Variação dinâmica da QoS oferecida por 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. *Diversos fatores podem influir no desempenho de caches para Web. Por isso, uma política de substituição em um cache para Web, escolhida inicialmente pelo seu bom desempenho, pode ter problemas devido à sua incapacidade em se adaptar a novas exigências de desempenho. Assim, é necessário que a estratégia de gerenciamento do espaço do cache consiga se adaptar às variações de características da demanda por documentos.*

Esse artigo apresenta o Gerenciador Dinâmico de Espaço (GDE), que além de apresentar desempenho na predição de utilidade futura comparável às melhores políticas de substituição existentes, possibilita ao cache oferecer diferentes qualidades de serviço (QoS) tanto a servidores Web quanto a clientes específicos.

Abstract. *Several factors can influence on the web cache performance. Because that, a replacement policy which was initially chosen due to its high performance can suffer of performance problems due to its incapacity of self-adapting to the new performance characteristics required by the clients. Thus, it is necessary that the cache space management strategy be capable of dynamically adapt itself to respond to new requirements from its clients.*

This paper presents the Dynamic Space Manager (GDE), whose quality in predicting files future usefulness is good as the best existent replacement policies. It also makes possible to the cache to offer different Qualities of Service (QoS) to web servers and specific clients.

1. Introdução

A World Wide Web é um sistema de distribuição de documentos baseado 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 alta carga ocasionada pela grande quantidade de requisições. A sobrecarga na rede e nos servidores faz com que o usuário dos serviços oferecidos pela Web experimente tempos de espera muito maiores do que os tempos experimentados se a carga na rede estivesse baixa [Malpani95].

O cache de documentos 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 [Kim98].

Caches proxy podem diminuir drasticamente a carga na rede. Por exemplo, segundo [Willians96], em casos especiais, é possível que um cache estrategicamente colocado consiga diminuir em mais de 80% a carga na rede. Porém, caches possuem limitações: em primeiro lugar, apesar de existirem trabalhos que estudam a possibilidade de armazenamento de documentos dinâmicos [Cao99], caches trabalham apenas com documentos estáticos e que, preferencialmente, sejam modificados com pouca freqüência em seus servidores [Abrams95]. Em segundo lugar, garantir a consistência entre os documentos no cache e nos servidores causa um aumento na carga das redes devido às mensagens trocadas entre os servidores e os caches [Dingle96].

Quando um cliente requisita um documento a um servidor de cache para Web, esse documento pode ou não estar no cache. 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*.

Para medir a eficiência de um sistema de cache para Web são utilizados os métodos de taxa de acerto (*hit ratio*)e taxa de acerto por *byte* (*byte hit ratio*). 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.

Políticas de substituição são algoritmos aplicados quando o tamanho do documento que deverá ser gravado no cache exceder o espaço livre do cache. Nesse caso, são removidos os documentos considerados mais dispensáveis até que o espaço seja suficiente para armazenar o novo documento.

Atualmente, existem muitas publicações que apresentam e discutem diferentes estratégias de gerenciamento do espaço disponível para cache de arquivos. A maioria desses artigos apresenta novas políticas de substituição, com o objetivo de maximizar o desempenho dos caches em uma ou mais métricas específicas. Alguns trabalhos propõem um gerenciamento mais inteligente do espaço disponível. Um exemplo é o modelo PART, proposto em [Murta98], que propõe o particionamento do espaço disponível. Cada divisão do espaço é então destinado a armazenar uma classe diferente de arquivos. Essas classes podem ser definidas, por exemplo, segundo critérios de tamanho e tipo dos arquivos.

Entretanto, muitas vezes deseja-se que um cache privilegie um grupo de arquivos ou servidores. O cache é um bom lugar para implantar estratégias que permitam oferecer diferentes QoS [Kelly99]. Isso porque ele fica estrategicamente instalado no fluxo de dados e pode ser utilizado como mediador entre os diversos servidores que buscam sempre maximizar a qualidade dos serviços oferecida aos seus usuários [Chan99].

Para privilegiar servidores ou conjuntos de arquivos específicos, o cache pode utilizar um valor de utilidade fornecido pelos servidores para estimar a utilidade futura dos arquivos e decidir quais remover do cache [Kelly99]. Do ponto de vista do usuário, é importante que o cache proporcione uma diminuição no tempo de resposta a requisições feitas [Afonso98]. Para que isso aconteça é necessário que a política de substituição implementada consiga prever quais documentos serão acessados posteriormente [Kelly99a].

