Um simulador paralelo para sistemas de caches para Web

Roberto Ferreira Brandão, Ricardo de Oliveira Anido

Instituto de Computação, UNICAMP
Rua Albert Einstein, 1251, Campinas, São Paulo, Brasil
{brandao, ranido@dcc.unicamp.br}

Resumo

O uso de caches para Web permite oferecer uma maior qualidade de serviço aos usuários da Internet. Devido à dificuldade em se obter uma malha real de caches para fazer experiências, pesquisas relacionadas com caches para Web são feitas utilizando simuladores.

Esse artigo apresenta um simulador paralelo de malhas caches para Web que executa simulações do uso de caches distribuídos e cooperativos. Esse simulador permite a verificação do impacto no desempenho da variação de parâmetros relativos a caches para web. Dessa forma, é possível projetar de forma mais consciente, malhas de caches para Web cooperativos.

Para mostrar um exemplo do uso do simulador, foram realizadas simulações baseadas na malha da Rede Nacional de Pesquisa - RNP brasileira.

Palavras chave - caches para web, simulação, modelagem da web, avaliação de desempenho.

Abstract

The use of web caches makes it possible to offer a service of better quality to Internet users. Due to the difficulty in obtaining a real mesh of caches to experience with, most research projects use web caches simulators.

This paper presents a web cache simulator capable of executing simulations of distributed and cooperative web caches servers. The simulator allows to measure the impact of configuration parameters in cache´s performance. In this way, it is possible to make better projects of cooperative caches meshes.

To show an example of simulations usage, it is presented simulations based in the Brazilian National Research Network - RNP.

Key-words - web caches, simulator, web modeling, performance evaluation.

1. Introdução

Utilizar caches para Web significa colocar cópias de documentos muito acessados em servidores mais próximos aos clientes, diminuindo assim a carga de requisições feitas à rede e a servidores de documentos muito populares. Dessa forma, como resultado final, o usuário dos serviços prestados pela Web experimenta uma diminuição do tempo de resposta a requisições [7], [5].

Vários parâmetros estão relacionados ao desempenho de caches para Web [8]. Dentre os principais, pode-se citar o tamanho do espaço de armazenamento, a capacidade de processamento do processador no servidor de cache e a política de substituição utilizada. Políticas de substituição são algoritmos executados para escolher um documento a ser removido do cache

para armazenar um novo documento, quando o espaço livre é insuficiente para armazenar o novo documento [3], [1].

Para descobrir qual a melhor configuração de parâmetros relativos a um determinado sistema de caches para Web, deve-se testar várias combinações de valores para os parâmetros de forma a determinar qual dessas combinações possibilita o melhor desempenho. No caso de malhas de caches cooperativos, deve-se considerar também a distribuição hierárquica dos servidores e a capacidade de transmissão das redes que os interligam. Devido à quantidade de recursos envolvidos nesse tipo de pesquisa, fica difícil conseguir uma malha real de caches para fazer experiências. A simulação se torna então a opção mais viável para pesquisas.

Um dos principais problemas encontrados na simulação de malhas de caches para Web é a manutenção de coerência entre as operações do simulador e as operações de uma malha real de caches. Assim, torna-se necessário utilizar históricos de requisições reais, feitas a caches reais, no simulador. Esses históricos de requisições são chamados *arquivos de log* ou *traces* e podem ser conseguidos em vários *sites* na Internet.

Nesse artigo é apresentado um simulador totalmente configurável de malhas hierárquicas de caches para Web. Esse simulador utiliza *traces* de caches reais, permitindo verificar o impacto da variação de parâmetros no desempenho de toda a malha de caches.

2. Simulação de malhas de caches cooperativos

A cooperação entre servidores de caches para Web permite um melhor desempenho de todo o sistema de caches. Quando um servidor não possui uma cópia de um documento requisitado, ele pode enviar uma mensagem a um outro servidor de cache próximo perguntando se esse outro servidor tem possui uma cópia do documento. Em caso afirmativo, pode-se satisfazer a requisição do cliente com essa cópia. Caso o outro cache também não possua uma cópia do documento, pode-se também optar por requisitar a um servidor de cache de hierarquia superior. Esse outro servidor de cache, geralmente de maior capacidade, recebe requisições não satisfeitas de servidores menores.

Esse procedimento pode prosseguir até o cache de maior nível hierárquico que, se também não encontrar o documento, irá requisitá-lo ao servidor real do documento, guardando uma cópia para si.

Além dos parâmetros que interferem no desempenho de caches não cooperativos, quando é configurada uma malha de caches cooperativos, novos parâmetros interferem no desempenho de toda a malha. Dentre esses parâmetros, pode-se citar a configuração de uma estrutura hierárquica entre os caches e as capacidades da rede de intercomunicação entre os caches. Além disso, a demanda por memória e capacidade de processamento em servidores de uma malha é maior pois eles devem executar também os protocolos de troca de informações entre caches.

A figura 1 apresenta um exemplo de malha de servidores de caches distribuídos e cooperativos. Durante a operação dos servidores representados nesse exemplo, usuários fazem requisições aos caches 1, 2 e 3. Caso o cache que recebeu a requisição (1, 2 ou 3) não encontre o documento requisitado, ele envia uma mensagem aos dois caches vizinhos (no caso do cache 1 os vizinhos são os caches 2 e 3), perguntando sobre o documento. Caso nenhum possua o documento, é feita uma requisição ao cache 4. Se o cache 4 possuir o documento requisitado, ele envia uma cópia ao cache de nível mais baixo que requisitou. Esse cache armazena então uma cópia e envia o documento requisitado para o cliente. Caso o cache 4 não possua o documento,

ele fará uma requisição ao servidor real do documento, guardando uma cópia para si e enviando outra para o cache que requisitou.

