Cooperação entre Caches para Web

Roberto Ferreira Brandão
Instituto de Computação
UNICAMP
brandao@dcc.unicamp.br

Ricardo de Oliveira Anido
Instituto de Computação
UNICAMP
ranido@dcc.unicamp.br

Resumo

O crescimento experimentado pela Internet nos últimos anos causa um aumento da carga sobre a rede devido ao aumento do número de requisições feitas por usuários na busca por documentos. Isto sobrecarrega os servidores de documentos e faz com que os usuários experimentem um alto tempo de espera pelos mesmos. A implantação de caches para Web é uma estratégia para minimizar os efeitos dessa sobrecarga, diminuindo assim a carga nos servidores e diminuindo também o tempo de espera dos usuários.

Sistemas de caches distribuídos são formados por malhas de servidores de cache para Web que agem cooperativamente, distribuindo a carga de requisições e aumentando a tolerância a falhas do sistema de cache, melhorando o desempenho do mesmo. Esse trabalho analisa as características relativas a um sistema de cache para Web distribuído, de forma a estabelecer quais configurações dessas características devem ser apresentadas pelo mesmo para que se obtenha do sistema de cache cooperativo o melhor desempenho possível. São apresentados um simulador de cache para Web, algumas simulações realizadas e discutidos os resultados dessas simulações.

Palavras-chave

caches para Web, cooperação, simulação de malhas de caches

1 Introdução

A WWW (World Wide Web ou simplesmente Web) é um sistema de distribuição de documentos baseada no modelo Cliente/Servidor. Atualmente, a WWW tem experimentado um crescimento exponencial e esse crescente uso dos serviços oferecidos pela Web resulta numa maior carga sobre a rede devido ao aumento do número de requisições feitas a servidores remotos. Além disso, com o crescimento da popularidade do uso da Web, os servidores considerados mais "populares" experimentam uma grande carga ocasionada pela grande quantidade de requisições. A sobrecarga na rede e nos servidores faz com que o

Trabalho parcialmente financiado pela FAPESP, projeto 96/1532-9, pelo CNPq e pelo Pronex/Finep, projeto "Sistemas Avançados de Informação", Nº 76.97.1022.00

usuário final dos serviços oferecidos pela Web, ao requisitar um documento, experimente um tempo de espera muito maior que o tempo experimentado quando a carga na rede está baixa [1].

O cache de documentos através da Web é uma forma de diminuir os problemas citados anteriormente. Devido ao crescimento exponencial do número de requisições, muitos esforços têm sido feitos no intuito de desenvolver um sistema de gerenciamento de cache que resulte numa maior economia de recursos de rede e proporcione ao usuário final o menor tempo de espera possível [2].

Serviços de cache de documentos são providos por um servidor proxy (por isso esse servidor é também conhecido como cache proxy) e tem a função de armazenar documentos mais próximo do usuário, fazendo com que as requisições feitas pelo usuário sejam respondidas mais rapidamente [3].

Caches proxy podem diminuir drasticamente a carga na rede. Por exemplo, segundo [4], em casos especiais, é possível que um cache estrategicamente colocado consiga diminuir em mais de 80% a carga na rede. Porém, caches proxy possuem limitações: em primeiro lugar, sistemas de cache trabalham apenas com documentos estáticos e, de preferência, atualizados com pouca freqüência. Documentos criados dinamicamente e que têm se tornado cada vez mais populares não podem ainda ser armazenados num cache, além de ser difícil identificar se um documento pode ou não ser armazenado num cache. Em segundo lugar, ainda não existem técnicas comprovadas que garantam a consistência entre os documentos originais e aqueles armazenados em caches.

Apesar de suas vantagens, o uso de um cache proxy introduz um novo conjunto de problemas inerentes a escalabilidade e tolerância a falhas [1]. A tolerância a falhas deve ser levada em consideração, pois um único proxy cache representa um “gargalo” de informações, além de ser um ponto em que caso aconteça uma falha, todo o sistema será paralisado. A escalabilidade também é muito importante, pois se um grande número de clientes compartilha um mesmo cache, deve ser possível fazer com que vários cache proxies cooperem, de modo a distribuir a carga de requisições.

Neste artigo são descritas a características relativas a um cache para Web, é apresentado um simulador de caches cooperativos para Web e são também descritas as primeiras simulações realizadas com o simulador, sendo apresentados e discutidos os resultados dessas simulações. Esse artigo está dividido da seguinte forma: a seção seguinte faz uma breve introdução aos sistemas de caches distribuídos cooperativos. A seção 3 descreve o funcionamento de meios de garantir a coerência entre os documentos no cache o os originais. A seção 4 apresenta algumas características relativas a performance do sistema de caches cooperativos tais como as políticas de substituição, a possibilidade do particionamento do cache, o posicionamento dos caches na malha, além de discutir alguns aspectos relativos à medida de desempenho do sistema de caches. A seção 5 descreve o funcionamento de um simulador de caches para Web cooperativos enquanto que na seção 6 são apresentadas as

primeiras simulações realizadas com o simulador desenvolvido, sendo apresentados e discutidos os resultados dessas simulações. A seção 7 apresenta conclusões relativas às experiências realizadas e apresenta algumas propostas para os trabalho futuros a serem realizados.

No contexto desse documento será utilizado o termo “acerto” referindo-se a quando o documento requisitado for encontrado no cache e será utilizado o termo “erro” em caso contrário.