Porém, as características da demanda de QoS dos serviços prestados pela Web não são estáticas. No caso de caches para Web, elas dependem de diversos fatores tais como o número de clientes, a capacidade de transmissão das redes, diferenças entre perfis de usuários, a hora e dia da semana, a época do ano e a variação da popularidade dos arquivos requisitados. Assim, o desempenho de uma política escolhida inicialmente pode diminuir drasticamente devido à incapacidade da política em se adaptar a novas características exigidas para um bom desempenho. Por isso, é necessário que a estratégia de gerenciamento do espaço para cache consiga se adaptar às variações de características da demanda por documentos.

Com esse objetivo, esse artigo apresenta e propõe a utilização do Gerenciador Dinâmico de Espaço (GDE). Essa estratégia pode alternar dinamicamente os parâmetros utilizados para estimar a utilidade futura dos arquivos armazenados. Assim, ela permite variar dinamicamente o perfil dos arquivos com maior preferência de serem mantidos no cache. Dessa forma, é possível direcionar o trabalho do cache para priorizar a manutenção de arquivos com características específicas desejadas, podendo assim variar a QoS oferecida aos clientes do sistema.

2. Trabalhos Anteriores

Recentes estudos mostraram que tanto o *hit ratio* quanto o *byte hit ratio* de um cache para Web crescem logaritmicamente em função do tamanho do cache. Isso implica que um algoritmo que aumenta o *hit ratio* e/ou o *byte hit ratio* em alguns pontos percentuais pode ser substituído pelo aumento do espaço em disco. Porém, atualmente, o crescimento da Web é muito maior que o da tecnologia de fabricação de memória. Isso implica que a melhor maneira de aproveitar um cache para web é através de técnicas eficientes de gerenciamento de espaço [Murta98].

Uma breve descrição das principais políticas de substituição estudadas é apresentada a seguir:

- FIFO (First In First Out) - Remove-se o documento mais antigo.
- LRU (Least Recently Used) - Remove-se o menos recentemente acessado.
- LRU-MIN [Abrams95] - Similar ao LRU, mas com considerações sobre o tamanho do documento, onde são removidos preferencialmente os documentos maiores.
- LRU-Threshold [Abrams95] - Similar à LRU, com a diferença de que documentos maiores que um limite nunca são armazenados no cache.
- LFU - (Least Frequently Used) - Remove-se o documento usado com menor freqüência.
- SIZE - Remove-se o maior documento.
- Hyper-G - Refinamento da política LFU, com considerações sobre o último acesso e tamanho.
- Pitkow/Recker - Usa LRU a não ser que todos os documentos tenham menos que 24 horas. Nesse caso, é aplicada a política SIZE.
- Lowest Latency First - Elimina o documento que apresenta a menor latência ao servidor de origem.

- LFU with Dynamic Aging (LFU-DA) [Cao97] - Utiliza a freqüência de acesso e o custo para trazer cada documento, associado a uma técnica de envelhecimento de documentos, para organizar o cache.
- Greedy-Dual – Adaptação de um algoritmo de gerenciamento de memória virtual para caches para Web. Os documentos têm o mesmo tamanho. A utilidade dos documentos para o cache depende do custo para trazê-los e do tempo desde o último acesso.
- GreedyDual-Size - (GDS) [Cao97] – Aprimoramento da política Greedy-Dual, com considerações aos tamanhos dos documentos. A importância dos documentos pode ser proporcional ao número de pacotes IP necessários para transmitir o documento. Nesse caso, a GDS é chamada GDS(packets). Se for considerado que o custo de transmissão é igual para todos os documentos, a GDS é chamada GDS(1) e é equivalente à Greedy-Dual.
- GreedyDual-Size with Frequency (GDSF) [Cao97] - Similar à GreedyDual-Size, porém utiliza também a freqüência de acesso a cada documento para decidir qual será removido.
- GD* [Jin00] – Aprimoramento da GDS, com considerações à popularidade esperada dos documentos. Assim como a GDS, pode-se definir GD*(1) e GD*(packets), dependendo da importância dos tamanhos dos documentos.

Estudos sobre quais documentos devem ser removidos do cache indicam que quanto mais informações sobre freqüências de acesso, tamanho e idade das requisições, melhor será a decisão de qual documento remover do cache.