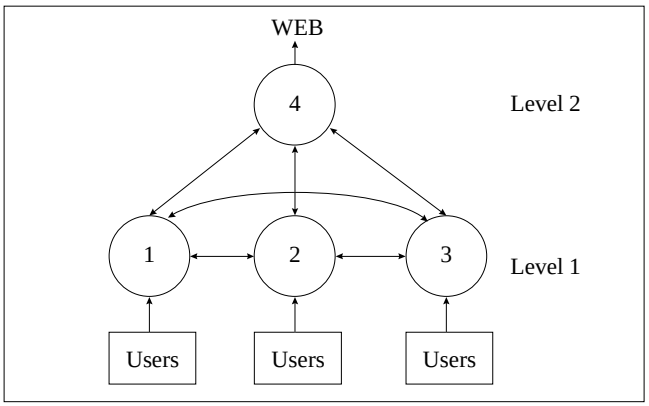

Fig.1 Exemplo de malha de caches cooperativos

É interessante ressaltar que a capacidade de comunicação entre os caches 1, 2 e 3 geralmente é maior que aquele entre esses caches e o cache 4. Isso ocorre porque geralmente o cache 4 está geograficamente distante dos caches 1, 2 e 3. Como o cache 4 recebe um número maior de requisições que os caches 1, 2 e 3, ele deve possuir uma capacidade de armazenamento e processamento maior que os outros caches.

Para quantificar o desempenho de caches para Web, normalmente usa-se o *Hit Ratio* e o *Byte Hit Ratio*. *Hit Ratio* é o percentual de requisições que foram satisfeitas pelo cache enquanto que o *Byte Hit Ratio* é o percentual de bytes requisitados que foram enviados pelo cache. Assim, quando um documento requisitado foi encontrado em um cache, diz-se que houve um *Hit*. Ao contrário, quando o documento não está no cache diz-se que ocorreu um *Miss.* Além do H*it Ratio* e do B*yte Hit Ratio*, esse trabalho usa uma outra métrica muito importante: o tempo de resposta ao usuário, que indica qual o impacto no tempo de espera do usuário das variações de parâmetros de configuração.

3. Trabalhos anteriores relacionados a caches para Web

Muitos trabalhos de pesquisa relacionados a caches para Web têm sido realizados nos últimos anos. A maioria desses trabalhos utiliza-se de simuladores, pelos motivos já citados. Esses simuladores, na grande maioria das vezes, trabalham com apenas um cache, permitindo apenas a comparação entre diferentes políticas.

Os principais artigos sobre caches para web são citados a seguir:

As políticas de substituição são estudadas em [1], [2], [3], [7], [17], [22], [32], [33], [34],

Sistemas de caches para Web distribuídos e cooperativos são estudados em [4], [5], [8], [12], [18], [20], [21], [22], [23], [24], [27], [29], [30], [36], [37] e [38].

Sistemas de manutenção de coerência entre documentos originais e armazenados em caches são apresentados em [6] e [13].

Sistemas que realizam o *prefetching* de documentos são apresentados em [9], [16] e [28]. Esses sistemas utilizam o tempo que um usuário leva para ler o conteúdo de uma página para buscar páginas com alta probabilidade de serem acessadas num futuro próximo.

Em [10] e [11] são especificadas as caracterísiticas do ICP - *Internet Cache Protocol*, usado para troca de informações entre servidores de caches.

As características do comportamento dos usuários ao acessar a Internet, através da análise de arquivos de log é descrita em [14], [15],[26], [31] e [35].

Em [18] é descrito o projeto de um retransmissor de pacotes com cache para WWW, enquanto que o desempenho de caches particionados é discutido em [25] e [39].

4. O simulador

As principais características do simulador apresentado nesse trabalho são a forma de configuração, o sistema de manutenção e troca de mensagens, a manutenção de listas de arquivos e aplicação das políticas de substituição, o cálculo do atraso em transmissões de dados e processamento e a geração de resultados. Cada uma dessas características será discutida a seguir.

4.1. Configuração da malha

A configuração da malha a ser simulada é feita através de um arquivo texto que contém, numa linguagem predefinida, a descrição das características da malha. A figura 2 apresenta um exemplo de arquivo de configuração que poderia ser utilizado para simular a malha representada na figura 1.

```
structdef
  label Example simulation
  cachedef 1 100
    logfile ..\logs\log1.log
    resolveMiss 4 10
    sub 1000 0 0 lru
    cacheCoop 2 100
    cacheCoop 3 100
  cachedef 2 100
    logfile ..\logs\log2.log
    resolveMiss 4 10
    sub 1000 0 0 size
    cacheCoop 1 100
    cacheCoop 3 100
  cachedef 3 100
    logfile ..\logs\log3.log
    resolveMiss 4 10
    sub 1000 0 0 lru
    cacheCoop 1 100
    cacheCoop 2 100
  cachedef 4 500
    logfile NONE
    resolveMiss 0 0
    sub 3000 0 0 din
end.
```

Fig. 2 - Exemplo de arquivo de configuração

Os comandos e parâmetros utilizados no arquivo de configuração da figura 2 são descritos a seguir.

- **structdef -** Marca o início da definição de uma estrutura de caches a ser simulada.

- **label** - Permite a atribuição de um título à simulação, possibilitando um melhor acompanhamento durante a execução da mesma. Permite também uma melhor identificação durante a análise dos resultados das simulações.
- **cachedef *id cap*** - Cria um cache com o identificador ***id*** sendo executado num processador de capacidade ***cap***. Essa capacidade indica o poder de processamento do computador onde o cache está implantado. Essa medida de capacidade não possui unidade, sendo que seu valor numérico não é importante se considerado sozinho. Terá grande importância para permitir a comparação de atrasos devido a capacidade de processamento entre dois computadores de capacidades diferentes.
- **resolveMiss *id*** -Indica que requisições não resolvidas no cache local nem em caches vizinhos deve ser enviada ao cache cujo identificador é *id*. Isso significa que o cache *id* é um cache de nível superior.
- **logfile *arqName*** -Define que esse cache vai simular o recebimento das requisições do arquivo de log *arqName*. Caso não exista um arquivo de log, utiliza-se a palavra NONE.
- **sub *tam arqMin arqMax política*** - Define uma partição de tamanho ***tam*** do espaço de armazenamento do cache. Nessa partição serão armazenados os arquivos cujos tamanhos estão entre ***arqMin*** e ***arqMax***. No caso de ***arqMax*** ser 0, significa que não existe tamanho máximo para os arquivos que podem ser armazenados. O parâmetro ***política*** indica qual política de substituição será utilizada nessa partição do cache.
- **cacheCoop *id cap*** - Define que documentos não encontrados localmente devem ser procurados no cache de identificador *id* antes de encaminhar a requisição ao cache de nível superior. O parâmetro *cap* indica a capacidade da rede entre o cache local e o cache *id*.
- **end.** - Indica o final do arquivo de configurações dos caches.