2 Sistemas de caches distribuídos

O cache é uma estratégia que coloca cópias de documentos populares mais próximos dos clientes, diminuindo o tráfego na rede e a taxa de requisições recebidas pelos servidores de origem, fazendo com que os usuários da Web experimentem um menor tempo de espera ao acessar uma URL [5]. O uso de caches em Web tem se tornado uma boa opção para proporcionar um decréscimo no tráfego na rede, no número de requisições feitas a servidores remotos e, por conseqüência, no tempo de espera experimentado por um usuário na tentativa de acessar um documento remoto [6].

A utilização de vários servidores de proxy cache cooperativos torna-se cada vez mais importante devido a sua maior escalabilidade e tolerância a falhas visto que, dessa forma, não existe mais apenas um ponto de falha no sistema de cache. São estudadas então maneiras de aumentar a probabilidade de que um documento requerido já esteja armazenado em um dos caches que compõe a rede de caches cooperativos, diminuindo assim o tempo de espera experimentado pelo usuário ao requisitar um documento e, além disso, estudos são realizados de forma a determinar quais parâmetros interferem no sistema de cache distribuído.

Em um cache proxy distribuído, processadores contendo espaços para cache individuais agem cooperativamente de forma a aumentar a performance do sistema, sendo que esse conjunto de caches é visto pelos clientes como se fosse um grande cache único. Mensagens são trocadas entre os processadores que formam o cache proxy distribuído de forma a possibilitar a cooperação entre os mesmos. Em um sistema centralizado o principal gargalo está no recurso central enquanto que em um cache distribuído o problema está na comunicação, onde deve-se evitar a troca desnecessária de mensagens e manter os dados consistentes[7]. Outra grande vantagem no uso de caches distribuídos é a tolerância a falhas, visto que sistemas de cache não podem apresentar apenas um ponto de falha, pois caso ocorra uma falha nesse ponto único haverá um comprometimento do funcionamento de todo o sistema.

Em [8] é sugerido que caches proxy podem ser organizados obedecendo a uma hierarquia, onde os caches são identificados por níveis, de modo a aumentar a performance do sistema de cache. Em sistemas de cache hierárquico, os servidores resolvem erros utilizando-se de caches de nível hierárquico superior. Para distribuir a carga em um servidor cache, são utilizados servidores com o mesmo nível hierárquico.

Quando recebe uma requisição, se o servidor contiver o documento requerido em seu cache ele o envia para o cliente. Caso o documento não esteja em seu cache, o servidor questiona aos demais servidores de nível igual ao seu. Conforme discutido em [1], esse questionamento aos demais servidores não deve sobrecarregar a rede, sendo então proposto um esquema que utiliza multicasting para verificar a existência de um documento no cache de outros servidores. Se algum outro servidor contiver o documento, ele é requisitado e depois enviado ao cliente. Caso contrário, o documento é buscado no servidor de origem.

Malhas de cache devem ser utilizadas para evitar alto tráfego em links de alto custo e alto tempo de resposta. É possível definir uma hierarquia dentro da malha de servidores de cache para Web, fazendo assim que as malhas apresentem as mesmas funcionalidades observadas em servidores dispostos hierarquicamente, além da vantagem de oferecer caminhos alternativos aos dados, o que permite uma opção a um link sobrecarregado e uma maior tolerância a falhas.

Os principais sistemas que implementam cache para Web são o Netscape Proxy Server e o Squid[8]. O Netscape Proxy Server tem a vantagem de permitir a fácil implementação de um sistema hierárquico de caches, porém, não conta com bons mecanismos de tolerância a falhas. O Squid tem a vantagem de ter uma maior tolerância a falhas, além de usar o *Internet Cache Protocol (ICP)* versão 2, o que permite uma maior cooperatividade entre os servidores.

As experiências realizadas com caches para Web são, em sua maioria, feitas utilizando-se de um simulador, devido à dificuldade em ter acesso a um servidor de cache para Web para a aplicação de testes. No caso de experiências realizadas com caches cooperativos, essa dificuldade torna-se ainda maior, pois seria necessário ter acesso a uma malha inteira de servidores de cache para realizar os testes. Para que sejam realizadas as simulações de caches para Web, os simuladores recebem os *arquivos de log* ou *traces*, que são arquivos que contém o histórico das requisições feitas a um cache real. Dessa forma, situações de trabalho real do sistema de cache podem ser simuladas.

3 Coerência entre caches distribuídos

Apesar da popularização da utilização de sistemas distribuídos e hierárquicos de cache para Web, a diminuição do tempo de espera tem seu preço: todo cache algumas vezes vai enviar ao usuário um documento antigo, pois o cache, muitas vezes, não foi atualizado com a versão mais nova do documento disponível no servidor de origem [6].

Todo cache para Web precisa contar com um mecanismo que garanta o maior nível possível de coerência entre os documentos no servidor de origem e no cache. Esse mecanismo, porém, é mais simples que o utilizado em *file systems* distribuídos pois ele não possui capacidade de escritas distribuídas, visto que o cliente não escreve dados no servidor ou no cache.

Existem duas classes gerais de mecanismos de coerência de cache: a verificação de validade e a chamada de retorno *(callback)*. No mecanismo de verificação de validade,

quando um documento é colocado no cache é associado a ele um carimbo de tempo *(timestamp)*, que indica quando aquele documento foi atualizado no cache. Quando o cliente vai utilizar um documento, o cache confere se o documento ainda é válido através de mensagens trocadas com o servidor e caso o documento ainda seja válido, ele é enviado ao cliente que o requisitou. Caso contrário, o cache busca o documento atualizado no servidor de origem, armazena uma cópia e envia uma cópia ao cliente. O cache pode fazer a verificação de validade a cada intervalo fixo, de modo a aumentar a performance pois, dessa forma, não haverá uma sobrecarga na rede no momento em que o cliente fizer a requisição. No caso do mecanismo de chamada de retorno, o cliente recebe junto com o documento um certificado de chamada de retorno, que garante que o cliente será notificado caso o documento seja alterado no servidor.