A proposta de novas estratégias de gerenciamento de espaço em caches para Web é feita utilizando heurísticas baseadas principalmente em características do fluxo de requisições. Exemplos desse estudo podem ser encontrados em [Cunha95], [Almeida96] e [Crovella95].

Em [Afonso98] é apresentada uma maneira de medir a QoS de um cache para Web através do tempo de resposta do mesmo. Uma proposta de como prover diferentes QoS em um servidor Web é apresentada em [Almeida98]. Em [Kelly99] é proposta uma simples generalização da política LFU, que permite ao servidor Web atribuir valores de importância para a manutenção de arquivos nos caches.

A estratégia introduzida em [Kelly99a] utiliza a predição e estimação de utilidade futura de arquivos para calcular a importância de manter arquivos no cache. O modelo sugerido por [Chan99] adota a perspectiva de que os arquivos mais valiosos para os clientes devem ser mantidos no cache. Nesse modelo, os servidores requisitam espaço no cache para armazenar os arquivos de sua escolha.

3. O Gerenciador Dinâmico de Espaço (GDE)

Os primeiros estudos sobre o fluxo de documentos verificaram que ele obedece a uma distribuição de cauda pesada e que as transferências apresentam auto-similaridade quando a demanda por documentos é alta [Crovella95]. A auto-similaridade *(self-similarity)* é uma propriedade associada a fractais, onde o objeto apresenta a mesma forma, não importando a escala na qual é visto. Estudos posteriores mostraram que o fluxo de documentos obedece à lei de Zipf [Breslau99].

A lei de Zipf diz que a probabilidade relativa de acontecer uma requisição ao *i-ésimo* documento mais popular é proporcional a $^1/_i$. A auto-similaridade e a obediência à lei de Zipf são as principais características analisadas quando do desenvolvimento de políticas de substituição, pois essas características podem ser traduzidas em valores de freqüência de acesso, intervalo entre os acessos e tamanho dos documentos.

A freqüência com que um documento foi acessado no passado tem fortes implicações na probabilidade de ser acessado novamente no futuro. Para retirar do cache documentos com baixa freqüência de acessos pode ser usada a política LFU.

O tempo desde o último acesso ao documento também é um fator muito importante no cálculo da probabilidade de um documento do cache ser acessado novamente. Documentos acessados mais recentemente têm maiores probabilidades de serem acessados novamente. A política LRU é utilizada para remover documentos antigos do cache.

O tamanho dos documentos armazenados no cache é também importante no desempenho do cache. A remoção de documentos grandes evita que o cache seja preenchido por poucos documentos grandes ao invés de muitos documentos de tamanhos médios e pequenos, o que diminui o desempenho do cache em *hit ratio*. A remoção dos maiores documentos é implementada pela política SIZE.

Para prover diferentes QoS, o GDE pode usar vários diferentes critérios para ordenar filas de prioridade de exclusão dos arquivos no cache. Nos exemplos apresentados nessa seção, o GDE utiliza os fatores de freqüência de acesso, tempo desde o último acesso e tamanho dos documentos para decidir quais documentos devem ser removidos do cache. Isso é feito através da escolha dinâmica de uma política a ser aplicada aos documentos no cache, dentre as políticas LRU, SIZE e LFU. Dessa forma, nesse caso, as políticas LRU, SIZE e LFU são chamadas políticas básicas do Gerenciador Dinâmico de Espaço. A cada política básica é associado um peso. O peso da política indica qual a sua importância na escolha do documento a ser removido.

O GDE utiliza uma fila de documentos com várias ordenações. Cada ordenação é feita através de uma política básica. A figura 3.1 apresenta um exemplo onde a fila de documentos possui três ordenações.

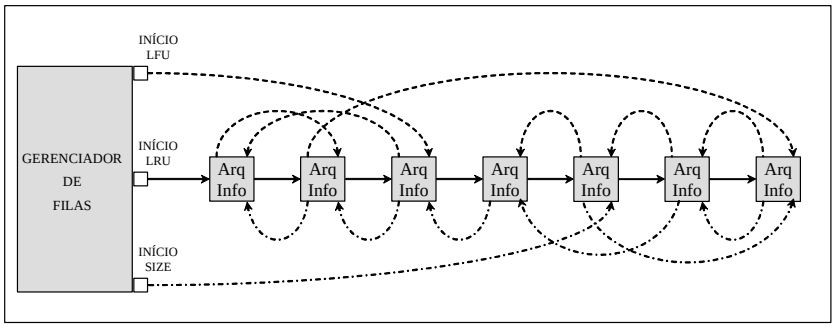

Figura 3.1. Exemplo de ordenação de documentos

Quando o GDE precisa ser aplicado ao conjunto de documentos armazenados no cache, cada política básica escolhe o documento considerado mais descartável. Serão escolhidos então 3 documentos, não necessariamente distintos. É verificada então a utilidade de cada documento escolhido para cada política básica. A utilidade depende da