4.2. Troca de mensagens

Cada cache da malha deve ser capaz de trocar mensagens com os outros caches para permitir a cooperação. A tabela da figura 3 apresenta os tipos de mensagens definidas no simulador. Essas mensagens correspondem a tipos de mensagens definidas no *Internet Cache Protocol* [10].

Tipo	Mensagem do ICP	Significado
1	ICP_OP_QUERY	Requisição de usuário
2	ICP_OP_QUERY	Procurar documento em cache vizinho
3		Procurar documento pedido por cache vizinho
4	ICP_OP_HIT / ICP_OP_MISS	Responder ao vizinho
5		Processar resposta de vizinho
6	ICP_OP_QUERY	Enviar requisição a cache de nível superior
7		Adicionar documento

Fig. 3 Mensagens definidas no simulador

Mensagens são trocadas entre os caches de modo a determinar se algum deles possui o documento requisitado. Se o documento é encontrado, ele é enviado ao usuário que o requisitou. Cada cache mantém uma fila de requisições a ser processada no futuro. As requisições lidas dos arquivos de log são transformadas em mensagens do tipo 1 e enfileiradas na fila de requisições do

cache. Ao encontrar uma requisição do tipo 1, é executada uma procura na lista de arquivos armazenados no cache. Se o arquivo não for encontrado, é enfileirada uma mensagem do tipo 2 na própria fila de requisições, indicando que aquela requisição deve ser enviada aos vizinhos. É enfileirada então, em cada vizinho, uma requisição do tipo 3. Quando é encontrada uma requisição do tipo 3 na fila, um cache entende que deve procurar um arquivo para um vizinho. A procura é feita e o resultado é colocado em uma mensagem do tipo 4. Quando uma mensagem do tipo 4 é encontrada, é enfileirada no cache que requisitou a pesquisa uma mensagem do tipo 5, que significa resposta a uma requisição. Quando algum cache vizinho responde que encontrou o documento, ele é enviado para o usuário. Caso nenhum vizinho contenha o documento, o cache enfilera na própria fila uma mensagem do tipo 6, que significa enviar a requisição ao cache de nível superior. Quando o documento requisitado é encontrado na malha, é enfileirada uma mensagem do tipo 7 em cada cache que enviou uma requisição a cache de nível superior. O processamento dessa mensagem faz com que o arquivo seja armazenado.

Além da fila de requisições, cada cache deve armazenar também quais requisições foram feitas a caches vizinhos e ainda não foram respondidas. Dessa forma, é possível determinar quantas respostas negativas faltam para concluir que nenhum vizinho possui o documento. São também armazenadas as requisições enviadas a caches de nível superior.

As filas de requisições são ordenadas segundo a hora em que a requisição foi feita. Dessa forma, é possível garantir que as requisições sejam processadas no simulador na mesma ordem que seriam processadas numa malha real de caches.

4.3. Políticas de Substituição

Nesse simulador podem ser simuladas as políticas LRU, SIZE e Dinâmica. As políticas LRU e SIZE são duas das políticas mais utilizadas em caches para Web. A política dinâmica foi desenvolvida nesse projeto, sendo o simulador descrito nesse artigo o primeiro a utilizá-la. As políticas que podem ser utilizadas são descritas a seguir.

- **LRU** *(Least Recently Used)* - É removido o documento menos recentemente acessado. Dessa forma, mantém-se no cache os documentos que foram acessados mais recentemente, sendo esses os documentos com maior probabilidade de serem reacessados.
- **SIZE** - Quando essa política é aplicada a um conjunto de arquivos, é removido o maior documento do conjunto. Dessa forma, tenta-se evitar que muitos arquivos pequenos tenham que ser removidos para a inserção de um único arquivo grande.
- **Dinâmica** - Essa política utiliza três filas de documentos. Essas filas contém os mesmos documentos, porém ordenados de maneira diferente. A primeira é ordenada utilizando a política LRU, a segunda usando a LFU e a última é ordenada através da política SIZE. Dessa forma, é possível considerar tanto o tamanho dos documentos quanto a freqüência com que foram acessados e o tempo desde seu último acesso, o que permite tomar uma decisão mais correta de qual documento remover do cache. A política dinâmica apresentou nos experimentos os melhores resultados, medidos em *Hit Ratio* quanto em *Byte Hit Ratio.*

Apesar de serem descritas nesse trabalho apenas as políticas LRU, SIZE e Dinâmica, a implementação do simulador permite que várias outras políticas, tais como as descritas em [36] sejam utilizadas para fins de análise de desempenho.

4.4. Particionamento do cache

O particionamento do cache permite uma melhor adequação do espaço de armazenagem às diferentes classes de tamanhos dos arquivos. O particionamento permite evitar que muitos arquivos pequenos sejam removidos para o armazenamento de um único arquivo grande [25]. Com uma configuração correta das partições é possível aumentar o *Hit Ratio* de um cache mantendo-se praticamente inalterado o valor do *Byte Hit Ratio.*

O particionamento possui a desvantagem de necessitar de um estudo aprofundado das classes de tamanhos dos arquivos de forma a definir uma correta divisão do espaço de armazenamento. Apresenta também o problema de fragmentação do espaço de armazenagem [39]. A fragmentação ocorre quando o espaço livre total do cache é suficiente para armazenar um arquivo porém, o espaço livre está fragmentado entre as partições. Assim, arquivos deverão ser eliminados do cache para a entrada de um novo.

Esse simulador é capaz de definir partições de qualquer tamanho para os caches. Em cada uma dessas partições pode ser utilizada uma política de substituição diferente.

4.5. Filas de documentos

Cada partição mantém uma fila dos nomes dos documentos armazenados na partição. A ordenação dessa fila depende da política de substituição utilizada pela partição. No caso de utilizar a política LRU, os documentos são ordenados por tempo de acesso, sendo que quanto mais próximo ao início da fila, mais antiga é a requisição ao documento. Essa ordenação é feita inserindo documentos apenas no final da fila. Para remover o elemento mais antigo da fila basta remover o primeiro elemento.

A fila SIZE é ordenada de acordo com o tamanho dos documentos. Nessa fila, quanto mais próximo da cabeça da fila, maior o documento. Nessa fila, para remover o maior documento, basta remover o primeiro da fila. Porém, ao contrário da fila LRU, deve-se procurar a posição correta com relação ao tamanho para inserir cada novo documento.

