequentemente encontrados bancos de dados de lojas virtuais que contam com mais de 100.000 items cadastrados. Em nosso estudo de caso, foram observados mais de 10.000 livros diferentes adicionados à cesta. Consequentemente, é preciso novamente definir um subconjunto de items que será efetivamente distribuído. Como nosso objetivo é abranger o maior número de requisições possíveis, uma alternativa intuitiva é a utilização da popularidade dos produtos como critério de seleção. Sendo assim, convencionamos um subconjunto de tamanho arbitrário e variável contendo os produtos mais populares como sendo os parâmetros de identificação do conjunto R de recursos a serem distribuídos.

3.3.4 Demandas

O parâmetro seguinte é a matriz de demandas por requisições dos nodos clientes. Como nosso estudo de caso utiliza como base os registros de log do servidor *web* de uma livraria virtual, a construção da matriz de demandas tornou-se bastante simples pois cada entrada do registro de log contém, entre outras informações, o endereço do cliente que originou a requisição, a operação requisitada e os eventuais parâmetros da operação. Assim, obtemos as demandas processando os registros de log em três passos. Primeiro, filtramos apenas as entradas referentes aos serviços que desejamos distribuir, ou seja, selecionamos apenas as requisições de adição à cesta, cujo item adicionado esteja incluído no conjunto de recursos R previamente definido. Em seguida, substituímos o endereço IP do cliente originador da requisição pelo identificador do sistema autônomo responsável por aquele endereço, de modo a associar aquela requisição à um nodo específico do nosso grafo topológico. E finalmente, acumulamos o total de requisições discriminadas pela origem e pelo tipo de serviço.

3.3.5 Custos de transporte

Chegamos, enfim, às funções de custo utilizadas na função objetivo do modelo linear. O primeiro parâmetro de custo, que é utilizado como referência para calibragem dos outros parâmetros de custo, é a matriz de custos de transporte por unidade/requisição c, que é diretamente associada à matriz de custos x do problema de transporte clássico. A primeira premissa adotada ao modelarmos o custo de transporte foi que os serviços a serem distribuídos não influenciam o custo associado ao transporte da requisição, já que sintaticamente as requisições por diferentes serviços podem ser diferenciadas por um simples identificador de produto utilizado como parâmetro para a requisição de adição à cesta. Com isso, a matriz c terá apenas duas dimensões C e S que representarão respectivamente a origem e o destino da requisição.

Para preencher a matriz de custos, é necessário definir a semântica associada ao custo de transporte. Como nosso grafo topológico representa uma rede inter-AS's, podemos associar o custo de transporte entre dois nodos quaisquer aos custos relativos à transmissão dos dados

através dos canais de comunicação que interligam os sistemas autônomos. Considerando ainda que as métricas que definem a performance de um canal independem dos outros canais a que estão ligadas as suas extremidades, podemos considerar que o custo de transmissão entre dois nodos é composto pelos custos acumulados de cada um dos canais atravessados para se chegar do nodo de origem ao nodo de destino. Ou seja, se uma requisição originada pelo nodo A será atendida pelo nodo D, e para ir de A até D é preciso passar pelos nodos B e C então o custo c_{AD} pode ser expresso por $c_{AD} = c_{AB} + c_{BC} + c_{CD}$. Note que os três custos do lado direito da equação anterior podem ser diretamente associados às arestas do grafo de topologia de rede gerado. Podemos com isso preencher toda a matriz de custos de transmissão se, ao construirmos o grafo, associarmos um custo a cada aresta do grafo.

O problema passa a ser portanto, definir uma função de custo de rede para os canais que interligam os AS's que compõem nosso grafo, e em última instância refletem uma imagem do que seria a Internet. Uma primeira opção que foi cogitada como função de custo foi utilizarmos a latência média do canal. Entretanto, ao procurarmos trabalhos e ferramentas que estimassem a latência entre dois nodos quaisquer da Internet (tanto no nível de sistemas autônomos quanto no nível de roteadores e subredes), não encontramos nenhum resultado que fosse satisfatório, ou por serem muito localizados (como registros de log contendo medições a partir de um sistema autônomo específico) ou não abrangentes o suficiente (como em [39], onde os autores estimaram o tempo de resposta utilizando um módulo Javascript que pode ser adicionado de forma transparente a um documento *HTML*, mas realizaram experimentos onde os servidores *Web* estivessem situados em sistemas autônomos distintos). Como a estimativa da latência por alguma metodologia ad-hoc serviria apenas para acrescentar um potencial fator de erro à nossa função de custo, optamos por definir nossa função de custos utilizando apenas o tráfego imposto pelas requisições. E, como as requisições para este estudo de caso foram definidas de forma que o custo de transmissão de requisições seja independente do serviço, o custo associado às arestas foi definido como constante e igual para todas as arestas (isto é, $c_{AB} = c_{BC} = c_{CD} = 1$). Em consequência, o custo c_{AD} pode ser calculado pelo número de arestas do grafo topológico que precisam ser atravessadas para se ir do nodo de origem ao de destino ($c_{AD} = 1 + 1 + 1 = 3$).

3.3.6 Custos de processamento e armazenamento

Finalmente, temos a definição dos dois vetores com o fator de proporcionalidade entre os custos de armazenamento α e o custo de processamento de transações β. Esses fatores são particularmente difíceis de serem estimados sem um embasamento empírico porque representam uma relação entre grandezas abstratas não relacionadas – custos de rede e custos financeiros. Nossa abordagem foi construir os dois vetores utilizando uma função que considerasse tanto a conectividade do nodo quanto uma constante parametrizada utilizada como entrada em tempo de execução do modelo linear. Desta forma, a conectividade representa um diferencial individual do nodo, segundo o raciocínio de que os nodos mais conectados irão cobrar uma taxa maior de utilização; e a constante parametrizada possibilita avaliar diferentes relações entre os custos em tempo de execução, simplificando a calibragem do modelo de acordo com o perfil do contratante da rede de distribuição. A título de exemplo, suponha que o conjunto de nodos candidatos possua três nodos com 5, 8 e 10 interconexões cada um respectivamente, e que em

tempo de execução, foram fornecidas as constantes $c_\alpha = 2$ e $c_\beta = 3$. Os fatores de proporção de cada um destes nodos serão então calculados fazendo $(\#\ interconexoes \times c_{custo})$, o que resultaria nos vetores $\alpha = [10, 16, 20]$ e $\beta = [15, 24, 30]$.

3.4 Avaliação e Resultados

Apresentamos agora os resultados da execução do modelo linear adaptado ao nosso estudo de caso, conforme descrito na seção anterior. Os registros de log utilizados foram coletados no período de agosto de 1999 e compreendem um período de 30 dias, que foram subdivididos em dois trechos de 15 dias. Os primeiros 15 dias – daqui em diante referidos como 'log #1' – foram utilizados para aplicar o modelo linear descrito e avaliar sua eficiência como solução para distribuição de servidores e serviços. Os últimos 15 dias – 'log #2' – foram usados para validar e avaliar a eficácia da solução de distribuição fornecida pelo modelo. Numericamente, o log #1 é composto de 1.056.040 requisições, das quais 30.275 (2,86%) são requisições de adição à cesta, e 747 (0,7%) são requisições de efetivação de compra. O Log #2 é composto de 1.408.706 requisições, sendo 35.238 (2,5%) requisições de adição à cesta e 861 (2,44%) de efetivação de compra.

Antes de definir quais os aspectos avaliados sobre as soluções do modelo, foi preciso definir os limiares utilizados para definição dos subconjuntos S e R. Como previsto pela análise de complexidade do modelo e constatado nas primeiras execuções do modelo, onde os limiares dos dois subconjuntos foram fixados na ordem dos milhares e posteriormente das centenas, o tempo de execução do modelo excedeu os limites de tempo mínimos aceitáveis (mesmo quando limitamos ambos conjuntos a valores relativamente baixos, não foi possível obter uma solução do modelo dentro de um período de quinze dias, o que excede o período de abrangência dos próprios registros de log utilizados). Estes resultados já serviram como indício de que, para obtenção de resultados em tempo real ou próximo disso, necessários para calibrar os fatores das funções de custo associadas aos parâmetros α e β, seria necessário reduzir significativamente o número de elementos dos conjuntos de candidatos e serviços originais. Nos experimentos realizados e descritos abaixo, nós limitamos iniciamos o conjunto de servidores potenciais aos 5 mais conectados, e numa bateria de testes posterior a 10 candidatos. Já o conjunto de serviços foi limitado aos 50 itens mais populares, considerando somente as requisições de adição à cesta. E, para calibragem dos parâmetros de custo proporcionais, variamos os fatores multiplicadores das funções α e β entre 0 e 1, e 0 e 2, respectivamente.

A avaliação de resultados propriamente dita foi dividida em três partes, procurando analisar as soluções de distribuição geradas sob três diferentes perspectivas: ganhos de custo, tempo de validade da solução e desempenho relativo a outros métodos de distribuição de serviços.

Para a primeira avaliação, relativa aos ganhos relativos ao custo, consideramos como parâmetro de referência a configuração sem distribuição de serviços, onde todas as requisições são servidas pelo nodo do servidor original. Neste caso, o custo relacionado ao tráfego é de 21911, e o aumento dos fatores de α e β aos valores máximos considerados resultou em um acréscimo no custo de apenas 17 unidades, o que pode ser explicado pela baixa conectividade do AS onde o servidor original está localizado.

Consideramos em seguida os ganhos obtidos através da utilização da solução de distribuição

$\beta\backslash\alpha$	**0.0**	**0.2**	**0.4**	**0.6**	**0.8**	**1.0**
0.0	11386 (100.0%)	12880 (88.4%)	14316 (83.4%)	15507 (77.0%)	16262 (85.3%)	16770 (87.0%)
0.4	12277 (92.7%)	13771 (82.7%)	15210 (78.5%)	16269 (85.2%)	16869 (82.2%)	17386 (83.9%)
0.8	13168 (86.5%)	14663 (77.7%)	16104 (74.1%)	16876 (82.2%)	17476 (79.3%)	17933 (85.1%)
1.2	14060 (81.0%)	15554 (73.2%)	16863 (77.7%)	17484 (79.3%)	18083 (76.7%)	18470 (82.6%)
1.6	14951 (76.2%)	16445 (69.2%)	17491 (79.3%)	18091 (76.6%)	18687 (81.7%)	19007 (80.3%)
2.0	15798 (75.2%)	17259 (76.0%)	18098 (76.6%)	18698 (74.1%)	19224 (79.4%)	19543 (78.1%)

Tabela 3.1: Resultados utilizando 5 candidatos.

proposta pelo modelo linear. Nos primeiros experimentos realizados foram utilizados apenas 5 nodos candidatos a servidores, mas ainda assim, o tempo de duração médio dos experimentos foi pouco superior a 15 minutos. Os resultados podem ser observados na Tabela 3.1. As colunas da tabela apresentam os valores utilizados para calibrar a constante associada à função de α, e as linhas apresentam os valores utilizados para a constante associada à função β. Podemos ver que a nossa estratégia de distribuição reduz os custos significativamente, de 11 a 48%, em comparação com a configuração não distribuída. Como esperado, os custos relativos ao transporte das requisições é responsável pela maior parte do custo em todos os casos, variando de 69 a 100% do total, quando os valores de α e β são 0.