A coerência dos documentos pode ser verificada a cada intervalo ou através de um mecanismo de expiração, que faz com que o documento não seja mais válido após certo tempo.

Apesar de o mecanismo de coerência por expiração ser um dos mais utilizados, ele possui as seguintes desvantagens: o usuário precisa esperar que o tempo de expiração do documento termine, não existindo a opção de se exigir coerência total através da busca do documento diretamente do servidor de origem, o mecanismo não possui uma forte garantia quanto a atualidade do documento, o usuário não consegue especificar o grau de atualidade requerido e o cache também cancela a transmissão do documento antes do término, caso o usuário venha a cancelar a transmissão antes do final da mesma.

Em [6] são propostos mecanismos de modo a resolver alguns desses problemas: para um maior controle da coerência entre os documentos, é proposto que sejam retornados também fluxos de versões dos documentos em resposta a cada requisição e que seja permitido ao usuário especificar o grau de atualidade exigido para cada requisição.

Uma técnica mais especulativa é proposta em [9]. Trata-se do pré-fetching, onde o sistema tenta “advinhar” quais documentos serão requeridos pelo cliente, de acordo com o histórico das requisições do mesmo. Desse modo, o sistema consegue, com um certo grau de erro, determinar quais documentos serão requisitados no futuro, fazendo com que o cliente tenha uma resposta muito mais rápida do sistema quando fizer uma requisição.

4 Parâmetros relacionados a caches para Web

Estudos com caches para Web concluíram que existem alguns parâmetros que interferem no desempenho de um sistema de cache. Dentre esses parâmetros, os principais são as políticas de substituição, a possibilidade de divisão do espaço destinado ao cache e a organização dos nós que formam a malha de caches.

4.1 - Políticas de substituição

Políticas de substituição são algoritmos que são aplicados quando o tamanho do documento que deverá ser gravado no cache exceder o espaço livre do cache, ou a cada período fixo de tempo, ou ambos. No primeiro caso, são removidos os documentos considerados mais dispensáveis até que o espaço seja suficiente para armazenar o novo documento.

A eficiência de um sistema de cache depende da política de substituição do cache. Existem três grandes questões quanto à remoção de documentos de um cache: o que remover, quando remover e quantos documentos devem ser removidos [2].

As políticas de substituição mais comuns são descritas resumidamente a seguir [4]:

- FIFO (First In First Out) - Remove-se o documento mais antigo.
- LRU (Least Recently Used) - Remove-se o menos recentemente acessado.
- LRU-MIN - Similar ao LRU mas com considerações sobre o tamanho do documento, onde são removidos preferencialmente os documentos maiores.
- LFU - (Least Frequently Used) - Remove-se o documento usado com menor freqüência.
- SIZE - Remove-se o maior documento.
- Log_2SIZE - Remove-se um dos maiores documentos.

A política FIFO também é conhecida como ETIME; a política LRU também é conhecida como ATIME e a política LFU também é conhecida como NREF.

Estudos sobre quais documentos devem ser removidos do cache indicam que é preferível manter no cache um arquivo recentemente acessado ao invés de um arquivo que foi acessado com freqüência no passado. Em [3], na busca de uma melhor performance em um proxy cache foi desenvolvida uma nova técnica, a LRV (Lowest Relative Value), que descarta o documento em função de um valor associado a ele. Esse valor é função da repetição de acessos ao documento, do tempo desde o último acesso e do tamanho do documento. Considera-se que o tipo e a fonte do documento não influem na sua probabilidade de ser reacessado.

Além da LRU, experimentos mostraram que a política SIZE (e suas variantes) também proporcionam uma alta taxa de acerto devido à característica apresentada pela maioria dos documentos transmitidos pela Web de ter um tamanho reduzido, em razão do menor tempo necessário para o envio dos mesmos. Porém, o fato de serem escolhidos como mais favoráveis para descarte os documentos maiores faz com que a política SIZE tenha um desempenho menor que os demais para caches pequenos, o que se explica pela poluição do cache por arquivos pequenos que nunca são reacessados e dificilmente são descartados.

Em [5] são propostas variantes do algoritmo LRU, as políticas LRU-MIN e LRU-THOLD. No algoritmo LRU-MIN são descartados os maiores documentos daqueles menos

recentemente acessados no cache. O uso do algoritmo LRU-THOLD faz com que seja estabelecido um tamanho máximo, sendo que são armazenados no cache apenas documentos com tamanhos menores que o máximo. Verificou-se que esse tamanho máximo de documentos é função da quantidade de espaço disponível no cache: quanto mais espaço disponível houver no cache, maior pode ser o tamanho máximo dos documentos.

Os experimentos realizados em simuladores em diversos trabalhos mostraram que a taxa de acerto tende a diminuir com o tempo, independente da política utilizada. Estima-se que a razão disso seja o fato de que quando os browsers usam seus próprios caches, o cache proxy funcione como um cache de segundo nível, que apresenta uma menor taxa de acerto que os de primeiro nível.

4.2 - Divisão do cache

A divisão do espaço de cache em partições destinadas a tamanhos distintos de documentos proporciona uma melhor performance do sistema de cache [10]. Essa melhoria na performance do sistema de cache com o particionamento por tamanho se deve à característica de que os documentos transmitidos via Web apresentam uma "distribuição de cauda longa" e, sendo assim, em um cache particionado, evita-se que vários documentos pequenos e populares sejam removidos do cache para que seja armazenado um documento muito grande e acessado poucas vezes. A divisão do cache não precisa ser necessariamente pelo tamanho, podendo-se dividir o espaço destinado ao cache, por exemplo, de forma que cada partição atenda a um domínio específico.