No caso da política dinâmica, existe uma fila de documentos com três ordenações, feitas pelas políticas LRU, LFU e SIZE. O modelo utilizado na política de substituição dinâmica está representado esquematicamente na figura 4.

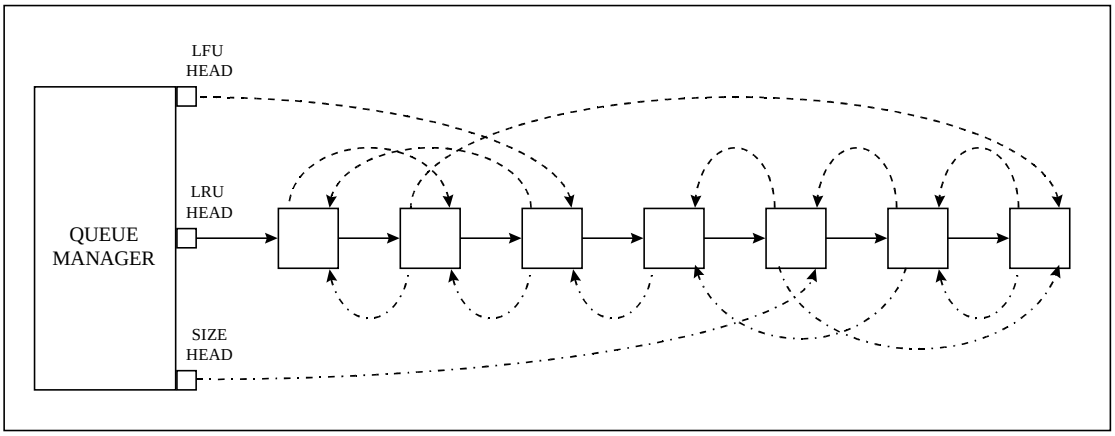

Fig. 4 Modelo da política de substituição dinâmica

Cada elemento da fila é ligado ao seu antecessor e sucessor na ordenação LRU, LFU e SIZE. Dessa forma, é possível escolher dinamicamente qual política aplicar para efetuar cada remoção de documento. Devido à essa característica, essa política possibilita um melhor desempenho do sistema de caches.

4.6. Cálculo de atrasos

Para simular os atrasos ocorridos na transmissão de mensagens e documentos é utilizada uma ordenação de eventos. Essa ordenação faz com que os eventos aconteçam no simulador na mesma ordem que aconteceriam num sistema de caches reais. Para isso, é associado à cada evento um *timestamp*, que indica o tempo em que o evento ocorreu. Assim, a cada operação é calculado o tempo que essa operação levaria num cache real, considerando as capacidades de rede e processamento. Esse tempo é somado ao *timestamp* do evento e ele é então colocado na fila do cache que irá processá-lo em ordem crescente de *timestamp*.

A cada iteração do simulador, é analisado qual evento tem o menor *timestamp* dentre todos os eventos nas filas de todos os caches do sistema. Processando sempre o evento de menor *timestamp* é possível garantir uma correta relação de antecedência entre os eventos, simulando corretamente a operação de um sistema de caches reais.

4.7. Resultados da simulação

Os resultados da simulação são armazenados em um arquivo texto. Esse arquivo contém informações tais como o número de requisições processadas, o *hit ratio* e o *byte hit ratio* de cada cache e de todo o sistema e também os tempos de atraso em processamento e comunicação.
A figura 5 apresenta uma parte de um arquivo de resultados relativo à simulação de uma malha semelhante a da figura 1.

```
Simulação  1:  Exemplo de malha

Numero de requisições : 1245157
Numero total de Hits: 644507 (51.761%)
Numero total de Miss: 600650 (48.239%)

Total de bytes requisitados: 4426.061MB
Total de bytes encontrados : 1772.698MB (40.051%)

Estatísticas por Caches e Partições
Cache ID: 1
  Total do cache:
    Numero de Hits: 39026
    Numero de Miss: 643233
    Bytes encontrados     : 220.298 MB
    Bytes não encontrados : 2989.098 MB

Cache ID: 2
    Numero de Hits: 238127
    Numero de Miss: 657249
    Bytes encontrados     : 1002.683 MB
    Bytes não encontrados : 3107.734 MB

Cache ID: 3
```

```
    Numero de Hits: 334470
    Numero de Miss: 661110
    Qtde de bytes encontrados     : 269.344 MB
    Bytes não encontrados : 3160.820 MB

Cache ID: 4
    Numero de Hits: 35696
    Numero de Miss: 600650
    Bytes encontrados     : 311.477 MB
    Bytes não encontrados : 2652.714 MB

Mensagens trocadas: 3893112 (3.127 por requisição)
Requisições enviadas a vizinhos : 2656116 (2.133 por requisição)
Requisições enviadas a cache pai: 636346 (0.511 por requisição)

Tempo médio de atraso em transmissões : 315.271 mili-segundos
Tempo médio de atraso em processamento: 38.474 micro-segundos
Tempo médio de espera do usuário: 1.614 segundos
```

Fig. 5 Exemplo de arquivo de resultado

4.8. Execução em computadores paralelos

Devido ao grande volume de processamento necessário para executar simulações, é interessante utilizar computadores paralelos. Porém, para aproveitar a capacidade de processamento dos computadores paralelos foi necessário desenvolver uma versão paralela do simulador. Essa versão paralela foi desenvolvida utilizando o MPI (Message Passing Interface). [41], [42].

Na versão paralela, a cada processo iniciado é associado um identificador. Esse identificador é um número inteiro e o primeiro processo recebe identificador 0, o segundo processo recebe identificador 1 e assim por diante. Além disso, cada processo sabe quantos processos estão sendo executados ao mesmo tempo.

Ao ler o arquivo de configuração, cada processo só executa as simulações quando a seguinte condição acontece: (NumSim *mod* NumProc = ID), onde NumSim é o número da simulação, NumProc é o número total de processos e ID é o identificador do processo.

Por exemplo, no caso de existirem 3 processos para realizar 8 simulações, o processo 1 vai executar as simulações 1, 4 e 7; o processo 2 executará as simulações 2, 5 e 8 e o processo 3 as simulações 3 e 6. Dessa forma, é possível aproveitar o podem de processamento de um computador paralelo para realizar um número muito maior de simulações.