Para avaliar o impacto de um número maior de candidatos e também da influência da redução significativa do número de candidatos em relação ao total de nodos, nós resolvemos o modelo considerando 10 nodos candidatos. Neste conjunto de experimentos, os tempos médios de execução ultrapassaram 2 horas e 30 minutos, evidenciando a complexidade exponencial do modelo. Os resultados, exibidos na Tabela 3.2 mostram que os ganhos variaram de 15 a 52% em relação à configuração não distribuída, ou seja, as soluções encontradas são quase sempre melhores que as soluções para o modelo com 5 candidatos a uma razão aproximadamente constante de 4%. Isso mostra que o critério de seleção de candidatos apresenta bons resultados quando a razão entre os custos de transmissão de requisições e os custos de armazenamento e processamento de transações for α e β menor (α e β maiores).

A segunda parte de nossa avaliação tratou da análise do impacto da variação temporal das demandas dos items na distribuição dos serviços, analisando o número de itens assinalados a cada candidato através do tempo. Essa análise foi baseada na solução do modelo para a carga de trabalho acumulada para cada um dos 15 dias do registro de log, ou seja, o dia 1 compreende as requisições do primeiro dia, o dia 2 compreende as requisições dos dois primeiros dias e assim por diante. Os resultados podem ser visualizados na figura 3.1, onde pode-se observar que o AS1 é o único nodo para o qual os items são assinalados até o sexto dia, quando os items passam a ser assinalados aos AS's 3561 e 7018. Depois do nono dia, os assinalamentos se tornam mais estáveis, indiciando um limiar da mínima quantidade de informação necessária para aplicação

$\beta\backslash\alpha$	**0.0**	**0.2**	**0.4**	**0.6**	**0.8**	**1.0**
0.0	10613 (100.0%)	12492 (86.3%)	14311 (75.0%)	15592 (76.9%)	16535 (81.3%)	17219 (84.9%)
0.4	11320 (93.8%)	13253 (80.6%)	15084 (71.2%)	16190 (80.3%)	16944 (81.3%)	17545 (83.3%)
0.8	12028 (88.2%)	13979 (75.9%)	15756 (78.1%)	16555 (78.7%)	17268 (80.1%)	18180 (76.8%)
1.2	12736 (83.3%)	14687 (72.3%)	16083 (81.0%)	17202 (86.7%)	17639 (78.1%)	18085 (82.5%)
1.6	13429 (81.7%)	15374 (79.3%)	16716 (73.5%)	17496 (73.8%)	17965 (83.0%)	18407 (81.0%)
2	14044 (78.1%)	15830 (77.0%)	17189 (70.9%)	17779 (79.3%)	18287 (81.5%)	18729 (79.6%)

Tabela 3.2: Resultados utilizando 10 candidatos

de uma estratégia de balanceamento de carga interativa. Esta janela de tempo é representativa pois indica um valor mínimo para aplicação de um método interativo de balanceamento de carga que utilizasse um modelo baseado nas mesmas métricas.

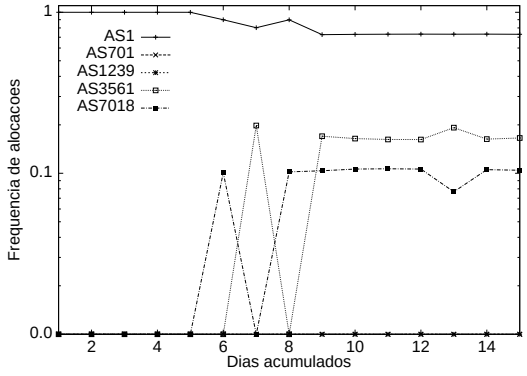

Figura 3.1: Distribuição dos produtos através do tempo.

Por fim, passamos a avaliação da qualidade das soluções propostas por nosso modelo para os períodos imediatamente posteriores ao da coleta dos dados utilizados como entrada de dado para o modelo. O objetivo destes experimentos é verificar a influência da variabilidade da popularidade dos acessos aos serviços sobre nossa solução de distribuição. Para isso, utilizamos o segundo trecho de logs para definir janelas de tempo onde a distribuição de serviços gerada pelo modelo seria aplicada.

Para medir a eficácia da estratégia de distribuição, foi necessário criar uma métrica que indicasse a proximidade entre a solução de distribuição e a carga de trabalho a que o mesmo estivesse submetido. Assim, definimos essa métrica como sendo o número de alocações remotas realizadas por todos os nodos selecionados como servidores, isto é, o número de requisições recebidas mas não satisfeitas localmente por um servidor, necessitando com isso ser redirecionada para um outro servidor que disponibilize o serviço desejado, o que representa um grande impacto no desempenho do sistema porque não apenas aumenta a latência do usuário mas também

o custo transacional. Portanto, para efeito de comparação, consideramos que quanto menor o número de alocações, mais próxima a estratégia proposta está da carga de trabalho real.

De forma a validar a estratégia proposta pelo nosso modelo, nós implementamos três estratégias de distribuição intuitivas que servirão como referência para comparação dos resultados obtidos pelos modelos:

Round-Robin: esta estratégia constrói uma lista dos serviços sendo distribuídos ordenada pela popularidade decrescente dos mesmos. Em seguida, gera uma lista circular dos servidores selecionados e assinala sequencialmente um item a um servidor. Exemplificando: suponha que devemos distribuir um conjunto de N serviços $s_1, s_2, ..., s_N$, cuja ordem no ranking de popularidade decrescente é indicada pelo índice do serviço (s_1 é o mais popular, e s_N o menos popular). Suponha ainda que temos M nodos servidores, $n_1, n_2, ..., n_M$ onde serão distribuídos os serviços, e que o número de serviços a serem distribuídos é maior que o de nodos servidores ($N > M$). A distribuição será feita, então, percorrendo as duas listas simultaneamente e retornando sempre ao início da lista de servidores após acessar seu último elemento: $s_1 \rightarrow n_1, s_2 \rightarrow n_2, ..., s_M \rightarrow n_M, s_{M+1} \rightarrow n_1, ...$, e assim por diante, até que o último elemento da lista de serviços esteja assinalado. Note que não há compartilhamento de responsabilidade entre os servidores, isto é, cada serviço é assinalado exclusivamente a um único servidor. Esta estratégia procura refletir os modelos de distribuição de serviços baseados em resolução de nomes (protocolo DNS), muito utilizados por *sites* que utilizam *clusters* para soluções de escalabilidade [6, 40].

Uniforme por serviço: esta estratégia, bem mais simplista, procura simplesmente balancear a capacidade armazenada em todos os servidores, distribuindo os serviços entre todos eles em frações idênticas. Em outras palavras, cada servidor é responsável por todos os serviços, mas é capaz de responder apenas a uma fração do total de requisições estimado para aquele serviço. Esta fração é igual para todos os servidores e inversamente proporcional ao número de servidores disponíveis. Para exemplificar, suponha o mesmo quadro descrito no item anterior referente à estratégia '*Round-robin*', onde há N serviços a serem distribuídos entre M nodos servidores. Considere ainda que, a cada serviço s_i temos um total de requisições t_i associado, obtido através dos registros de log utilizados como entrada de dados ao gerar o modelo. Então, cada nodo servidor será capaz de responder a t_i/M requisições pelo serviço i. Esta estratégia procura priorizar o balanceamento de carga dos servidores, sem levar em consideração as diferenças de popularidade dos acessos sob a perspectiva da topologia da rede, sendo portanto uma estratégia mais apropriada para *clusters* onde todos os servidores estão localizados em um mesmo ponto geográfico e a distribuição de acessos é realizada por um agente centralizado [36].

Ponderada por carga: esta estratégia utiliza a carga total de requisições (independentes do serviço) a que cada servidor foi submetido como referência para determinação da fração base utilizada para determinar a capacidade de resposta de cada serviço. Similarmente à estratégia 'Uniforme por serviço', todos os serviços são assinalados a todos os servidores, segundo uma fração comum a todos os serviços. O diferencial está no fato que cada servidor tem uma fração base utilizada para assinalar a capacidade de cada serviço individualmente. Esta fração base é obtida dividindo-se o total de requisições a que um

Período	Total de reqs.	Reqs. não atendidas	Redireções			
			Round-Robin	Uniforme por serviço	Ponderada por carga	Modelo Linear
1 dia	7788	937 (12.0%)	6062 (77.8%)	3076 (39.5%)	2577 (33.1%)	2586 (33.2%)
3 dias	8056	198 (14.9%)	6275 (78.0%)	2782 (34.5%)	2183 (27.1%)	2177 (27.0%)
7 dias	6180	669 (10.8%)	4782 (77.4%)	1496 (24.2%)	851 (13.7%)	936 (15.1%)
15 dias	7480	631 (8.4%)	5894 (78.8%)	2681 (35.8%)	2057 (27,5%)	2110 (28.2%)

Tabela 3.3: Resultados da avaliação das estratégias de distribuição

servidor está sujeito, pelo total de requisições que compõem a carga de trabalho. Para exemplificar, considere o exemplo utilizado no item anterior, onde existem N serviços e M nodos a se distribuir. Suponha agora, que a carga de trabalho utilizada como entrada de dados consiste de um total de T_{reqs} requisições e que, após processar os resultados de assinalamento fornecidos pelo modelo, calculamos para cada nodo servidor n_j o total de requisições u_j a que o mesmo foi submetido. Então, determinamos a fração base f_j deste nodo como sendo $f_j = n_j/T_{reqs}$, e calculamos o número de requisições pelo serviço i que este nodo é capaz de responder fazendo $f_j \times t_i$. Esta estratégia não leva em consideração, portanto, as diferenças de popularidade entre os diferentes serviços sob a perspectiva topológica, uma vez que desconsidera a natureza das requisições ao determinar o peso associado a cada nodo utilizado para fazer a distribuição, sendo contraindicada para distribuição de serviços cuja semântica seja facilmente ligada a algum fator regionalista (como, por exemplo, a língua em que é oferecido o serviço)

Cada estratégia foi avaliada considerando diferentes janelas de tempo aplicadas sobre os registros de log #2. Ao todo, quatro janelas de tempo foram empregadas: (1) o primeiro dia; (2) os primeiros 3 dias; (3) a primeira semana; e (4) o registro de log completo, correspondente a 15 dias. A tabela 3.3 mostra os resultados dos experimentos realizados com cada estratégia. Para cada janela de tempo nós utilizamos um fator multiplicativo, explicitado na segunda coluna da tabela, de forma a ajustar o período e o número de requisições utilizados para gerar e avaliar a solução proposta por cada estratégias. A terceira coluna, (*Requisições não atendidas*) apresenta o número de requisições que não puderam ser atendidas localmente pelos servidor distribuído devido à diferença entre as cargas de trabalho representadas pelos dois trechos de log. E as demais colunas mostram o número de redirecionamentos e o valor percentual associado dos redirecionamentos sobre o número total de requisições enviadas aos servidores distribuídos.