4.3 - Localização dos nós em caches cooperativos

Segundo [8], os servidores de cache para Web devem ser colocados no fluxo de dados, no lado mais próximo dos clientes (tomando como referência um gargalo, que pode ser representado, por exemplo, por um link de baixa capacidade de transferência de dados). O cache de primeiro nível deve ser colocado o mais próximo possível da conexão externa à Internet, próximo aos usuários e, no caso de existirem vários servidores, eles devem ser colocados nas junções das LANs. Para caches de nível mais alto, esses devem ser colocados em junções de redes e em pontos de conexão nacional e internacional da Internet.

As taxas de acerto diminuirão em servidores de nível mais alto, pois eles receberão uma variedade maior de requisições. O alto tráfego no nível mais alto da hierarquia exige que o equipamento seja dimensionado para suportar a alta carga que irá receber.

4.4 – Métricas do desempenho de um cache distribuído

Para medir a eficiência de um sistema de cache para Web são utilizados os métodos de taxa de acerto e taxa de acerto por byte. No método de taxa de acerto é medido o percentual de requisições que são satisfeitas por documentos do cache enquanto que no método do taxa de acerto por byte é medido o percentual de bytes que são enviados ao cliente diretamente do

cache, sem ser necessário fazer uma requisição ao servidor de origem. Dessa forma, é possível calcular o tempo de espera pelo documento economizado pelo usuário.

O simulador descrito nesse artigo utiliza as duas métricas para calcular o desempenho das malhas de caches simuladas.

5 Um simulador de cache cooperativo para Web

Para que pudessem ser feitas as experiências com caches para Web foi implementado um simulador capaz de simular situações reais de operação de um sistema cooperativo e hierárquico de caches para Web. O funcionamento do simulador está representado esquematicamente na figura 5.1.

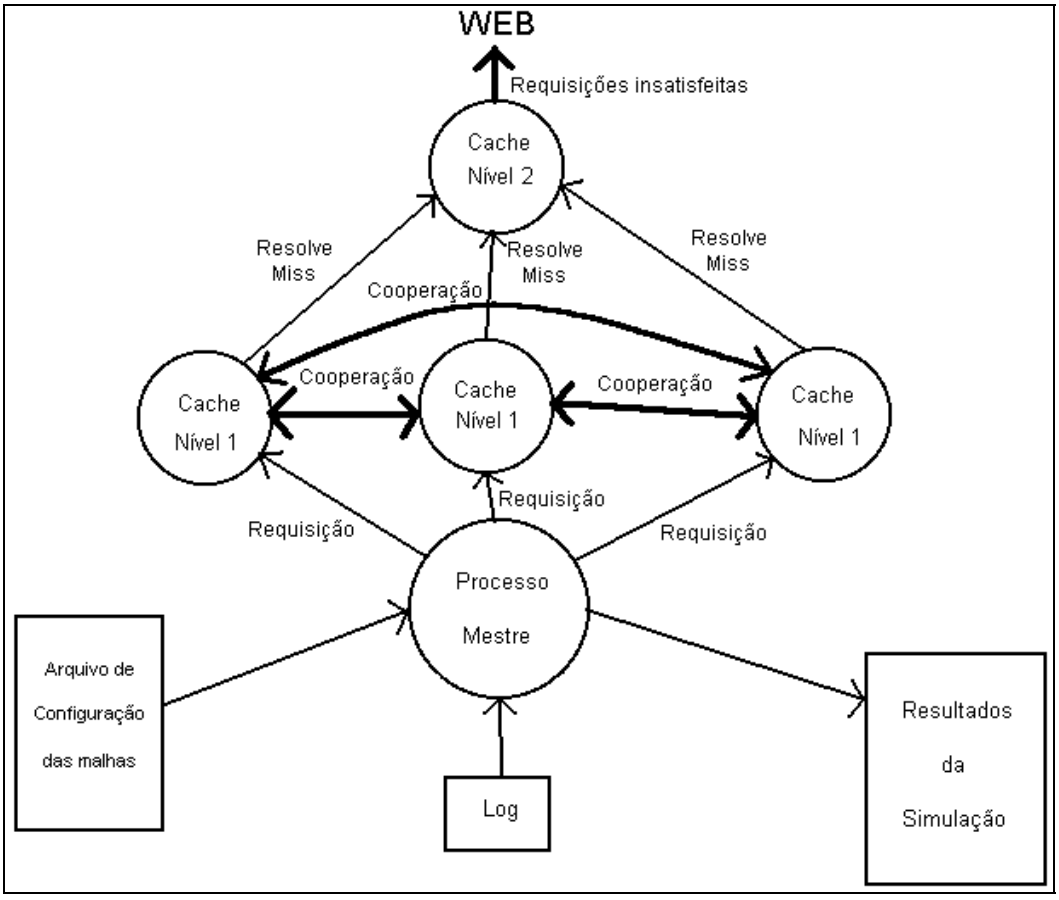

Fig. 5.1 - Funcionamento do simulador de cache para Web

A figura 5.1 mostra um exemplo de simulação de uma malha formada por quatro caches cooperativos. Nesse exemplo, a malha é formada por três caches de nível 1 e um cache de nível 2. Um processo mestre cria e configura os caches a serem simulados de acordo com um arquivo que contem os parâmetros de configurações das malhas simuladas. Na simulação, o processo mestre atribui um arquivo de *log*, com exemplos de requisições reais feitas a um

cache real e conseguidas em [11] a cada cache de nível 1, sendo que as requisições são feitas seguindo a ordem cronológica das requisições existentes no arquivo de *log*.