A execução paralela das simulações possui uma alta granularidade, pois as execuções são independentes. Além disso, como não existe troca de mensagens entre os processadores, o poder de processamento de cada processador é usado apenas para executar simulações.

5. Simulação da RNP brasileira

Para demonstrar a utilização do simulador, foram feitas simulações baseadas da malha da RNP (Rede Nacional de Pesquisa) brasileira [43]. Foi utilizado na simulação o *backbone* RNP2, que começou a ser implantado em julho de 2000. Os Pontos de Presença (PoPs) que concentram maior fluxo de tráfego de dados utilizam conexões com tecnologia ATM. A tecnologia *Frame*

Relay foi utilizada como solução de melhor relação custo/benefício nos PoPs que apresentam menor taxa de tráfego de dados.

O RNP2 possui atualmente quatro conexões internacionais de 2 Mbps e 23 Pontos de Presença (PoPs) instalados nas principais cidades e capitais do país. A velocidade das Portas de Acesso dos PoPs, de até 155 Mbps, garante o atendimento da soma das diversas conexões virtuais estabelecidas (VP) e permite a elevação da largura de banda dessas conexões na medida em que a demanda justificar esse aumento de velocidade. A malha da RNP brasileira, incluindo os PoPs é apresentada na figura 6.

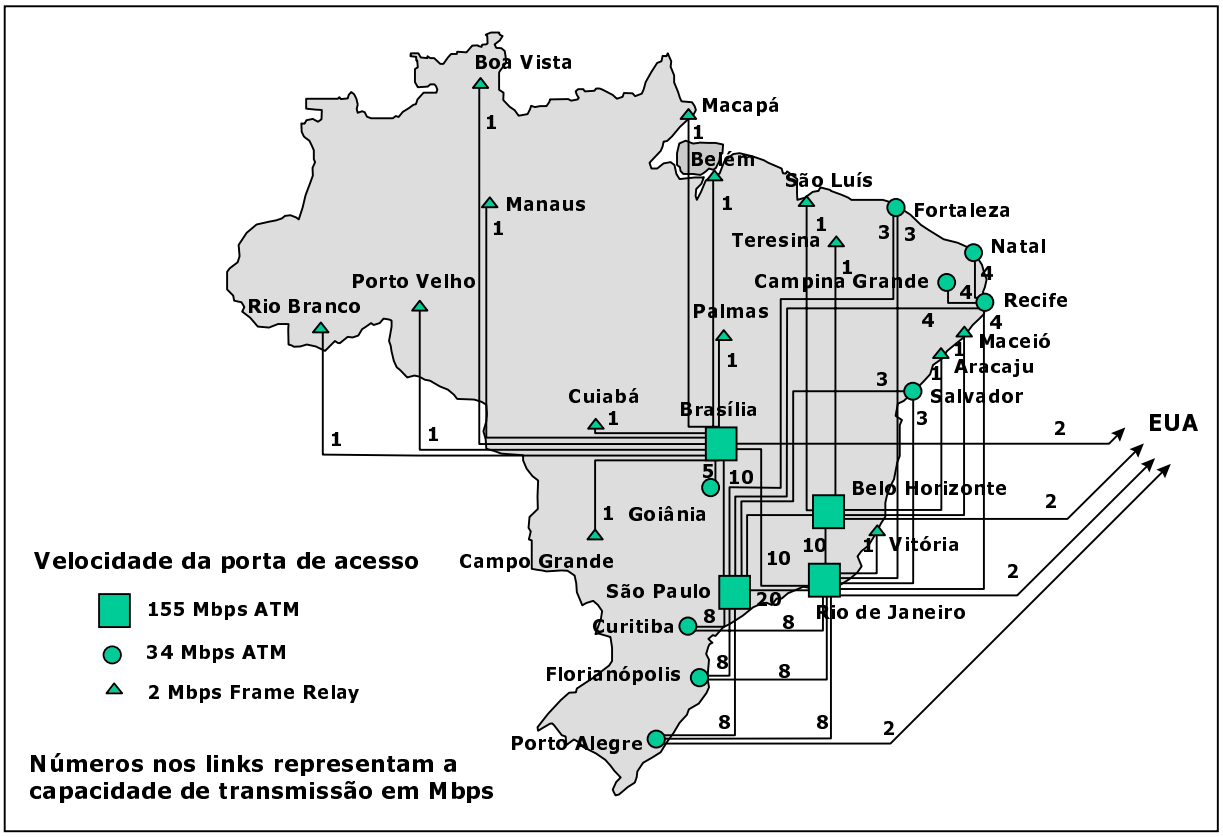

Fig. 6 Malha da RNP brasileira

Nas simulações foram utilizadas as características relativas à arquitetura e capacidades da RNP2. A lista das cidades cujos caches pertencem à malha simulada são apresentados na tabela da figura 7. São apresentados também os nomes das localidades onde estão situados os caches de nível hierárquico superior de cada cache bem como a capacidade de transmissão da rede que interliga-os.

Cache	Cache superior	Link ao cache superior
São Paulo	Rio de Janeiro	20 Mbps
Florianópolis	São Paulo, Rio de Janeiro	16 Mbps
Curitiba	São Paulo, Rio de Janeiro	16 Mbps
Recife	São Paulo, Rio de Janeiro	8 Mbps
Salvador	São Paulo, Rio de Janeiro	6 Mbps
Fortaleza	São Paulo, Rio de Janeiro	6 Mbps

Goiânia	Brasília	5 Mbps
Natal	Recife	4 Mbps
Campina Grande	Recife	4 Mbps
Brasília	USA	2 Mbps
Rio de Janeiro	USA	2 Mbps
Belo Horizonte	USA	2 Mbps
Porto Alegre	USA	2 Mbps
Belém	Brasília	1 Mbps
Manaus	Brasília	1 Mbps
Campo Grande	Brasília	1 Mbps
Cuiabá	Brasília	1 Mbps
Vitória	Rio de Janeiro	1 Mbps
São Luiz	Belo Horizonte	1 Mbps
Teresina	Belo Horizonte	1 Mbps
Maceió	Belo Horizonte	1 Mbps
Aracaju	Belo Horizonte	1 Mbps
Rio Branco	Brasília	1 Mbps
Porto Velho	Brasília	1 Mbps
Boa Vista	Brasília	1 Mbps
Macapá	Brasília	1 Mbps
Palmas	Brasília	1 Mbps

Fig. 7 Pontos de presença simulados.