A tabela 3.3 mostra que o número de requisições redirecionadas varia significativamente entre as janelas de tempo consideradas, como uma consequência da variação temporal das demandas por diferentes serviços, conforme ilustrado pela figura 2.3. Nós também podemos observar que as estratégias '*Round-robin*' e 'Uniforme por serviço' apresentam resultados muito ruins em todas as janelas de tempo. É também notável a similaridade entre os resultados da estratégia 'Ponderada por carga' e do nosso modelo de otimização, não sendo possível determinar clara-

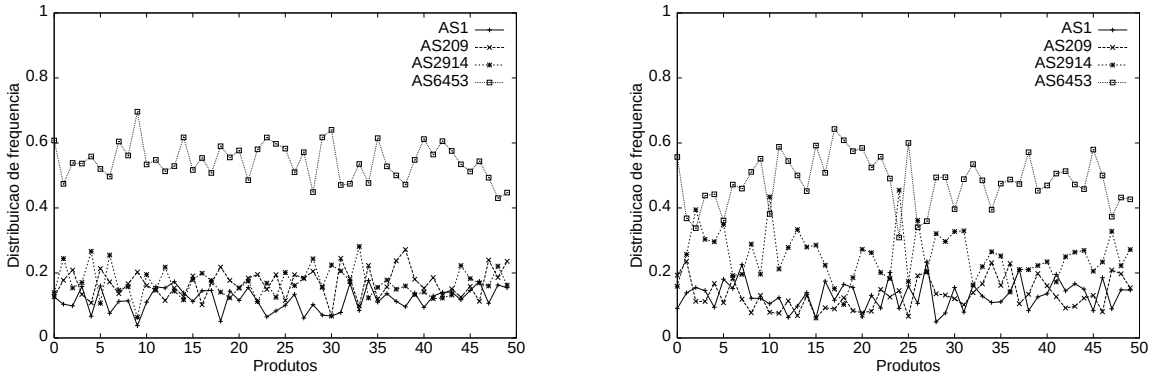

Figura 3.2: Distribuição de frequência para os 50 serviços mais populares nos registros de log #1 (a) e #2 (b)

mente qual das duas é mais efetiva. Para explicar estes dois fenômenos, nós realizamos uma análise adicional sobre os dois registros dois logs utilizados, utilizando como base a solução de distribuição proposta pelo modelo linear. Nós discretizamos a distribuição de frequência do subconjunto de serviços utilizados como parâmetro de entrada do modelo entre cada um dos nodos servidores selecionados e presentes na solução de distribuição utilizada como referência para todas as estratégias. As figuras 3.2(a) e 3.2(b) ilustram as distribuições de frequência para os logs #1 e #2 respectivamente. Como pode ser observado, existe uma alta variância de popularidade entre os diferentes serviços para ambos os casos, o que explica a baixa efetividade das duas primeiras estratégias. A semelhança entre a última estratégia e o modelo de otimização pode ser explicada se analisarmos a diferença entre a variância de demanda por serviços entre servidores nos dois registros de log considerados. Comparando as duas figuras, podemos observar claramente que a variância presente nos registros do log #1 é significativamente menor do que nos registros do log #2. Como exemplo, podemos observar que o AS6453 é responsável por no mínimo 43% da demanda dos 50 items mais populares no log #1, enquanto no log #2 as demandas variam entre 35% e 63%. Essa pequena variância dos registros de log utilizados como entrada de dados do modelo linear resultaram em uma solução de distribuição que se aproximou bastante à estratégia ‘Ponderada por carga’. Esta singularidade, entretanto, não é uma propriedade essencial da carga de trabalho, como pode ser comprovado pelo log #2.

Fazendo uma análise final das propriedades e resultados obtidos pelo modelo, observamos que as soluções geradas por estratégias baseadas em carga são tratadas naturalmente, a exemplo do que ocorreu neste estudo de caso. Além disso, ele provê meios para solucionar apropriadamente os cenários em que a popularidade dos serviços distribuídos apresente alta variabilidade entre os servidores. Os ganhos relativos ao tráfego de rede, em particular, merecem destaque, porque são diretamentes associados à ganhos de tempo de resposta, para os clientes, e redução de custo para os provedores de conteúdo.

Algumas resalvas precisam ser feitas, todavia. Em primeiro lugar, deve-se destacar as restrições relativas ao alto poder computacional necessário exigido pelo modelo, que irão certamente comprometer sua usabilidade à medida que os dados utilizados como entrada aumentam de volume. Esta mesma restrição gera um agravante adicional que é a validade das soluções providas pelo modelo, já que o tempo de obtenção de uma solução pode facilmente suplantar

o tempo em que o comportamento da carga de trabalho é estável o suficiente para aplicação dos resultados. Se considerarmos o caráter extremamente dinâmico de algumas aplicações da Internet atuais, que registram picos de carga com durabilidade de pouco minutos, a natureza estática das soluções propostas pelo modelo inviabiliza sua utilização. Indo um pouco mais além nesta análise, deve-se lembrar que, como em qualquer modelagem, a acurácia do modelo não é o único fator determinante para garantir bons ou maus resultados. A estabilidade da abstração que o modelo representa precisa ser alta, para que as propriedades que o modelo pretende refletir continuem válidas quando da sua aplicação. Os resultados evidenciados pelo estudo de carga, ilustrados pela Figura 3.2 são um exemplo claro destes compromissos.

Capítulo 4

Heurísticas para Distribuição de Serviços

Apesar dos resultados apresentados pelo modelo linear em nosso estudo de caso confirmarem a correção e efetividade de nossa solução, a aplicabilidade da mesma está sujeita a fortes restrições relativas à extensão dos dados de entrada, o que compromete sua utilização como uma ferramenta de uso geral que independa das propriedades e características inerentes à aplicação sendo distribuída. Um exemplo evidente dessas restrições foi a redução da cardinalidade dos conjuntos R (recursos) e S (nodos servidores) no nosso estudo de caso a aproximadamente 1% da cardinalidade original, de forma a reduzir o tempo necessário para solução do modelo de otimização a alguns minutos, permitindo com isso calibrar empiricamente os parâmetros de custo α e β. Especificamente na nossa aplicação, ambas as reduções dos conjuntos eram justificáveis, seja por razões semânticas – o conjunto de servidores disponibilizados pelos provedores de redes de distribuição atuais é normalmente limitado ou pré-definido – seja por análise quantitativa – a popularidade dos serviços distribuídos segue uma distribuição de frequências de cauda pesada, o que faz com que um grupo reduzido dos serviços mais populares seja responsável por um percentual muito grande da carga de trabalho total. Entretanto, é razoável supor que, entre os inúmeros serviços WWW sendo utilizados atualmente, encontremos um bom número de aplicações que utilizem bases de dados de grande porte mas que não permitam a simplificação ou redução dos parâmetros de distribuição a exemplo do que foi realizado com os serviços da livraria virtual de nosso estudo de caso. Para estas aplicações, faz-se necessária, portanto, a elaboração de novas estratégias onde a ordem de complexidade seja menos restrita em relação aos parâmetros de entrada cuja cardinalidade é crítica.

O próximo passo natural em nosso estudo de estratégias para distribuição de serviços é a elaboração e avaliação de métodos que utilizem métodos heurísticos. A característica principal desses métodos é que eles utilizam algoritmos cujo tempo de execução é sensivelmente menor, mas em compensação, não oferecem a garantia de que a solução proposta será ótima – que, para nossa aplicação, implica no menor custo possível. Estas heurísticas, têm, portanto, um compromisso inerente entre a qualidade da solução proposta e a complexidade computacional que precisa ser considerado cuidadosamente. As estratégias de distribuição de recursos descritas na seção 3.4 são alguns exemplos de heurísticas bem simples que utilizamos para avaliar a qualidade das soluções propostas pelo modelo de otimização.

Frequentemente, a escolha da heurística utilizada na solução de um problema depende das restrições de tempo da aplicação e do volume de dados necessários para obtenção da solução.

Entretanto, como são métodos baseados em estimativas, não necessariamente a heurística mais complexa e mais lenta irá prover a melhor solução. Muitas vezes, as características próprias da aplicação ou dos dados de entrada são determinantes para definição da melhor heurística para uma situação em particular. Este seria o caso da aplicação utilizada em nossa avaliação. Como ficou evidenciado pelos resultados apresentados na Tabela 3.3, a distribuição de popularidade da carga de trabalho utilizada como entrada de dados teve um grande impacto na efetividade da solução de distribuição proposta. Similarmente, a estrutura topológica da rede sobre a qual se faz a distribuição tem grande influência no modelo pois, além de modificar diretamente a matriz de custos de transmissão entre os nodos, afeta as razões entre os custos de processamento, armazenamento e transporte, já que a conectividade é um dos parâmetros utilizados na função que determina estas razões. Em consequência desta forte dependência entre as propriedades da aplicação e a qualidade da solução obtida, é importante definir o escopo de validade da mesma e, principalmente, os fatores e eventos que podem levar a modificações no contexto que invalidem ou comprometam o desempenho do sistema distribuído.

Um passo importante, anterior à elaboração das heurísticas que irão substituir o modelo de otimização, é a divisão do problema de distribuição nas duas primeiras questões discutidas na Seção 2.2: seleção de servidores e distribuição de conteúdo. Um fator determinante para o aumento da complexidade do modelo de otimização é a proposta de solução simultânea destas duas questões, pois, formalizando matematicamente, esta premissa se reflete através da adição de uma variável binária (s_j) à equação da função de minimização do modelo. A utilização de variáveis binárias em modelos lineares aumenta significativamente seu custo computacional porque requer a utilização de algoritmos de *branch-and-bound* para varrer todo o espaço de soluções possíveis [18]. De forma a evitar este problema e diminuir a complexidade das heurísticas utilizadas, nós optamos por analisar separadamente cada uma das questões e utilizar uma combinação das duas heurísticas como solução conjugada que serviria para substituição do modelo linear.