A seguir, a figura 5.2 mostra a listagem do arquivo de configuração de uma malha formada por quatro caches, semelhante à malha mostrada na figura 5.1.

```
structdef
        cachedef
          cacheId 11
          resolveMiss 21 2
          sub 100 0 0 lru
          cacheCoop 12 1
          cacheCoop 13 1

        cachedef
          cacheId 12
          resolveMiss 21 2
          sub 100 0 0 lru
          cacheCoop 11 1
          cacheCoop 13 1

        cachedef
          cacheId 13
          resolveMiss 21 2
          sub 100 0 0 lru
          cacheCoop 11 1
          cacheCoop 12 1

        cachedef
          cacheId 21
          resolveMiss 0 2
          sub 100 0 0 lru
```

Fig. 5.2 - Exemplo de arquivo de configuração

O arquivo de configuração contém informações de configuração de uma ou mais malhas. Cada malha deve ser definida a partir da palavra-chave **structdef**. A listagem apresentada na figura 5.2 mostra um exemplo de arquivo de configuração com parâmetros para a criação de apenas uma malha. Cada cache é definido a partir da palavra-chave **cachedef**, sendo definidos quatro caches na listagem apresentada na figura 5.2.

Os parâmetros de configuração de cada cache são apresentados a seguir:

cacheId *id* - atribui um identificador para o cache.

resolveMiss *id latência* - indica qual é o identificador do cache pai, utilizado para resolver erros. No caso do identificador de cache pai ser identificado por 0, significa que não possui cache pai, sendo que suas requisições não satisfeitas serão repassadas para os

servidores de origem, através da Web. O parâmetro latência representa uma constante relativa ao tempo de transmissão do meio de transmissão que liga o cache ao seu cache pai. Foi assumido nas simulações que esse valor representa o tempo em milisegundos necessário para a transmissão de 1 Kbyte através do meio de transmissão.

sub ***tamanho mínimo máximo política*** - atribui ao cache uma partição de tamanho em Mbytes definido por ***tamanho***. O parâmetro ***mínimo*** indica o tamanho do menor arquivo que pode ser armazenado nessa partição enquanto que o parâmetro ***máximo*** indica o tamanho do maior arquivo. Caso o parâmetro ***máximo*** tenha o valor 0, significa que não existe teto máximo para o tamanho do arquivo. O parâmetro ***política*** indica qual política de substituição será utilizada na partição. Os valores possíveis são LRU, SIZE e LFU.

cacheCoop ***id latência*** - indica que o cache cujo identificador é ***id*** vai agir cooperativamente com o cache. O parâmetro latência representa uma constante relativa ao tempo de transmissão do meio de transmissão que liga o cache ao seu cache pai. Mais uma vez, foi assumido nas simulações que esse valor representa o tempo em milisegundos necessário para a transmissão de 1 Kbyte através do meio de transmissão.

O simulador funciona da maneira descrita a seguir: ao receber uma requisição, um cache procura nas listas de arquivos armazenados se o arquivo requisitado está armazenado. Em caso afirmativo, um acerto é anunciado. Caso contrário, cada vizinho (cache de mesmo nível), com os quais existe uma cooperação com o primeiro cache recebe a requisição do arquivo. Cada vizinho faz uma busca em suas listas de arquivos armazenados e caso um ou mais encontre o arquivo, o primeiro cache é notificado e um acerto é anunciado. No caso de nenhum vizinho encontrar o arquivo, o cache inicial envia então a requisição para o cache de nível acima que vai proceder da similarmente até que seja verificado um acerto ou até que não existam mais caches de nível superior, significando que o arquivo teria de ser buscado na Web, quando então é anunciada a ocorrência de um erro, pois o arquivo não se encontra em nenhum cache da malha.

É interessante mencionar que saber em qual cache aconteceu o acerto é importante, sendo que cada acerto ocorrido é associado ao número de *hops* necessários para sua ocorrência. Além disso, é medida a quantidade de bytes encontrada no cache; calculando o percentual dos bytes encontrados no cache em relação ao total de bytes de todas as requisições é possível determinar o tempo médio economizado ao usuário com a implantação de cada sistema de caches cooperativos. A cada comunicação entre caches é atribuído um tempo de envio, proporcional à latência do meio de transmissão que liga os dois caches em comunicação. São simuladas também as mensagens trocadas entre os caches cooperativos de forma que obedeçam ao padrão do ICP *(Internet Cache Protocol)*, descrito em [12].

Cada cache simulado possui as capacidades representadas na figura 5.3, que ilustra também os procedimentos realizados em cada cache simulado.