posição relativa de cada documento em cada ordenação. Usando os pesos de cada política básica, o GDE então escolhe um dos documentos para ser removido. O documento escolhido é aquele cuja soma das utilidades dentre as políticas é a menor.

A tabela da figura 3.2 apresenta um exemplo do cálculo da utilidade dos documentos. Considere que o documento LRUChoice é o escolhido pela política LRU para ser removido. Da mesma forma, a política LFU escolheu o documento LFUChoice e a política SIZE escolheu SIZEChoice. Os valores LRUValue, LFUValue e SIZEValue são os valores de utilidade dos documentos escolhidos para cada política. Esses valores foram escolhidos aleatoriamente para esse exemplo.

	LRUChoice	**LFUChoice**	**SIZEChoice**
LRUValue	0.2	0.4	0.2
LFUValue	0.8	0.1	0.4
SIZEValue	0.1	0.3	0.1
Total	1.1	0.8	0.7

Figura 3.2. Exemplo de valores de utilidade

No exemplo da figura 3.2, considerando iguais os pesos das políticas básicas, o documento escolhido pela política SIZE, SIZEChoice, seria removido do cache.

A variação dos pesos de cada política básica permite melhorar o desempenho do cache na métrica mais importante para cada cache específico. O cache pode, por exemplo, ser programado para garantir um *Byte Hit Ratio* específico. Durante a operação do cache, se é detectada uma diminuição do *Byte Hit Ratio*, o GDE aumenta o peso da política LRU, que apresenta altas taxas de *Byte Hit Ratio*. Por outro lado, se o cache for programado para apresentar taxas de *Hit Ratio* maiores que um certo limite, ao detectar a queda de *Hit Ratio*, o GDE aumenta o peso da política SIZE, que apresenta melhores resultados em *Hit Ratio*.

4. Avaliação de Desempenho

Antes de explorar as características de adaptação dinâmica do GDE, é importante mostrar que essa política permite obter um desempenho, nas métricas tradicionais (*Hit Ratio* e *Byte Hit Ratio*), compatível com as melhores políticas existentes. Apesar do alto desempenho não ser o principal objetivo do GDE, resultados de avaliação de desempenho são apresentados nesta seção de forma a mostrar que além das funcionalidades de diferenciação da QoS oferecida, a utilização do GDE não compromete o desempenho do cache.

A avaliação do desempenho do GDE foi feita utilizando simulação baseada em *traces*. Os *traces* utilizados são provenientes da DEC [DEC00] e NLANR[NLANR00]. Da NLANR foram utilizados os *traces* UC e RTP. Todos os *traces* foram pré-processados para retirar as requisições a documentos não “cacheáveis” tais como páginas dinâmicas, que representavam menos de 5% das requisições. Na análise foram utilizados *traces* de uma semana da DEC e duas semanas da NLANR.

O trace DEC contém cerca de 700 mil requisições, e o número total de bytes requisitados foi de cerca de 8 GB. O trace RTP contém 3 milhões de requisições, a arquivos cuja soma é aproximadamente 18 GB. O trace UC é formado por 1 milhão de requisições a aproximadamente 6 GB.

Os testes de desempenho foram realizados comparando o GDE com as políticas GDS, GD*, LRU e LRU-DA. As políticas GDS e GD* foram escolhidas devido ao seu alto desempenho. A política LRU foi escolhida por ser largamente conhecida enquanto que a LFU-DA é um exemplo de política que privilegia arquivos com grande freqüência de acesso.

Os resultados do desempenho são apresentados nas figuras 4.1a, 4.1b e 4.1c. Os valores no eixo X representam os tamanhos dos caches analisados, em GigaBytes. Os pesos utilizados nas políticas básicas do GDE foram iguais. Os gráficos apresentam os resultados das políticas LRU, LFU-DA, GDS(1) e GD*(1) obtidos em [Jin00] usando os mesmos *traces* usados neste artigo. Nesse caso, as políticas GDS e GD consideram fixo o custo de cada documento.