4.1 Heurísticas para Seleção de Servidores

Descrevemos nesta seção os algoritmos utilizados para resolver o problema da seleção dos nodos que serão utilizados como servidores pela rede de distribuição. A pertinência e necessidade de obtermos uma solução para este problema é uma questão discutível porque a maior parte das redes de distribuição vigentes atualmente oferecem um conjunto muito reduzido de réplicas, que pode, a princípio, ser avaliado através de intervenção humana. Para fins de completeza, porém, é importante estudarmos também esse problema pois devemos considerar o cenário onde o conjunto de réplicas potenciais contem todos os nodos conhecidos pela rede de distribuição. Esta é, inclusive, uma tendência recentemente observada nas aplicações Web, que têm migrado do paradigma cliente/servidor para o modelo de compartilhamento de informação baseado em redes ponto-a-ponto, em que todos os nodos são simultaneamente clientes e servidores.

Todas as heurísticas propostas utilizam dados derivados de um ou mais parâmetros de entrada do modelo linear como critério para definição dos servidores utilizados. Essa restrição é importante porque, conceitualmente, o modelo de otimização está correto, o que estabelece um

limite superior da informação necessária à obtenção de uma solução ótima para o problema de seleção de servidores. As heurísticas de seleção de servidores são todas simples, quase sempre resumindo-se a utilizar uma métrica base para ordenar ou delimitar os nodos da rede e em seguida eleger como servidores os N primeiros nodos da lista ordenada. A métrica base dos algoritmos de seleção foi utilizada para nomear as heurísticas, de forma a facilitar a associação quando as mesmas forem referenciadas no texto. São elas: (1) seleção por conectividade, (2) seleção pelo total relativo de requisições e (3) seleção pela razão entre custo e total de requisições.

1. **Conectividade:** Esta heurística baseia-se no mesmo princípio utilizado na aplicação do nosso estudo de caso ao modelo de otimização linear (veja Seção 3.3), ao definirmos o parâmetro S que representa o conjunto de nodos candidatos a servidores e um critério para limitar a cardinalidade desse conjunto. A métrica base desta heurística é o número de arestas de cada nodo no grafo topológico, seguindo a premissa de que, quanto maior a conectividade de um nodo, menor a distância média entre este nodo e os demais clientes, o que reduziria os custos com transmissão de requisições da solução de distribuição. O algoritmo de seleção é bem simples, limitando-se a utilizar a métrica base como referência para gerar um *ranking* decrescente e um parâmetro definido em tempo de execução N que representa o número de servidores que deverão ser selecionados. Finalmente, o algoritmo retorna os N primeiros nodos do *ranking* como sendo os servidores escolhidos.

2. **Total de Requisições:** Esta heurística utiliza o mesmo algoritmo de seleção implementado na heurística baseada em conectividade (N primeiros elementos da lista de nodos em ordem decrescente) utilizando porém outro critério como função de comparação para ordenação. Enquanto a heurística baseada em conectividade utiliza a estrutura topológica da rede de distribuição para gerar a métrica base, esta heurística utiliza a carga de trabalho da aplicação – quantitativamente representada pelo total de requisições originadas por cada nodo. Com isso, procura-se explorar um dos resultados obtidos em nossa caracterização de carga, onde observamos uma distribuição de cauda pesada na carga de trabalho gerada pelos diferentes nodos da rede (Figura 2.4). O princípio em que se baseia a heurística é o de que uma fração significativa das requisições passariam a ser respondidas localmente, já que seriam originadas dos nodos escolhidos como réplicas e, consequentemente, eliminaríamos o custo de transporte relativo a essa fração das requisições.

3. **Razão entre Custo e Total de Requisições:** Assim como as duas primeiras, esta heurística seleciona os nodos servidores utilizando uma lista ordenada e um parâmetro especificando o número de réplicas selecionadas. A modificação novamente está na métrica utilizada para ordenação dos nodos. Nesta heurística, procuramos mimetizar o comportamento do modelo de otimização criando uma função de ordenação que é baseada na função objetivo definida no modelo. Nós definimos a métrica base como sendo a razão entre o custo de processamento e transmissão dos nodos clientes e o número de requisições geradas pelos mesmos. Desta forma, os primeiros nodos da lista ordenada serão aqueles com baixo custo e grande número de requisições. É importante notar que o custo depende da distância dos clientes, do número de requisições que cada um gera, e do fator de custo α relativo à replica em questão. Utilizando as variáveis definidas nas equações 3.13, 3.14

e 3.15, podemos expressar a métrica base como $(\sum_{i=1}^{C} c_{ir} * d_i + \alpha_r * p_r)/p_r^{\gamma}$ onde r é o índice do nodo candidato à réplica e γ é o expoente (passado em tempo de execução) que determina a proporção entre a função de custo e o número de requisições, determinado em tempo de execução. Quanto maior o valor de γ, maior o peso das requisições na determinação das melhores réplicas, e vice-versa.

4.2 Heurísticas para Distribuição de Conteúdo

A segunda questão que iremos atacar em nosso estudo trata da distribuição de serviços dentre um conjunto de servidores previamente selecionados. Desde o surgimento das redes de distribuição de conteúdo convencionais, voltadas para conteúdo estático, este problema tem sido alvo de diversos trabalhos na área de arquiteturas de rede e sistemas distribuídos. Em [27] especificamente, os autores abordam a questão de uma maneira bem similar à nossa, e propõem algumas heurísticas derivadas das soluções utilizadas para servidores Cache [24]. Os resultados obtidos neste trabalho através de simulações realizadas com cargas de trabalho geradas sinteticamente mostram que três das heurísticas – por popularidade, gulosa simples, e gulosa global – apresentam bons resultados em relação à solução ótima.

Uma medida natural foi então utilizar estas três heurísticas como base das estratégias desenvolvidas para a questão de distribuição de conteúdo. Entretanto, a implementação das heurísticas segundo o modelo de avaliação baseado em registros de log e dentro de um contexto diferente – no caso, distribuição de serviços críticos, e não apenas objetos estáticos – tornou necessárias algumas adaptações, tanto por parte do modelo quanto por parte das heurísticas. A principal modificação no modelo foi a adição de uma limitação na capacidade de armazenamento das réplicas. Todas as três heurísticas utilizam essa restrição para determinar a condição de término do algoritmo que efetivamente distribui os objetos. Nós implementamos essa restrição segundo dois critérios diferentes para determinar o limite de capacidade de um servidor: número de requisições e número de serviços armazenados. O primeiro critério procura modelar uma restrição relativa ao poder de processamento de um servidor, determinando o número máximo de requisições que esse servidor poderia responder considerando o período de avaliação. O segundo modela uma restrição quanto ao espaço de armazenamento das réplicas, seguindo a suposição de que cada serviço distinto replicado requer alocação de espaço adicional no servidor.

A heurística baseada em popularidade não necessitou de nenhuma modificação para ser adaptada ao nosso modelo de avaliação. A heurística gulosa-global, por outro lado, teve de ser descartada, mesmo após sua adaptação ao modelo, porque uma das modificações necessárias – a substituição das taxas de requisições pelas demandas absolutas obtidas através dos registros de log – invalidava a propriedade que conferia contexto global à heurística o que, em última instância, reduzia a heurística gulosa global à heurística gulosa simples. Por sua vez, a heurística gulosa foi adaptada, modificando-se a métrica de aproveitamento utilizada localmente em cada réplica para seleção dos serviços. Como compensação pela inutilização da heurística gulosa-global, nós propusemos uma adaptação da heurística gulosa simples, migrando a entidade usada como referência para distribuição dos recursos das réplicas para os clientes. A heurística gulosa original passou a ser referenciada por *gulosa por réplicas*, e a nova heurística proposta como

gulosa por clientes. A seguir, descrevemos a semântica e algoritmos utilizados por cada uma das três heurísticas implementadas, ressaltando as diferenças de cada uma.

1. **Popularidade:** O primeiro passo do algoritmo baseado em popularidade é determinar o conjunto de clientes associado a cada réplica. Para isso, construímos a árvore geradora mínima [42] a partir dos nodos clientes e definimos como réplica associada aquela com o menor nível na árvore. Em seguida, em cada réplica, calculamos a popularidade acumulada de cada serviço somando as requisições originadas nos clientes que foram associados àquela réplica. Reutilizando as definições das equações 3.13, 3.14 e 3.15, podemos equacionar a função de popularidade P de uma réplica r como $P_{rk} = \sum_{i=1}^{C_r} d_{ik} \text{ para } k = 1, 2, ..., R$. Os serviços são então ordenados decrescentemente segundo a função de popularidade, e em seguida alocados até que o limite de capacidade do servidor seja atingido.

2. **Gulosa por Réplicas:** Em alto nível, o algoritmo desta heurística comporta-se de forma idêntica ao da heurística de popularidade, associando inicialmente pares cliente/serviço a uma réplica a partir da árvore geradora mínima construída para cada cliente, ordenando em seguida os serviços para cada réplica independentemente e por fim alocando a cada réplica os primeiros serviços da sua respectiva lista até que a restrição de capacidade seja atingida. A diferença entre as duas heurísticas está essencialmente na métrica utilizada para ordenação da lista de serviços. Ao invés da função de popularidade, utilizamos a razão ponderada entre custo e demanda, analogamente à função definida na terceira heurística de seleção de réplicas, com a diferença que, neste caso, a função é discretizada por serviço e o número de requisições utilizado é relativo apenas aos clientes de cada réplica individualmente. Matematicamente podemos expressar a função custo por demanda CD de uma réplica r como $CD_{rk} = (\sum_{i=1}^{C_r} c_{ir} * d_{ik} + \alpha_r * p_r)/p^{\gamma} \text{ para } i = 1, 2, ..., R$.

3. **Gulosa por Clientes:** A principal diferença entre esta heurística e a outra heurística gulosa é o algoritmo de associação de clientes às réplicas. Enquanto a anterior utiliza uma abordagem orientada pelos nodos servidores – cada réplica seleciona seus clientes a partir das respectivas árvores geradoras mínimas – esta heurística utiliza uma abordagem orientada pelos clientes. Cada cliente estima a distância entre todas as réplicas e seleciona a mais próxima. Em seguida, a razão custo/demanda entre o cliente e a réplica selecionada é calculada para todos os serviços requisitados pelo cliente em questão. Por fim, cada réplica seleciona os serviços que serão armazenados com base na razão custo/demanda acumulada por todos os clientes que a selecionaram, até atingir a restrição de capacidade do servidor. Em teoria, esta heurística diminui os custos relativos à transmissão em relação a outra heurística gulosa, devido à redução da distância média entre clientes e réplicas, já que o novo algoritmo de assinalamento aumenta substancialmente o número de potenciais associações entre clientes e réplicas analisadas.

Vale ressaltar que cada heurística procura definir uma solução que prioriza um ou mais critérios individualmente, em detrimento de outros. Dependendo da estrutura topológica e da carga de trabalho de uma determinada aplicação, as heurísticas se aproximarão mais ou menos da configuração que essas variáveis determinam. Além disso, as heurísticas de distribuição dependem

de um conjunto de réplicas já pré-estabelecidas e, obviamente, a variação desses elementos também influi diretamente no esquema de distribuição de cada heurística. Consequentemente, devemos sempre procurar analisar o par de heurísticas de seleção/distribuição, e não apenas as heurísticas de distribuição individualmente. Na avaliação das heurísticas, descritas na seção seguinte, mostramos os resultados obtidos através da combinação exaustiva dos métodos propostos.