Um problema muito importante é qual *arquivo de log* utilizar nas simulações. Devido à impossibilidade de obter arquivos de log reais de cada servidor a ser simulado, foram utilizados os *arquivos de log* provenientes de servidores de cache do sistema IRCACHE [40] devido à grande disponibilidade de arquivos. Esses arquivos de log foram utilizados em vários trabalhos de pesquisa relacionados a caches para Web, apresentando por isso uma boa referência na análise dos resultados obtidos.

Apesar dos *arquivos de log* do sistema IRCACHE não representarem com absoluta precisão as requisições feitas a servidores brasileiros, as características dos documentos requisitados, tais como tamanho e freqüência, são semelhantes. Assim, pode-se usar os arquivos do sistema IRCACHE sem alterar as conclusões obtidas nas simulações. *Os arquivos de log* utilizados nas simulações desse artigo são relativos a dez dias de uso dos servidores de cache.

A análise dos *arquivos de log* utilizados mostrou que um cache de tamanho infinito conseguiria uma máximo de 67% de *Hit Ratio*. Cache de tamanho infinito é aquele cuja capacidade de armazenamento é superior à soma dos tamanhos de todos os diferentes documentos que devem, em algum tempo, serem armazenados. Para os *arquivos de log* utilizados, qualquer cache de capacidade maior ou igual a 42 Giga Bytes pode ser considerado de tamanho infinito.

O primeiro experimento analisou o impacto da variação do tamanho do espaço de armazenamento do cache no desempenho. Foi também avaliado o impacto no desempenho da variação da política de substituição utilizada. Os resultados são apresentados na figura 8. Esses resultados foram medidos em *Hit Ratio* e representam o desempenho de toda a malha da RNP2.

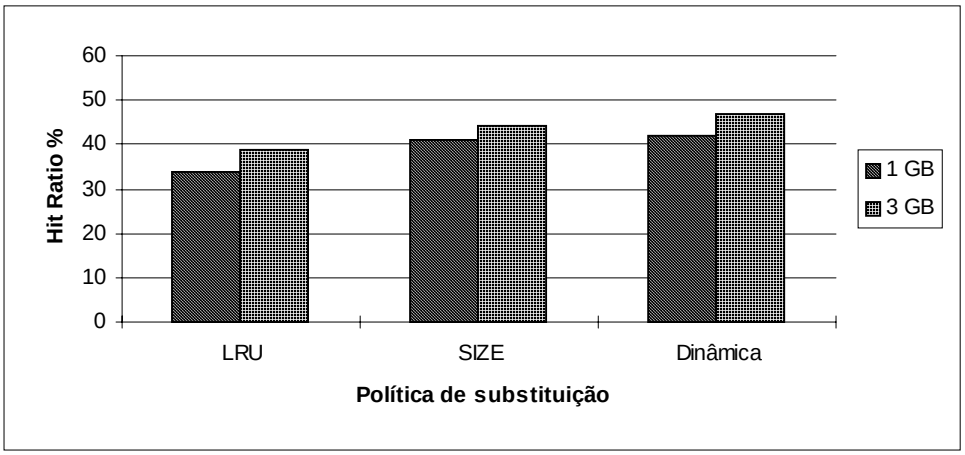

Fig. 8 Variação de desempenho relativo a tamanho e política

Através da figura 8, pode-se notar que apesar do aumento do espaço de armazenamento gerar uma melhora no desempenho do cache, a utilização de uma política de substituição melhor causa uma melhora no desempenho mais significativa. Nessa simulação, verificou-se que o tempo médio de espera por um documento foi de 44 milisegundos.

O segundo experimento testou a possibilidade em que houvesse a cooperação entre todos os servidores de cache. Os resultados dessa simulação são apresentados na figura 10.

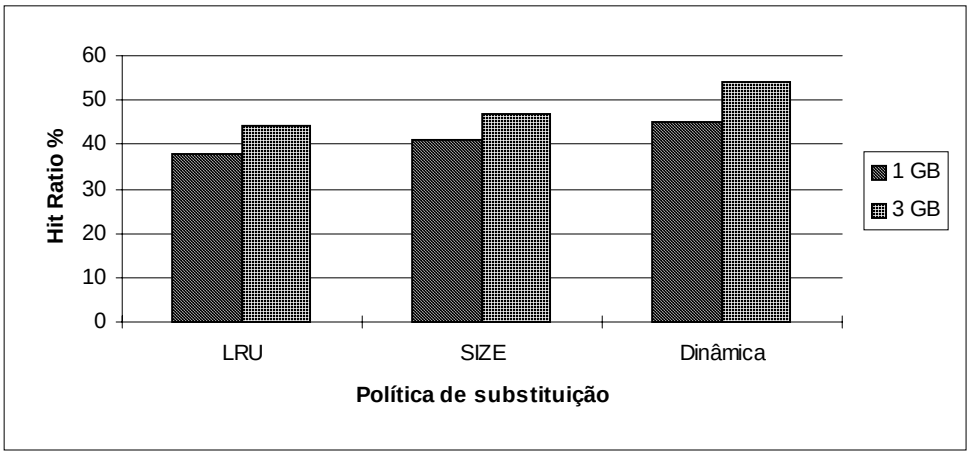

Fig. 9 Cooperação entre todos os caches

Através da figura 9 observa-se que o desempenho do sistema com cooperação entre todos os caches foi melhor que o do sistema sem cooperação nenhuma. Porém, houve um aumento de 980% no número de mensagens trocadas entre servidores de cache. Esse grande aumento no número de mensagens trocadas faz com que o usuário experimente um alto tempo de espera pois, nesse caso, o usuário terá que esperar a troca de mensagens entre todos os servidores, inclusive entre aqueles ligados por redes de baixa capacidade. Nessa simulação, o tempo de espera médio foi de 1190 milisegundos, que é um tempo impraticável em redes reais. Assim, pode-se concluir que utilizar cooperação entre caches ligados por redes de baixa capacidade de transmissão não é interessante.

O último experimento consistiu em medir o desempenho quando utilizada a cooperação apenas entre alguns dos caches do sistema. Foi então definida a cooperação entre os caches cuja

localização é apresentada na figura 10. Note que a cooperação foi definida apenas entre caches ligados por redes de alta capacidade.