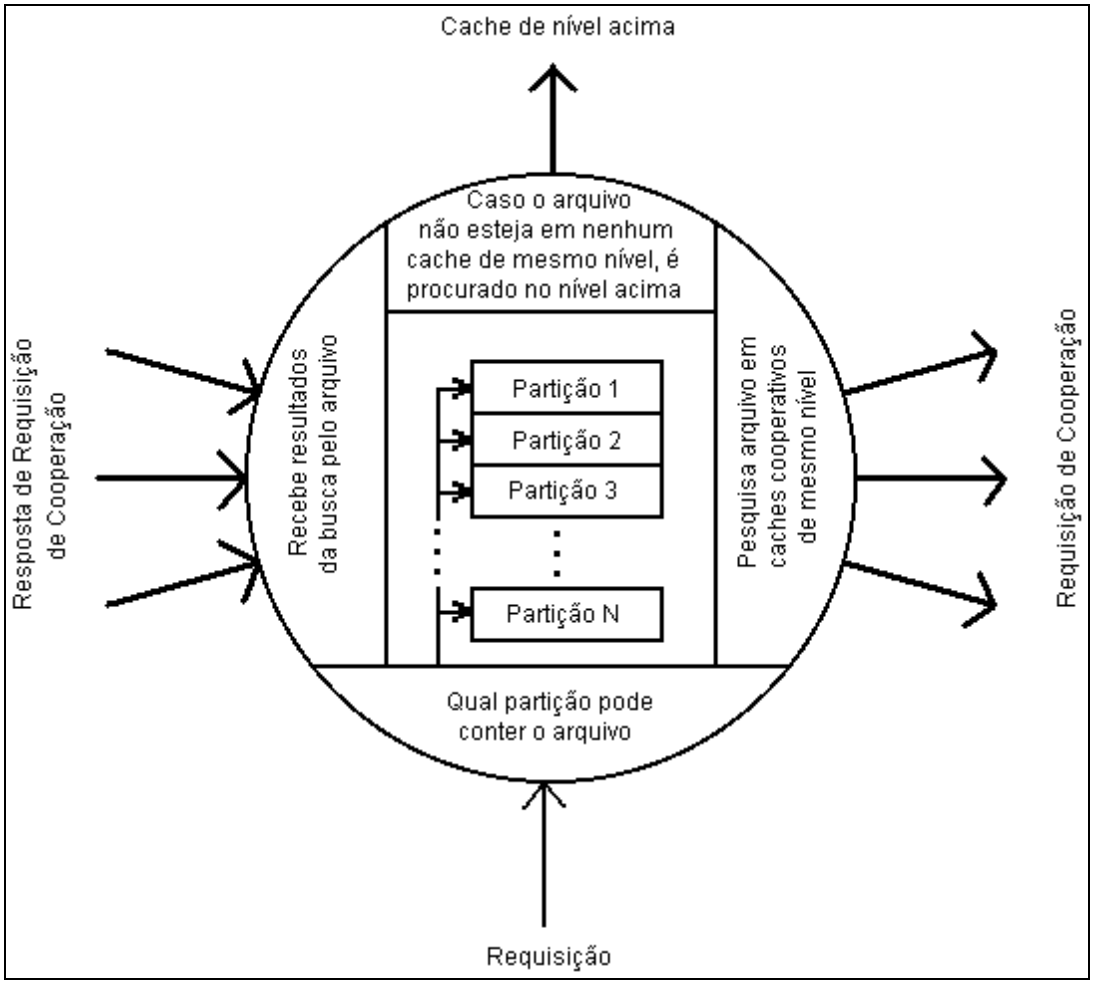

Fig. 5.3 - Funcionamento de um cache simulado

Conforme pode ser observado na figura 5.3, cada cache simulado é formado de uma ou mais partições. A cada partição está associado um intervalo de tamanhos e uma política de substituição que será aplicada na remoção de arquivos quando o espaço disponível no cache não for suficiente para o armazenamento de um novo arquivo. O intervalo de tamanhos contém os possíveis tamanhos de arquivos que podem ser armazenados na partição. Foram implementadas as três principais políticas de substituição existentes: LRU, SIZE e LFU.

Cada cache simulado é capaz de interagir tanto com seus vizinhos quanto com o cache de nível acima ao qual são repassados as requisições não satisfeitas tanto localmente quanto nos vizinhos.

A possibilidade de uma maior coerência entre os documentos dos caches e os documentos originais foi aumentada utilizando-se um mecanismo de expiração que remove a cada intervalo fixo os arquivos armazenados anteriormente a um determinado momento. Esse mecanismo também é importante para evitar a poluição dos caches por arquivos antigos e que, por características particulares das políticas de substituição dificilmente serão removidos, ocupando espaço dos caches. Como exemplo, considere o caso de um cache que utilize a política SIZE, onde pode acontecer do armazenamento de um arquivo muito pequeno e

acessado apenas uma vez e que dificilmente será removido, pois a política SIZE remove em primeiro lugar os arquivos maiores.

6 Simulações realizadas

As primeiras simulações realizadas utilizando o simulador de caches cooperativos foram feitas de forma a comparar o desempenho de uma malha de caches hierárquicos em relação a uma malha sem hierarquia, onde os caches são colocados em linha. Foram utilizados caches idênticos de 100 Mbytes, não particionados, para compor todas as malhas simuladas. Foi utilizada a política LRU em todos os caches, sendo processadas cerca de 10 milhões de requisições, conseguidas em [11]. Apesar de 100 Mbytes ser um tamanho pequeno em relação à maioria dos caches reais, esse tamanho foi escolhido devido ao número reduzido de requisições utilizadas, sendo que esse número de requisições foi escolhido de forma a diminuir o tempo total da simulação. Como essas foram as primeiras simulações realizadas, espera-se como trabalhos futuros a simulação de um número bem maior de requisições, utilizando-se modelagens de malhas reais de caches cooperativos.

Para a simulação dos caches em linha, foram feitas as simulações de um a oito caches cooperativos colocados no nível 1. A estrutura genérica da malha simulada pode ser vista na figura 6.1.