4.3 Avaliação

Passamos agora à avaliação das heurísticas descritas nas seções anteriores. Assim como na avaliação do modelo linear, nós utilizamos como carga de trabalho os registros de log #1 e #2, possibilitando com isso verificar também a efetividade das heurísticas em comparação ao modelo linear. Como as heurísticas não dispõem do embasamento matemático, como é o caso do modelo de otimização, e estão normalmente sujeitas ao ajuste empírico de uma série de parâmetros, nós implementamos uma infra-estrutura de avaliação baseada em um simulador, que permitisse facilmente variar os parâmetros e dados de entrada e ao mesmo tempo comparar o desempenho das configurações. Para essa última funcionalidade, nós definimos um conjunto de métricas baseadas nas funções de custo utilizadas no modelo linear para quantificar as diferenças entre as heurísticas sob diferentes aspectos. As métricas utilizadas foram:

- Redirecionamentos: consiste no total de requisições que não puderam ser atendidas localmente em uma réplica e consequentemente precisaram ser redirecionadas para uma outra réplica (ou mesmo para o servidor original).

- Requisições não atendidas: é a soma de 'Redirecionamentos' com as requisições excedentes em cada réplica. Para o caso de uma solução ótima, como a do modelo linear, representa a diferença entre a carga de trabalho utilizada para gerar a distribuição e a carga de trabalho utilizada para avaliar o modelo.

- Custo de transmissão: calculado de maneira idêntica à função de custos utilizada no modelo linear (número de arestas entre o cliente e a réplica, ponderado pelo número de requisições por cada serviço) utilizando, porém, a carga de trabalho do período de avaliação.

- Custo de processamento de transações: associado diretamente ao modelo linear, utiliza o número de requisições dirigidas a cada réplica em conjunto com o parâmetro de proporção de custos α.

- Custo de armazenamento: também associado diretamente ao modelo linear. Utiliza a conectividade dos nodos em conjunto com o fator de proporção β para determinar o custo individual de cada réplica. Depende unicamente da heurística de seleção, pois apenas a utilização ou não de uma réplica influi no custo total de armazenamento.

- Custo de consistência: representa o custo de manutenção de coerência de dados devido ao protocolo que realiza as alocações remotamente quando uma determinada réplica não

pode satisfazer uma requisição localmente. É calculado a partir dos redirecionamentos registrados em uma determinada réplica e da distância entre esta réplica e as demais. Quantitativamente não deve ser comparado com as outras métricas de custo porque o tráfego associado às mensagens de controle é bem menor do que o de requisições.

- Custo de atualização por item: representa o custo de atualização de todos os itens armazenados, utilizando a distância entre os nodos servidores e o nodo onde estaria situado o servidor original. Em termos práticos, representa a modificação de um dado pouco volátil, como, por exemplo, o preço de um produto. Independe da carga de trabalho utilizada na avaliação.

- Custo de atualização por requisição: similar à métrica anterior, representa o custo de atualização de cada recurso individualmente acessível por uma requisição. É calculado da mesma forma que a métrica anterior, utilizando-se porém o número alocações em cada servidor como peso do respectivo ítem replicado. Pode ser interpretado como a atualização decorrente de uma modificação de um dado muito volátil – como uma atualização no estoque – ocorrida no servidor original do *site*.

Nota-se que as métricas são afetadas pelas propriedades da distribuição de serviços em diferentes intensidades. Entender como ocorre esta diferenciação é essencial para elaboração dos experimentos que irão averiguar as modificações no comportamento das heurísticas segundo diferentes configurações. Por exemplo, seria inútil monitorar os custos de atualização em um experimento em que variássemos o parâmetro referente ao custo de armazenamento (β) porque são entidades independentes. Por outro lado, a variação do parâmetro com o número de réplicas que devem ser retornadas pela heurística de seleção certamente influenciará fortemente esta mesma métrica.

Passamos então à estrutura desenvolvida para calibrar empiricamente e avaliar as configurações de distribuição de serviços providas pelas heurísticas descritas anteriormente. Nós dividimos o processo de simulação em duas fases independentes, seguindo o modelo baseado em treinamento e execução frequentemente utilizado na avaliação de sistemas. A primeira fase, que corresponde à solução do modelo de otimização, utiliza uma configuração de heurísticas e parâmetros em conjunto com um trecho de logs utilizados como entrada para gerar a distribuição de serviços. E a segunda fase utiliza um outro registro de logs para simular uma carga de trabalho aplicada sobre um sistema distribuído configurado de acordo com as informações geradas pela primeira fase.

Como as cargas de trabalho utilizadas para representar as demandas do período de treinamento e de avaliação são baseadas em registros de logs, foi necessário aplicar filtros restritivos sobre os trechos do período de avaliação para que apenas os dados conhecidos durante a execução dos algoritmos de distribuição fossem considerados. Com isso, na fase de simulação, são descartadas todas as requisições que foram originadas de nodos que não haviam sido incluídos no grafo original ou originaram um serviço até então desconhecido. Uma outra adaptação implementada no simulador foi a definição de uma janela de requisições que possibilitou a utilização de trechos de log independente da quantidade de registros presentes ou do período de tempo contemplado. Assim, passa a ser possível calibrar as heurísticas utilizando registros referentes a períodos de tempo mais curtos do que os registros utilizados para a fase de avali-

ação. O tamanho da janela é determinado a partir da capacidade acumulada de todas as réplicas, de forma que as métricas vinculadas à carga de trabalho sejam normalizadas para futura comparação. O simulador move a janela pelo registro de log que simula a demanda dos clientes, calculando a cada deslocamento o total de redirecionamentos e requisições não atendidas. Ao fim da simulação, o valor final das métricas é calculado como a média dos valores instantâneos, criando com isso um compromisso entre o tamanho da janela e o total de requisições válidas do log: se a janela é proporcionalmente muito pequena, pode haver grandes variações nos padrões de popularidade e distribuição dos acessos que causarão um aumento irreal das duas métricas consideradas; por outro lado, se ajustamos o registro de log à janela, corremos o risco de perder características e padrões de comportamento dos usuários da aplicação.

4.3.1 Resultados

Passamos enfim, à descrição dos experimentos realizados para analisar o comportamento e eficiência das heurísticas elaboradas. Conforme mencionado, a avaliação de métodos é muito mais complexa porque está mais sujeita às variações da carga de trabalho e calibração de parâmetros empiricamente. Tendo em vista essas dificuldades inerentes, nós procuramos estruturar nossos experimentos segundo três objetivos bem definidos: (1) verificar a sensitividade de cada heurística a variações dos seus respectivos parâmetros; (2) verificar a sensitividade de cada heurística a variações da carga de trabalho a que o sistema será submetido; e (3) comparar o desempenho das heurísticas em relação ao modelo de otimização linear.

Para avaliar a sensitividade das heurísticas em relação aos parâmetros usados, nós optamos por avaliar cada par de heurísticas (seleção de servidores/distribuição de conteúdo) de maneira independente porque o número de combinações que precisam ser avaliadas, se realizarmos a avaliação simultânea, é excessivamente grande (utilizando 5 instâncias distintas para cada variável considerada, obtivemos mais de 1000 configurações a serem analisadas), o que dificulta ainda mais a identificação das propriedades de cada heurística. Assim sendo, nós optamos por reduzir o número de variáveis simultaneamente consideradas fixando alternadamente uma parte do algoritmo. Primeiro, utilizamos uma configuração estável para a heurística de distribuição e variamos os parâmetros das heurísticas de seleção e, em seguida, variamos os parâmetros das heurísticas de distribuição e utilizamos uma mesma configuração para a heurística de seleção. A configuração estável escolhida em ambos os casos foi a mais independente de ajustes empíricos (connectividade e popularidade, para as heurísticas de seleção e distribuição respectivamente), procurando com isso reduzir os fatores que potencialmente influenciariam os resultados.

O primeiro parâmetro avaliado foi o número de servidores, utilizado pelas heurísticas de seleção. Variando o número de réplicas selecionadas entre 3 e 20 réplicas, nós verificamos como as métricas diretamente associadas ao número e posicionamento das nodos selecionados eram afetadas. As Figuras 4.1 e 4.2 mostram os resultados das três heurísticas de seleção relativas às métricas de custo de armazenamento, custo de tráfego, custo de atualização por recurso e custo de atualização por item, respectivamente. Podemos observar, logo à primeira vista, que não existe claramente uma heurística com melhor desempenho. Além das variações observadas no escopo de uma mesma métrica (nos custos de atualização, por exemplo, as três heurísticas aparecem eventualmente com melhor desempenho, dependendo do valor do número de réplicas

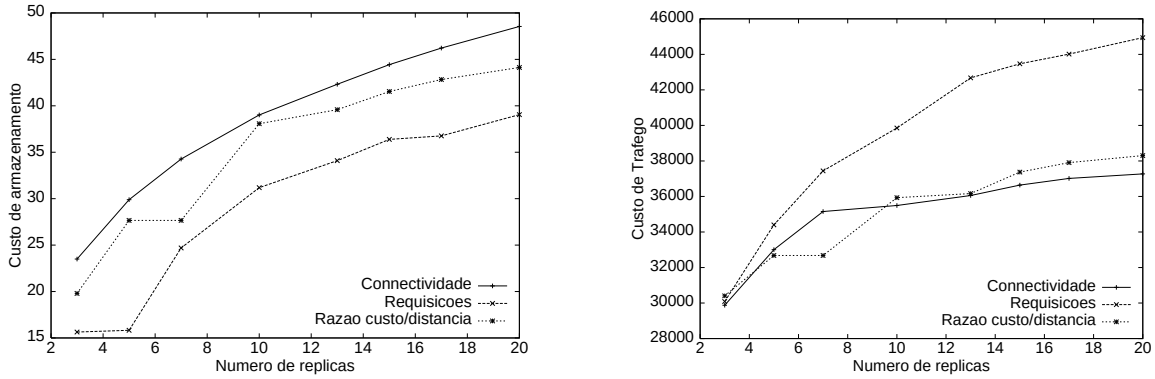

Figura 4.1: Custo de armazenamento e tráfego das heurísticas de seleção de réplicas