Cache	Coopera com
São Paulo	Brasília, Belo Horizonte e Rio de Janeiro
Rio de Janeiro	Belo Horizonte e São Paulo
Belo Horizonte	Rio de Janeiro e São Paulo
Brasília	Rio de Janeiro e São Paulo

Fig. 10 Alguns caches em cooperação

O desempenho total da malha da RNP2, com cooperação apenas entre os caches ligados por redes de maior capacidade é apresentado na figura 11.

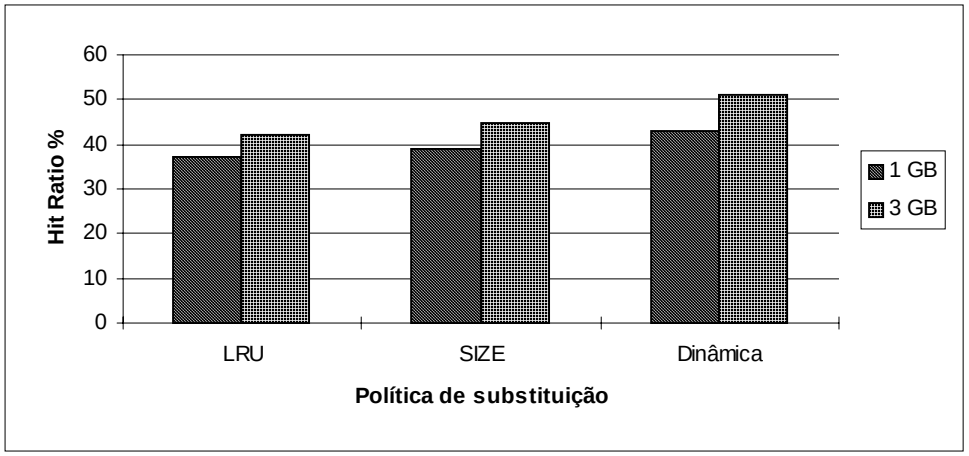

Fig. 11 Simulação com caches cooperativos.

A utilização da cooperação apenas entre caches ligados por redes de alto desempenho causou um aumento de 181% no número de mensagens trocadas entre os caches, em relação à primeira simulação. Por outro lado, houve uma diminuição do tempo de espera do usuário de 37%. Isso aconteceu porque o aumento do desempenho em *Hit Ratio* compensou o aumento do número de mensagens. Assim, conclui-se que a cooperação entre os caches melhorou o desempenho do sistema como um todo. Porém, note que não é garantido que os resultados apresentados na figura 11 são os melhores possíveis. Para determinar qual configuração maximiza o desempenho do cache são necessárias muitas outras simulações, testando muitas outras possibilidades.

Apesar dos caches simulados serem de pequeno tamanho quando comparados com caches reais de alto desempenho, cujo tamanho do espaço de armazenamento chega a centenas de Giga Bytes, os resultados obtidos são válidos. Isso ocorre porque tanto o *hit ratio* quanto o *byte hit ratio* de um cache para web crescem logaritmicamente em função do tamanho do cache [38]. Assim, salvo as devidas proporções as conclusões são as mesmas para caches maiores. Além disso a análise de resultados para caches pequenos tem sido considerada com grande relevância porque esse tamanho de cache pode ser colocado na memória RAM de servidores. Dessa forma, esses servidores podem responder a requisições sem fazer nenhum acesso à dispositivos de memória secundária[37].

As simulações foram executadas utilizando um computador paralelo *Beowulf 64*, formado por 64 computadores Intel, cada um com 256 Mega Bytes de memória RAM, rodando o sistema

operacional Linux Red Hat. Esse computador pertence ao Laboratório de Alto Desempenho do Instituto de Computação da Unicamp. A utilização de um simulador capaz de aproveitar o poder de processamento de um computador paralelo possibilitou o processamento de quantidades de dados que seriam praticamente impossíveis de serem analisados em um computador monoprocessado.

6. Conclusões e trabalhos futuros

O simulador paralelo apresentado nesse artigo mostrou ser uma ferramenta de grande importância para a análise de malhas de caches distribuídos. Usando esse simulador, é possível testar várias configurações dos parâmetros que influem no desempenho de sistemas de caches para web. Esse simulador é capaz de utilizar o alto poder de processamento de computadores paralelos, possibilitando a execução de um grande número de simulações. Esse aumento do número de simulações gera uma maior quantidade de resultados, o que permite uma análise mais apurada dos sistemas simulados.

Esperamos esse simulador seja de grande utilidade no projeto e análise de sistemas de caches para Web distribuídos e cooperativos.

Referências bibliográficas

[1] S. Willians, M. Abrams, C. R. Standridge, G. Abdulla, E. A. Fox: Removal policies in network caches for World Wide Web Documents. *ACM SIGCOMM 96*, 293-305, 08/1996.

[2] P. Lorenzetti, L. Rizzo, L. Vicisano: Replacement policies for a proxy cache. Technical report, Universita di Pisa, Italy, 10/1996.

[3] H. Kim, K. Chon: Update policies for network caches. Technical Memo, Department of Computer Science, Korea Advanced Institute of Science and Technology, 07/1998.

[4] M. Kurcewicz, W. Sylwestrzak, A. Wierzbicki: A distributed WWW cache. Proceedings of the Third International WWW Caching Workshop, Manchester, England, 06/ 1998..

[5] R. Malpani, J. Lorch, D. Berger: Making World Wide Web caching servers cooperate. 4th International World-wide Web Conference,107-117, 12/1995.

[6] A. Dingle:. Web cache coherence. *Computer Networks and ISDN systems*, vol. 28, N° 7-11, 907-920, 1996.

[7] M. Abrams, C. R. Standridge, G. Abdulla, S. Willians, E. A. Fox: Caching proxies: Limitations and potentials. 4th International World-wide Web Conference, 119-133, 12/1995.

[8] I. Melve, L. Slettjord, H. Bekker, T. Verschuren: Building a Web caching system - architectural considerations. Desire Project, 03/1997. URL:http://www.uninett.no/prosjekt/desire/arneberg

[9] K. Chinen, S. Yamaguchi: An interactive prefetching proxy server for improvement of WWW latency. Proceedings of the Seventh Annual Conference of the Internet Society (INET'97), Kuala Lumpur, 06/1997.

[10] D. Wessels, K. Claffy: Internet Cache Protocol (ICP), version 2. Networking Working Group, RFC 2186, 09/1997

[11] D. Wessels, K. Claffy: Application of Internet Cache Protocol (ICP), version 2.