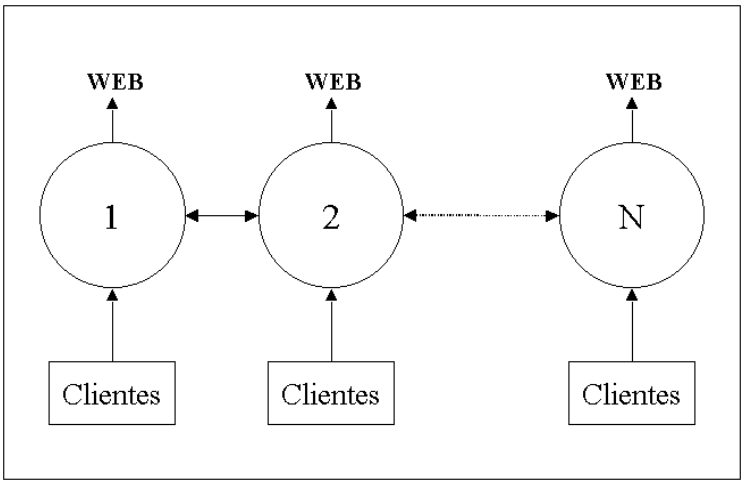

Fig. 6.1 - Estrutura genérica das malhas não-hierárquicas simuladas

Os resultados da simulação das malhas não-hierárquicas são apresentados na tabela 6.2 a seguir. O número de *hops* indica a quantidade de transmissões da requisição, contadas a partir do cliente. Um acerto com 1 *hop* indica que o cache que recebeu a requisição continha em seu cache o arquivo enquanto que um acerto com 2 *hops* indica que a requisição foi satisfeita por vizinhos do primeiro cache a receber a requisição. O percentual de acertos foi dividido entre os números de *hops* relativos aos acertos pois um acerto ocorrido com menor número de *hops* indica uma resposta mais rápida ao usuário.

	Percentual de acertos por *hop*	
Número de caches	1 *hop*	2 *hops*
1	22.5981	----
2	22.3635	2.2542
3	20.3704	7.3659
4	20.1455	9.7225
5	20.3391	13.4722
6	20.4973	17.6039
7	20.7718	22.3856
8	20.4310	24.4784

Tabela 6.2 - Resultados da simulação de malhas não-hierárquicas

Para a simulação dos caches hierárquicos, foram feitas simulações de malhas formadas por um a oito caches, sendo que um dos caches foi colocado no nível 2 e o restante foi colocado no nível 1. Os caches de nível 1 foram configurados para agir cooperativamente e enviar requisições não satisfeitas ao cache de nível 2. A estrutura genérica da malha hierárquica simulada pode ser vista na figura 6.3.

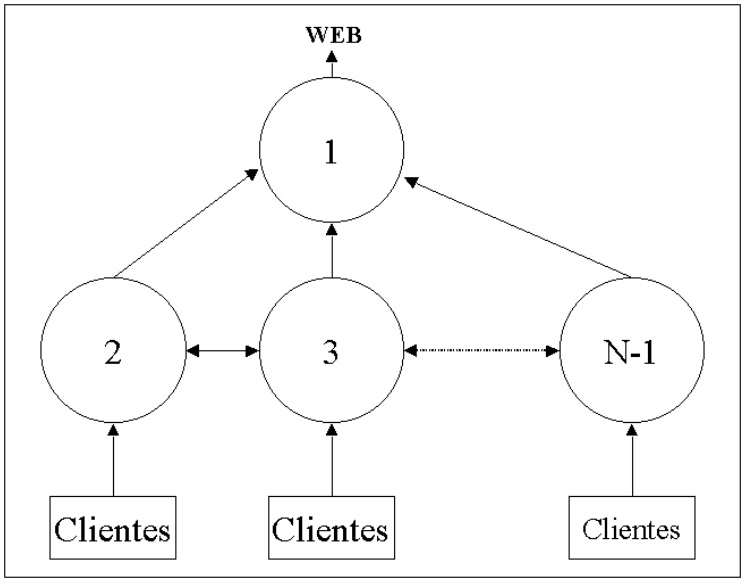

Fig. 6.3 - Estrutura genérica das malhas hierárquicas simuladas

Os resultados da simulação das malhas hierárquicas são apresentados na tabela 6.4 a seguir. Assim como na simulação anterior, o número de *hops* indica a quantidade de transmissões da requisição, contadas a partir do cliente. Um acerto com 3 *hops* indica que apenas o cache pai do cache que recebeu a requisição continha o arquivo.

	Percentual de acertos por *hop*		
Número de caches	1 *hop*	2 *hops*	3 *hops*
1	22.5981	----	----
2	22.5969	0.3400	----
3	22.3628	2.2537	0.2483
4	20.3707	7.3674	0.1725
5	20.1442	9.7223	0.1686
6	20.3379	12.4720	0.1372
7	20.4966	17.6026	0.1171
8	20.7711	22.3841	0.1121

Tabela 6.4 - Resultados da simulação das malhas hierárquicas

A figura 6.5 mostra o gráfico dos percentuais de requisições satisfeitas com 2 *hops*, apresentados pelas malhas hierárquicas e não-hierárquicas de caches cooperativos em função do número de caches na malha. É mostrado também, o gráfico dos resultados da simulação de um único cache com o tamanho igual à soma dos tamanhos de todos os caches de uma das malhas simuladas. Os resultados apresentados na simulação com um único cache representam, nas condições da simulação, o melhor desempenho possível para a capacidade de armazenamento disponível.