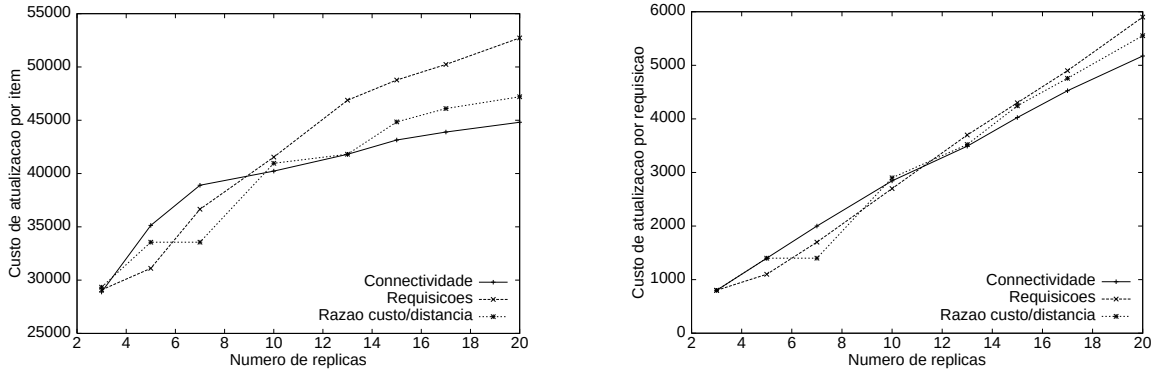

Figura 4.2: Custo de atualização por item e por recurso das heurísticas de seleção de réplicas

definido), podemos notar que algumas heurísticas priorizam determinados critérios em detrimento de outros (por exemplo, a heurística baseada no total de requisições apresenta o melhor desempenho relativo à métrica de custo de armazenamento, mas o pior desempenho relativo ao custo de tráfego), e consequentemente, as prioridades definidas pelo cliente das rede de distribuição de conteúdo (que se refletem nos fatores de proporção de custos α, β e γ previamente mencionados) terão papel fundamental na determinação da heurística mais apropriada para uma determinada aplicação.

Passando à avaliação dos parâmetros relativos às heurísticas de distribuição, nosso primeiro experimento procurou verificar como as métricas definidas eram afetadas à medida que variávamos a capacidade de armazenamento das réplicas. Neste caso, além de variarmos numericamente o parâmetro que definia a capacidade, nós variamos o critério que define a capacidade da réplica, que como mencionado na seção 4.2 pode ser associado ao poder de processamento do nodo (estimado pelo número de requisições) ou ao espaço de armazenamento (estimado pelo número distinto de itens replicados). A Figura 4.3 mostra a variação das métricas de custo de tráfego, custo de transações, custo de consistência, diferença absoluta de carga e alocações remotas para os dois casos. Para facilitar a visualização, o custo de transações e as alocações remotas foram normalizadas utilizando um fator multiplicativo arbitrário. Nossa primeira observação é a estabilidade de crescimento de todas as curvas, em proporções bastante próximas. Isso sugere que o comportamento das heurísticas em relação a este parâmetro é bastante previsível, o que facilita bastante sua calibração, especialmente se considerarmos que o mesmo está associ-

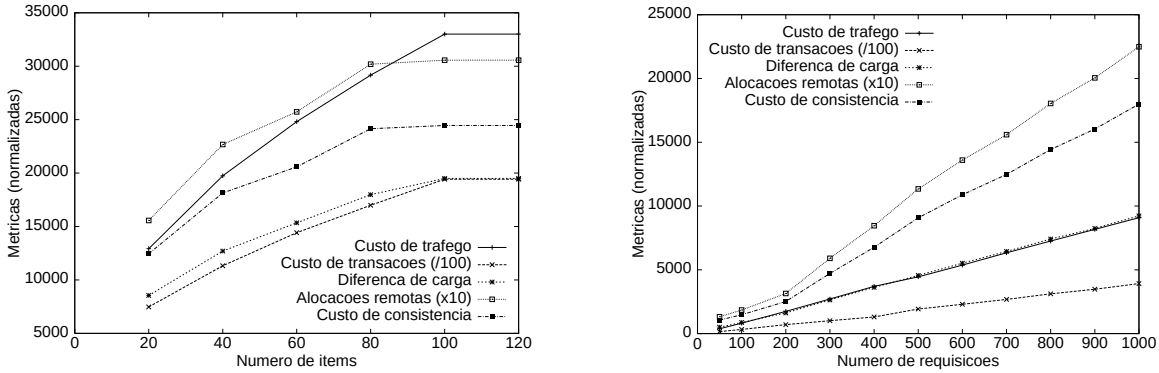

Figura 4.3: Variação da capacidade das réplicas: (a) limitada por armazenamento – número de itens; (b) limitada por processamento – número de requisições

ado a uma limitação definida, a princípio, pelos provedores da rede de distribuição. Uma outra propriedade interessante observada nos gráficos é o diferente padrão de crescimento absoluto das métricas. No gráfico de capacidade por armazenamento (4.3a), as curvas assemelham-se à exponencial negativa, enquanto que no gráfico de capacidade por processamento (4.3b) as curvas assemelham-se a uma reta. Uma possível simplificação a se considerar, seria portanto, remover os parâmetros de capacidade de réplicas e incorporá-los aos modelos heurísticos como equações, reduzindo com isso a base de dados de entrada

Por fim, nós avaliamos a variação do parâmetro γ, que é utilizado como expoente da função de ordenação das heurísticas de distribuição gulosas. Este parâmetro determina uma maior ou menor influência da distância para determinar quais os itens associados a cada nodo. Quanto menor o valor de γ, menor o peso da distância na função de ordenação, e vice-versa. A figura 4.4 mostra o efeito da variação do parâmetro sobre as métricas diretamente associadas ao total de requisições e a distância média entre as réplicas e os clientes para as duas heurísticas. Podemos observar claramente que, à medida que as requisições têm maior peso na função de ordenação (maior valor de γ), o custo de tráfego aumenta exponencialmente, e o custo de atualização cai analogamente. O primeiro fato pode ser explicado porque, à medida que o peso do fator de custo da função de ordenação diminui, a distância média entre clientes e réplicas – que é um dos fatores da função de custo – tende a aumentar, porque a prioridade de seleção passa a ser o total de requisições individual de cada item. Aumentando a distância média, aumenta-se, obviamente, os custos relativos ao tráfego. O custo de atualização, por sua vez, decai devido à capacidade finita das réplicas, que é mantida constante. Aumentando a prioridade do total de requisições individual de cada item, os algoritmos de distribuição tenderão a selecionar um conjunto de recursos menor, e que seja responsável pelo mesmo total de requisições que definem a capacidade da réplica. Diminuindo o total de itens distintos replicados, diminui também o custo de atualização.

Para outras métricas, entretanto, a variação do parâmetro γ mostrou-se muito pouco previsível. A figura 4.5 mostra alguns exemplos, especificamente, as métricas de custo de transações e custo de consistência. Nós acreditamos que este comportamento não-determinístico se deve a dois fatores. Primeiro, o fato de que algumas das métricas propostas dependam simultaneamente dos dois fatores da função de ordenação, custo e número de requisições, criando com

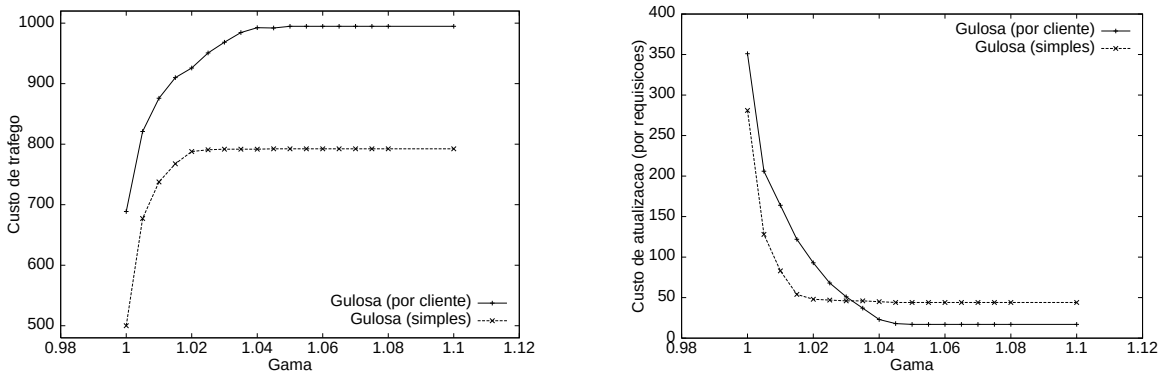

Figura 4.4: Variação do parâmetro γ: (a) Custo de Tráfego; (b) Custo de atualização por requisições

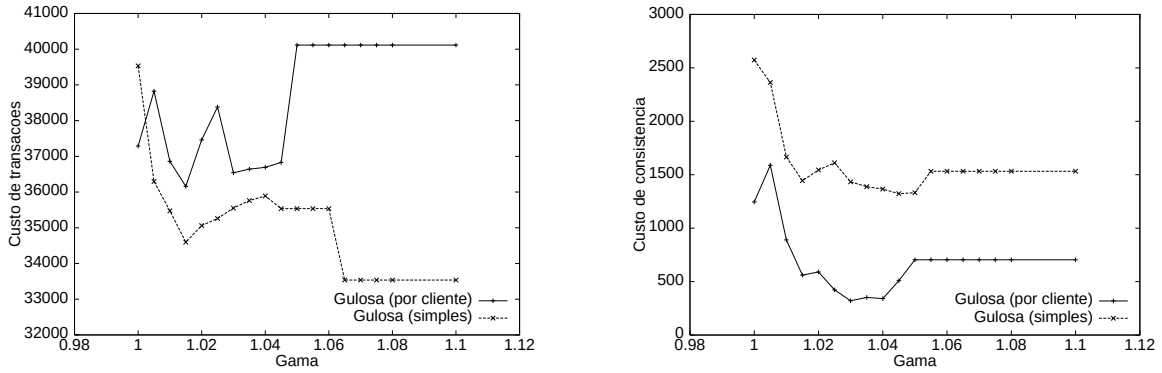

Figura 4.5: Variação do parâmetro γ: (a) Custo de transações; (b) Custo de consistência de dados

isso um compromisso entre os dois critérios bastante difícil de ser avaliado. E segundo, a dependência de algumas métricas de outros parâmetros (como por exemplo, o fator de proporção entre custos β, que é associado à função de custo das heurísticas gulosas e também na métrica de custo de transações) que pode amenizar ou acentuar as modificações decorrentes da variação do parâmetro γ.

Outro resultado interessante obtido foi a definição da janela de validade do parâmetro, que pode ser estimada empiricamente através da inspeção visual dos gráficos. Podemos notar nas figuras 4.4 e 4.5 que a partir de determinado valor de γ (em torno de 1.6 para a heurística gulosa simples e 1.8 para a heurística gulosa por cliente) não ocorrem mais alterações em nenhuma das métricas. Isso ocorre porque, acima destes limites, o aumento do parâmetro não acarreta modificações na configuração de distribuição. O que acontece neste caso, é que, a função de ordenação atinge um estado onde o peso das requisições é tão maior que os recursos são selecionados como se apenas este fator fosse considerado. Um aumento subsequente do peso das requisições não altera, portanto, a seleção dos itens, resultando em uma distribuição idêntica dos serviços.