Networking Working Group, RFC 2187, 09/1997

[12] A. Chankhunthod, P. D. Danzig, C. Neerdaels, M. Schwartz, K. J. Worrell: A Hierarchical Internet Object Cache. USENIX Annual Technical Conference, 153-164, 1996

[13] J. Gwertzman, M. Seltzer: World Wide Web Cache Consistency. USENIX Annual Technical Conference, 141-152, 1996

[14] C. R. Cunha, A. Bestavros, M. E. Crovella: Characteristics of WWW Client-based *Traces*. Technical Report TR-95-010, Boston University Computer Science Department, 06/1995.

[15] V. Almeida, A. Bestavros, M. Crovella, A. Oliveira: Characterizing Reference Locality in the WWW. Proceedings of the Fourth International Conference on Parallel and Distributed Information Systems, Miami Beach, Florida, USA, 92-103, 12/1996

[16] N. P. Venkata, J. C. Mogul: Using predictive prefetching to improve World Wide Web latency. Computer Communication Review, 26(3),22-36, 07/1996.

[17] D. Povey, J. Harrison: A Distributed Internet Cache. Proceedings of the 1997 Australasian Computer Science Conference. Sydney, Australia, 1997.

[18] S. Glassman: A Caching Relay for the World Wide Web. Proceedings of the First International Conference on the WWW, 1994. Also published in "Computer Networks and ISDN Systems", volume 27 (1994), Number 2, 11/1994.

[19] M. Reddy, G. P. Fletcher: Intelligent web caching using document life histories: A comparison with existing cache management techniques. Proceedings of the Third International WWW Caching Workshop, Manchester, England, 06/1998.

[20] M. Dahlin, R. Wang, T. E. Anderson, D. A. Patterson: Cooperative Caching: Using Remote Client Memory to Improve File System Performance. Proceedings of the First USENIX Symposium on Operating Systems Design and Implementation (OSDI), Monterey, California, 67-280, 11/1994

[21] C. Grimm, J. S. Vöckler, H. Pralle: Load and Traffic Balancing in Large Scale Cache Meshes. Computer Networks 30, 16-18, 1998

[22] Michael Rabinovich, Jeffrey S. Chase, Syam Gadde: Not all Hits are Created Equal: Cooperative Proxy Caching Over a Wide-Area Network. Computer Networks 30, 22-23, 1998

[23] A. Heddaya, S. Mirdad: WebWave: Globally Load Balanced Fully Distributed Caching of Hot Published Documents. Proceedings of the 17th International Conference on Distributed Computing Systems, Baltimore, MD, USA, 27-30, 05/1997

[24] Andrew Cormack. Web caching. Url: http://www.jisc.ac.uk/acn/caching.html, 1996.

[25] C. D. Murta, V. A. F. Almeida, Jr. W. Meira: Analyzing performance of partitioned caches for the WWW. In Proceedings of the Third International WWW Caching Workshop, Manchester, England, 06/1998.

[26] M. E. Crovella, A. Bestavros: Explaining World Wide Web Traffic Self-Similarity. Technical Report TR-95-015, Boston University, CS Dept, Boston, MA, 10/1995.

[27] R. B. Tewksbury: Is the Internet heading for a cache crunch? In Proceedings of the Eighth Annual Conference of the Internet Society (INET'98), Geneva, Switzerland, 07/1998.

[28] A. Bestavros: Using Speculation to Reduce Server Load and Service Time on the WWW. Proceedings of the 1995 International Conference on Information and Knowledge Management, Baltimore, Maryland, USA, 403-410, 12/1995

[29] C. Grimm, J.-S. Vöckler, H. Pralle: Request Routing in Cache Meshes. Computer Networks 30, 2269-2278, 1998

[30] H. Inoue, T. Sakamoto, S. Tamaguchi: Webhint: An automatic configuration mechanism for optimizing World Wide Web Cache system utilization. Proceedings of the Eighth Annual Conference of the Internet Society (INET'98), Geneva, Switzerland, 07/1998.

[31] R. Morris and D. Lin, Variance of Aggregated Web Traffic. IEEE INFOCOM 2000, Tel Aviv, 360-366, 03/2000

[32] A. Vakali: A Web-based evolutionary model for Internet Data. 2nd Int. Workshop on Network-based Information Systems, Florence, 08/1999.

[33] T. Kelly, Y. Chan, S. Jamin, and J. Mackie-Mason. Biased Replacement Policies for Web Caches: Differential Quality-of-Service and Aggregate User Value. Proceedings of the 4th International Web Caching Workshop, San Diego, California , 04/1999.

[34] J. Almeida, M. Dabu, A. Manikutty, P. Cao. Providing Differentiated Levels of Service in Web Hosting Services. Proc. Workshop on Internet Server Performance, 06/1998.

[35] L. Breslau, P. Cao, L. Fan, G. Phillips, S. Shenker: Web Caching and Zipf-like Distributions: Evidence and Implications. Proceedings of IEEE INFOCOM'99, New York, 03/1999.

[36] P. Cao, S. Irani: Cost-Aware WWW Proxy Caching Algorithms. USENIX Symposium on Internet Technologies and Systems, 1997

[37] P. Cao, S. Irani: Cost-Aware WWW Proxy Caching Algorithms. Proceedings of the USENIX Symposium on Internet Technoloy and Systems, 193-206, 12/1997.

[38] S. Jin, A. Bestavros: Greedy-Dual* Web Caching Algorithm: Exploiting the Two Sources of Temporal Locality in Web Request Streams. Web Caching Workshop (available from Boston University Department of Computer Science, technical report 2000-011), 2000.

[39] C. D. Murta, PhD Thesis. Deparatmento de Ciência da Computação, UFMG, Brasil, 1999.

[40] Examples of Squid Log Files. URL: ftp:// ftp.ircache.net/Traces/, 01/2000.

[41] The Message Passing Interface (MPI) standard.
Url: http://www-unix.mcs.anl.gov/mpi/

[42] W. Gropp, E. Lusk, A. Skjellum: Using MPI, 2nd Edition. MIT Press.
Available from Url: ftp://ftp.mcs.anl.gov/pub/mpi/using/UsingMPI.tar.gz

[43] Sistema de servidores proxy da RNP brasileira. URL: proxy.rnp.br, 2001