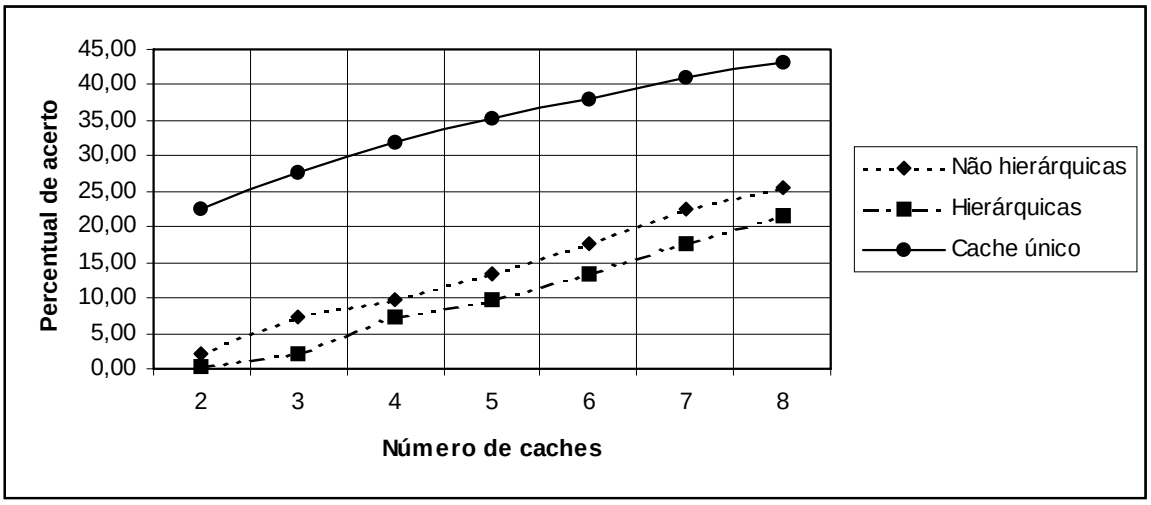

Fig. 6.5 - Gráfico comparativo da taxa de acerto

Através do gráfico da figura 6.5, podemos observar que as malhas não-hierárquicas apresentaram um melhor desempenho que as malhas hierárquicas. Porém, nessa simulação foi considerado um tempo de transmissão igual para todos os meios de transmissão que ligam caches vizinhos. Esse fato não acontece na prática em redes de caches cooperativos geograficamente dispersos, onde os tempos de transmissão são diferentes e altos, em comparação com redes locais. Isso força à configuração das malhas de uma maneira

hierárquica como forma de diminuir tanto o número de caches consultados em uma única requisição quanto a carga nos meios de transmissão. Porém, apesar da estruturação hierárquica apresentar piores resultados, a semelhança no comportamento das duas curvas no gráfico da figura 6.5 mostra que é possível utilizar malhas hierárquicas com resultados compatíveis aos resultados apresentados pela utilização de malhas não-hierárquicas. Com relação ao tamanho dos caches na simulação, novas simulações serão realizadas de forma a verificar se as curvas de taxa de acerto mantém seu comportamento, apesar do aumento da taxa de acerto causado pelo aumento da capacidade de armazenamento dos caches simulados.

7 Conclusões e trabalhos futuros

A utilização de caches cooperativos possibilita um melhor aproveitamento dos recursos dos meios de transmissão, possibilitando um menor tempo de espera aos usuários dos serviços da Web. Devido às baixas taxas de transmissão apresentadas pelos meios de transmissão que interligam os caches geograficamente dispersos, a configuração de hierarquias dentro das malhas de caches cooperativos permite uma melhor utilização dos meios de transmissão entre caches geograficamente dispersos, apresentando desempenhos compatíveis aos de malhas não hierárquicas.

Como trabalhos futuros, pretende-se implementar uma versão paralela do simulador, usando plataformas tais como o PVM de forma a realizar um número maior de simulações, utilizando um número maior de requisições, possibilitando assim a análise das muitas configurações possíveis para malhas de caches hierárquicos cooperativos.

8 Referências bibliográficas

[1] Malpani, R.; Lorch, J.; Berger, D. Making World Wide Web caching servers cooperate. 4th International World-wide Web Conference, pp 107-117, Dezembro 1995.

URL: http://www.w3j.com/1/Iorch.059/paper/059.html, junho 1998.

[2] Kim, H.; Chon, K.; Update policies for network caches.

URL: http://cosmos.kaist.ac.kr/salab/publication/technical-memo/SAL-TM-75/index.html, julho 1998.

[3] Lorenzetti, P.; Rizzo, L. Replacement policies for a proxy cache.

URL: http://www.iet.unipi.it/~luigi/caching.ps.gz, julho 1998.

[4] Willians, S.; Abrams, M.; Standridge, C. R.; Abdulla G.; Fox E. A.; 1996. Removal policies in network caches for World Wide Web Documents. *ACM SIGCOMM 96*, pages 293-305, agosto 1996.

[5] Abrams, M.; Standridge, C. R.; Abdulla, G.; Willians, S.; Fox, E. A. Caching proxies: Limitations and potentials. 4th International World-wide Web Conference, pp 119-133, Dezembro 1995. URL: http://ei.cs.vt.edu/~succeed/WWW4/WWW4.html.

[6] Dingle, A. Web cache coherence. *Computer Networks and ISDN systems,* vol. 28, Nº 7-11, pp 907-920, 1996.

[7] Kurcewicz, M.; Sylwestrzak, W.; Wierzbicki, A. A distributed WWW cache. URL: http://wwwcache.ja.net/events/workshop/09/paper.html, junho 1998.

[8] Melve, I. 1997. Web caching architecture. URL: http://ircache.nlanr.net/Cache/www.uninett.no/prosjekt/desire/arneberg/altsamme n.html, julho 1998.

[9] Chinen, K.; Yamaguchi, S. 1997. An Interactive Prefetching Proxy Server for Improvement of WWW Latency.
URL: http://fc.vdu.lt/Conferences/F%3A%20[INET97]/A1/A1_3.HTM, junho 1998.

[10] Murta, C. D. 1998. Cache Particionado - Uma nova abordagem para cache na WWW. URL: http://wwwcache.ja.net/events/workshop/24/, agosto 1998.

[11] Examples of Squid Log Files.
URL: ftp:// ftp.ircache.net/Traces/, janeiro 1999.

[12] Wessels, D.; Claffy, K.; Internet Cache Protocol (ICP), version 2. Networking Working Group, RFC 2186, set. 1997