Nosso segundo objetivo, nesta avaliação, é verificar a sensitividade de cada heurística a diferentes cargas de trabalho. Nós utilizamos, então, a mesma abordagem do modelo linear: o registro de log #1 foi utilizado para gerar as distribuições e como carga de trabalho de referência.

E o registro de log #2 foi utilizado como uma carga de trabalho de avaliação, para compararmos com a carga de trabalho de referência. Os resultados desta série de experimentos podem ser observados na Tabela 4.1. Cada coluna corresponde aos resultados de uma métrica, especificada na primeira linha. Cada linha apresenta os resultados obtidos para uma combinação de heurísticas de seleção e distribuição, especificada na primeira linha. E as células com os resultados apresentam, no canto superior, o valor absoluto da respectiva métrica nos experimentos realizados com a carga de referência e, logo abaixo, entre parênteses, o valor percentual relativo desta métrica obtida nos experimentos realizados com a carga de trabalho de avaliação.

Heurísticas	**Custo de tráfego**	**Custo de consistência**	**Diferença de carga**	**Alocações remotas**
Connectividade Popularidade	33006 -4.32%	24451 91.11%	19500 40.99%	3056 91.11%
Requisições Popularidade	34401 5.55%	4249 48.46%	20343 45.50%	3659 133.37%
Razão C/D Popularidade	32676 -4.88%	23607 97.42%	18547 45.11%	2951 97.42%
Connectividade Gulosa	32830 -4.24%	24409 95.36%	19513 41.52%	3051 95.36%
Requisições Gulosa	34344 5.68%	4232 46.93%	20317 45.63%	3649 133.66%
Razão C/D Gulosa	32430 -4.65%	24290 90.30%	18722 42.79%	3036 90.30%
Connectividade Cliente	87277 -36.81%	21184 347.77%	32958 45.65%	460 2479.50%
Requisições Cliente	62857 -1.67%	427 1130.48%	23655 95.35%	305 3815.18%
Razao C/D Cliente	88248 -36.89%	17884 435.15%	33414 44.00%	409 2823.13%

Tabela 4.1: Avaliação das heurísticas sob diferentes cargas de trabalho

Observamos pela tabela, que as métricas de distribuição de recursos são as mais afetadas pela variação da carga, confirmando as propriedades observadas na figura 3.2. O custo de tráfego, ao contrário, sofreu variações bem pequenas na maior parte dos experimentos, chegando em alguns casos apresentar um índice um pouco melhor no período de avaliação. A diferença absoluta da carga também mostrou-se bem estável, com um aumento percentual entre (40% e 46%) em quase todos os casos. Dentre as nove heurísticas consideradas, as únicas que mostraram diferenças significativas em relação às alterações na carga foram a heurística de seleção baseada no total de requisições, e a heurística de distribuição gulosa dirigida por clientes. A primeira apresenta índices muito bons relativos aos custos de consistência (aproximadamente 50%, contra 90% das outras heurísticas), porém o número de alocações remotas é maior do que as demais (aproximadamente 135% e 95%). A heurística gulosa dirida por clientes, por sua vez, apresenta em alguns casos, variações superiores a uma ordem de grandeza (particularmente, ao ser utilizada em conjunto com a heurística de seleção baseada no total de requisições). Um ponto interessante a ser ressaltado é que, para a métrica de alocações remotas onde observam-

se as maiores diferenças entre os dois períodos, o desempenho absoluto da heurística é pouco maior que o desempenho das demais heurísticas. A diferença percentual acentuada se deve ao índice muito baixo apresentado por esta heurística quando a carga de referência foi utilizada. Este padrão não se repete porém nas métricas de custo de consistência e diferença de carga, onde os índice da heurística gulosa dirigida por cliente chega a quase duas vezes o das demais.

Por fim, passamos ao terceiro objetivo de nossa avaliação que é justamente avaliar o desempenho das heurísticas entre si e em relação ao modelo de otimização linear. Nós utilizamos os mesmos registros de log e valores para os parâmetros proporção entre custos utilizados na avaliação do modelo linear, de forma a possibilitar a comparação das métricas de custo. A tabela 4.2 mostra os resultados relativos a este conjunto de experimentos. Novamente, as colunas delimitam as métricas avaliadas e as linhas delimitam as diferentes soluções de distribuições. A métrica de custo total é obtida somando-se os três custos considerados no modelo linear – tráfego, processamento e armazenamento. Em cada célula, o valor absoluto da métrica está colocado na parte superior, e o percentual em relação à configuração de distribuição do modelo linear na parte inferior.

Heurísticas	**Custo total**	**Custo de tráfego**	**Alocações Remotas**	**Diferença de carga**
Modelo Linear	18083.00	13869.66	2110.00	7480.00
Connectividade Popularidade	31503.12 74.21%	22140.37 59.63%	2704.71 28.19%	17067.76 128.18%
Requisições Popularidade	29837.42 65.00%	23993.14 72.99%	3800.77 80.13%	19091.37 155.23%
Razão C/D Popularidade	30663.36 69.57%	21838.80 57.46%	2814.73 33.40%	17292.21 131.18%
Connectividade Gulosa	23681.02 30.96%	14232.86 2.62%	8475.90 301.70%	25757.08 244.35%
Requisições Gulosa	44288.65 144.92%	31858.65 129.70%	14765.34 599.78%	49279.50 558.82%
Razão C/D Gulosa	23022.99 27.32%	14438.98 4.30%	8152.69 286.38%	24648.83 229.53%
Connectividade Cliente	22543.56 24.67%	17404.54 25.49%	2358.23 11.76%	14950.69 99.88%
Requisições Cliente	32606.17 80.31%	29602.01 113.43%	2964.90 40.52%	22274.89 197.79%
Razão C/D Cliente	32838.03 81.60%	26392.87 90.29%	2415.34 14.47%	19903.19 166.09%

Tabela 4.2: Avaliação das heurísticas em relação ao modelo linear

O principal resultado destes experimentos é o compromisso claramente evidenciado entre as métricas de custos e as métricas de desempenho de rede. As heurísticas que privilegiam o custo (como a heurística de distribuicão gulosa simples) apresentaram valores relativamente baixos nas métricas de custo total (27,32% a mais que a solução do modelo de otimização) e custo de tráfego (apenas 2,62% a mais que o modelo). Por outro lado, os índices relativos às alocações

remotas e diferença absoluta de carga foram bem altos, chegando a quase 5 vezes os valores de referência destas métricas no modelo linear.

Outro resultado interessante é a inexistência de uma solução única para qualquer das duas questões analisadas. No caso das heurísticas de seleção, a heurística com os melhores resultados varia de acordo com a heurística de distribuição a que a primeira está associada. Por exemplo, se tomarmos a métrica de custo total como referência para determinação da melhor solução de distribuição, a heurística baseada em conectividade apresenta os melhores índices quando associada às heurísticas gulosas, mas a heurística baseada na razão entre custo e número de requisições apresenta os melhores índices para a heurística baseada em popularidade. De forma similar, não podemos definir uma heurística de distribuição que tenha melhor desempenho para todos os casos – a heurística gulosa orientada por clientes apresenta os melhores resultados quando a heurística baseada em conectividade é utilizada para seleção de réplicas e, de forma análoga, a heurística orientada por popularidade tem melhores resultados quando utiliza-se a heurística baseada no total de requisições, e ainda, a heurística gulosa simples tem os melhor desempenho quando utilizamos a heurística baseada na razão entre custos e distâncias. Mais ainda, as diferentes heurísticas de distribuição de recursos impactam fortemente nas métricas de desempenho de rede – alocações remotas e diferença de carga – e, caso a escolha da solução adotada utilizar estes critérios, os compromissos entre estas métricas mencionados acima também devem ser avaliados.

Um último dado que deve ser ressaltado é a comprovação da utilidade de algoritmos gulosos e baseados em popularidade para obtenção de soluções para problemas de distribuição. No primeiro trabalho [27] que avaliou esses algoritmos já considerando a arquitetura de redes de distribuição, as estratégias avaliadas apresentaram aproveitamento relativo à solução ótima entre 110% e 150%, utilizando o custo total da solução como métrica base. Em nossa avaliação, os índices de aproveitamento relativos à solução do modelo de otimização variaram entre 120% e 180%, confirmando o bom desempenho apresentado pelos algoritmos propostos e adaptados para este trabalho.

Sumarizando, a utilização de métodos não ótimos para obtenção de soluções de distribuição pode envolver um processo trabalhoso de calibração dos modelos mas, uma vez definidos os valores de referência dos parâmetros, o bom desempenho das heurísticas em relação ao modelo de otimização e a capacidade de processamento de bases de dados muito maiores a um custo computacional muitas vezes inferior, compensam sua utilização.

Capítulo 5

Conclusões

A expansão da WWW e do Comércio Eletrônico nos últimos anos provocou um aumento significativo da demanda sobre os servidores que proveêm estes serviços. Como consequência, observa-se com frequência a diminuição da escalabilidade do sistema, e o aumento da latência percebida pelos clientes. Uma estratégia cada vez mais popular que trata estes problemas é a replicação do conteúdo oferecido em nodos mais próximos aos clientes, através de servidores Cache ou redes de distribuição de conteúdo (CDNs). Entretanto, estas soluções não oferecem mecanismos de distribuição inteligentes, que suportem serviços que requerem controle de integridade de dados e geração dinâmica de conteúdo, muito frequentemente encontrados nas aplicações de comércio eletrônico.

A elaboração de uma rede de distribuição de serviços inteligente, que contorne as deficiências dos modelos atualmente disponíveis, tem sido foco de diversos trabalhos recentes. Um dos principais problemas relacionados à implementação deste novo serviço é a definição das estratégias de alocação dos recursos replicados, de forma a otimizar o desempenho da rede, tanto em termos de custos para os clientes quanto da qualidade do serviço oferecido.

5.1 Sumário dos resultados e contribuições

Neste trabalho, nós propusemos e avaliamos algumas estratégias para o problema da alocação de recursos em uma rede de distribuição. Mais especificamente, nós dividimos o problema em duas questões menores que podem, eventualmente, ser tratadas de maneira independente: (1) quantas e quais réplicas devem ser utilizadas; e (2) como distribuir os dados entre as réplicas selecionadas.

Como primeira abordagem a estas questões, nós propusemos um modelo de otimização linear inteira derivado do problema clássico de pesquisa operacional comumente conhecido como ‘problema de transporte’. As soluções de distribuição de recursos fornecidas pelo modelo apresentaram reduções significativas de custo – em particular do custo relativo ao tráfego de requisições pela rede – o que melhora a qualidade de serviço percebida pelos clientes e diminui os custos financeiros dos provedores de conteúdo. Por outro lado, por ser baseado em métodos exaustivos, a complexidade computacional do modelo é exponencial, e consequentemente o tempo de solução do modelo linear excede os limites práticos quando a aplicação modelada

utiliza um volume de dados de entrada muito grande.

Procurando contornar estas limitações, nossa segunda proposta foi a utilização de métodos aproximados para o problema de distribuição de recursos. Utilizando como referência trabalhos anteriores no contexto dos servidores WWW tradicionais e CDNs, nós implementamos três heurísticas para cada uma das duas questões mencionadas. Para a questão de seleção de servidores, nós utilizamos heurísticas baseadas em conectividade, número de requisições e razão entre custo e requisições. Para a questão de distribuição de recursos, nós utilizamos uma heurística baseada em popularidade, uma heurística gulosa simples e uma heurística gulosa dirigida pelos nodos clientes.

Para avaliação das estratégias propostas, nós utilizamos os registros de log de uma livraria virtual. Os experimentos realizados com o modelo otimização confirmaram a correção do mesmo e mostraram sua flexibilidade em se ajustar às prioridades dos clientes relativas aos custos e à qualidade de serviço. Por outro lado, foi também evidenciado a vulnerabilidade do modelo às variações de carga entre os períodos de calibração e avaliação. Os experimentos realizados com os modelos heurísticos procuraram, primeiramente, analisar a sensitividade de cada heurística à variação dos parâmetros associados. Em seguida, verificamos os efeitos da variação de carga sobre as heurísticas combinadas como soluções de distribuição. Por último, comparamos o desempenho das heurísticas em relação à solução ótima do modelo linear, obtendo índices próximos dos evidenciados nos trabalhos que avaliaram estas mesmas heurísticas em outros contextos.

5.2 Trabalhos futuros

Como continuação ao trabalho desenvolvido para esta dissertação, nós podemos vislumbrar algumas linhas bem definidas a serem seguidas.

Em primeiro lugar, seria interessante realizar outro estudo de caso sobre os métodos avaliados, preferencialmente utilizando uma aplicação cuja carga de trabalho tenha propriedades bem distintas das apresentadas pela livraria virtual utilizada neste trabalho. Leilões eletrônicos têm se popularizado bastante nos últimos anos, e são aplicações com restrições de consistência de dados muito mais severas, pois o preço dos itens passa a depender das interações entre os usuários.

Uma outra extensão do trabalho seria a extensão do modelo de otimização original, de forma a considerar também os custos relativos à atualização dos dados por parte do *site* original. Para algumas aplicações, como por exemplo boletins de notícias *on-line*, esses custos têm um peso muito maior no desempenho da aplicação, porque a maior parte da informação é acessada apenas para leitura. Adicionalmente, seria interessante comparar o desempenho das políticas de atualizações propostas em [11] (propagação, invalidação e mista) com o modelo extendido.

Finalmente, restam os outros problemas relacionados à implementação de redes de distribuição que suportem conteúdo dinâmico. Conforme mencionado, algumas propostas relacionadas à estrutura interna destes sistemas já foram apresentadas em [29] e [5], mas até o momento não há conhecimento de nenhum protótipo que esteja operacional ou ao menos em teste. Com relação ao projeto das arquiteturas de rede, especificamente, resta ainda o problema de atualização da distribuição – associado ao balanceamento de carga da rede no longo termo –

que determina qual a frequência com que a redistribuição de recursos e o novo assinalamento dos clientes devem ser realizados.

Bibliografia

[1] Akamai: Delivering a Better Internet. Akamai technologies, inc. *http://www.akamai.com*, 1999.

[2] D. Agrawal, A. El Abbadi, and R. C. Steinke. Epidemic algorithms in replicated databases. In *In Proceedings of the sixteenth ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems*, pages 161–172, 1997.

[3] V. Almeida, D. Menascé, R. Riedi, R. Fonseca, W. Meira Jr., and F. Ribeiro. Characterizing and modeling robot workload on e-business sites. In *ACM Sigmetrics*, Boston, MA, June 2001.

[4] Paulo Henrique Araújo. Caracterização e análise de redes peer-to-peer: um estudo de caso baseado na gnutella. Master's thesis, Departamento de Ciência da Computação, Universidade Federal de Minas Gerais, 2002.

[5] André Beck and Markus Hofmann. Enabling the internet to deliver content-oriented services. In *Proceedings of the 6th Internation Workshop on Web Caching and Content Distribution*, June 2001.

[6] T. Brisco. Dns support for load balancing. RFC 1794, 1995.

[7] 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.

[8] Inktomi Corporation. The inktomi website. *http://www.inktomi.com*, 1996-2001.

[9] Microsoft Corporation. Microsoft internet security and acceleration server. *http://www.microsoft.com/isaserver/default.asp*.

[10] Christos Faloutsos, Michalis Faloutsos, and Petros Faloutsos. On power-law relationships of the internet topology. In *Proc. of ACM SIGCOMM*, August 1999.

[11] Zongming Fei. A novel approach to managing consistency in content distribution networks. In *Proceedings of the 6th Internation Workshop on Web Caching and Content Distribution*, June 2001.

[12] Cooperative Association for Internet Data Analisys. The caida web site. *http://www.caida.org*.

[13] Free Software Foundation. Gnutella home page. *http://www.gnutella.org*.

[14] Gustavo Gama, Wagner Meira Jr., Márcio Carvalho, Dorgival Guedes, and Virgílio Almeida. Resource placement in distributed e-commerce servers. In *Proceedings of the Global Internet Simposium, held in conjunction with Globecom*, San Antonio, TX, Nov 2001.

[15] Gustavo Gama, Wagner Meira Jr., Márcio Carvalho, and Dorgival Guedes Neto. Traffic-aware distribution of e-commerce services. In *Proceedings of the 17th International Teletraffic Conference*, Salvador, BA, Dez 2001.

[16] Gustavo Gama, Eduardo Kraemer, Wagner Meira Jr., and Virgílio Almeida. Representantes eletrônicos: Integrando caches www e lojas virtuais. In *Anais do XVIII Simpósio Brasileiro de Redes de Computadores*, pages 411–414, Maio 2000.

[17] M. R. Garey and D. S. Johnson. *Computers and Intractability: A Guide to the Teory of NP-Completeness*. Freeman, 1979.

[18] Marco Cesar Goldbarg and Henrique Pacca L. Luna. *Otimização Combinatória e Programação Linear - Modelos e Algoritmos*. Editora Campus, 2000.

[19] Mor Harchol-Balter, Tom Leighton, and Daniel Lewin. Resource discovery in distributed networks. In *Proceedings of ACM Principles of Distributed Computing*, 1999.

[20] Digital Island Inc. Digital island: a cable and wireless company. http://www.digisle.net, 2000.

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

[22] Oracle Corporation Inc. Oracle7 server distributed systems: Replicated data. *http://www.oracle.com/products/oracle7/server/whitepapers/replication/html/index*, 1998.

[23] Sugih Jamin, Cheng Jin, Anthony R. Kurc, Danny Raz, and Yuval Shavitt. Constrained mirror placement on the internet. In *Proceedings of Infocom 2001*, Apr 2001.

[24] S. Jin and A. Bestavros. Greedydual web caching algorithm: Exploiting the two sources of temporal locality in web request streams. In *Proceedings of 5th Web Caching and Content Distribution Workshoop*, May 2000.

[25] 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.

[26] Wagner Meira Jr., Cristina Murta, and Rodolfo Resende. *Comércio Eletrônico da WWW*. XII Escola de Computação, São Paulo, SP, Julho 2000.

[27] Jussi Kangasharju, James Roberts, and Keith W. Ross. Object replication strategies in content distribution networks. In *Proceedings of the 6th Internation Workshop on Web Caching and Content Distribution*, June 2001.

[28] P. Krishnan, Danny Raz, and Yuval Shavitt. The cache location problem. In *IEEE/ACM Transactions of Networking*, pages 8(5):568–582, October 2000.

[29] Wei-Ying Ma, Bo Shen, and Jack Brassil. Content services network: The architecture and protocols. In *Proceedings of the 6th Internation Workshop on Web Caching and Content Distribution*, June 2001.

[30] A. Medina, I. Matta, and J. Byers. On the origin of power laws in internet topologies. In *ACM Computer Communication Review, vol. 30, no. 2*, pages 18–28, April 2000.

[31] W. Meira Jr., M. Cesário, R. Fonseca, and N. Ziviani. Integrating www caches and search engines. In *Proceedings of the IEEE 1999 Global Telecommunications Internet*, Rio de Janeiro, RJ, December 1999.

[32] D. Menasce, V. Almeida, R. Fonseca, and M. Mendes. A methodology for workload characterization of e-commerce sites. In *Proceedings of ACM Conference on Electronic Commerce*, Denver, CO, November 1999.

[33] 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.

[34] Inc. Merit Networks. Merit radb services. *http://www.radb.net*.

[35] J. Nielsen. A comprehensive study on e-commerce workloads. *http://www.useit.com/alertbox/990207.html*.

[36] Vivek S. Pai, Mohit Aron, Guarav Banga, Michael Svendsen, Peter Druschel, Willy Zwaenepoel, and Erich Nahum. Locality-aware request distribution in cluster-based network servers. In *Proceedings of the ACM Eighth International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS-VIII)*, Oct 1998.

[37] Lili Qiu, Venkata N. Padmanabhan, and Geoffrey M. Voelker. On the placement of web server replicas. In *Proceedings of Infocom 2001*, Apr 2001.

[38] Pavlin Radoslavov, Ramesh Govindan, and Deborah Estrin. Topology-informed internet replica placement. In *Proceedings of the 6th Internation Workshop on Web Caching and Content Distribution*, June 2001.

[39] Ramakrishnan Rajamony and Mootaz Elnozahy. Measuring client-perceived response times on the www. In *Proceedings of the 3rd USENIX Symposium on Internet Technologies and Systems*, March 2001.

[40] Anees Shaikh, Renu Tewari, and Mukesh Agrawal. On the effectiveness of dns-based server selection. In *Proceedings of IEEE INFOCOM*, April 2001.

[41] Rita Tehan. Internet and e-commerce statistics: What they mean and where to find them on the web. Technical Report RL30435, National Library for the Environment, 2000.

[42] J. van Leeuwen. *Handbook of Theoretical Computer Science*. Elseview Science Publishers, 1990.

[43] Duane Wessels. Squid web proxy cache. *http://www.squid-cache.org*.

[44] Ellen W. Zegura, Kenneth L. Calvert, and Michael J. Donahoo. A quantitative comparison of graph-based models for internet topology. *IEEE/ACM Transactions on Networking*, 5(6):770–783, 1997.

[45] G. Zipf. *Human Behaviour and the Principle of Last Effort*. Addison-Wesley, Cambridge, MA, 1949.