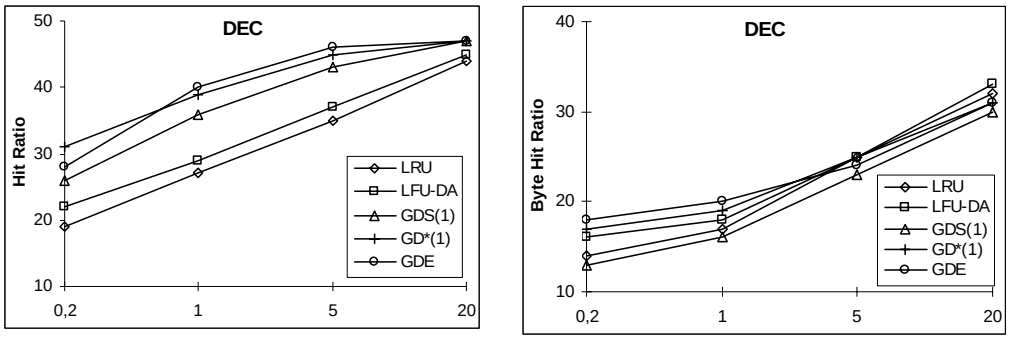

Figura 4.1a. Políticas LRU, LFU-DA, GDS(1), GD*(1) e GDE – Trace DEC

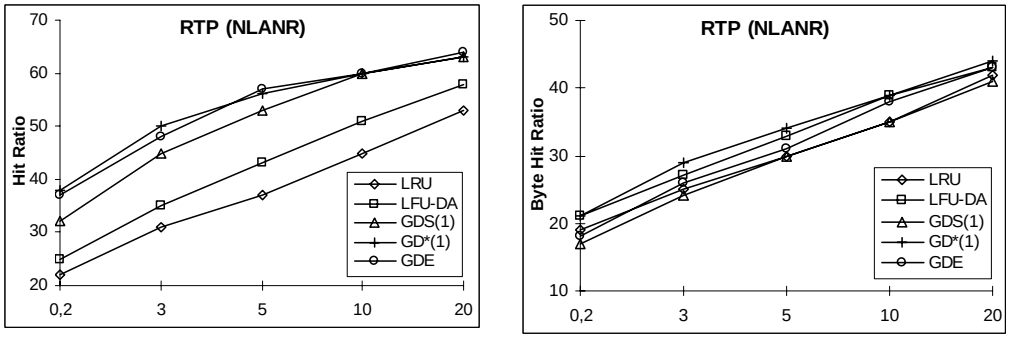

Figura 4.1b. Políticas LRU, LFU-DA, GDS(1), GD*(1) e GDE – Trace RTP (NLANR)

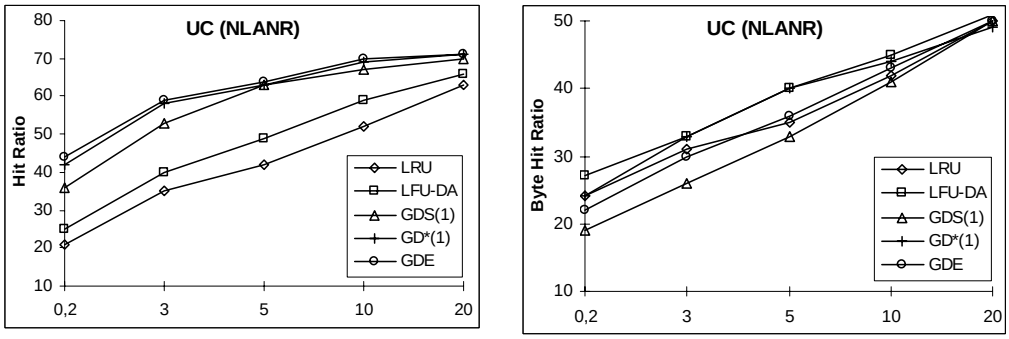

Figura 4.1c. Políticas LRU, LFU-DA, GDS(1), GD*(1) e GDE – Trace UC (NLANR)

Os resultados do desempenho do GDE comparado às políticas LRU, LFU-DA, GDS(packets) e GD*(packets) é apresentado nas figuras 4.2a, 4.2b e 4.2c. Nesse caso, as políticas GDS e GD consideram o custo de cada documento igual ao número de pacotes transferidos no envio do documento.

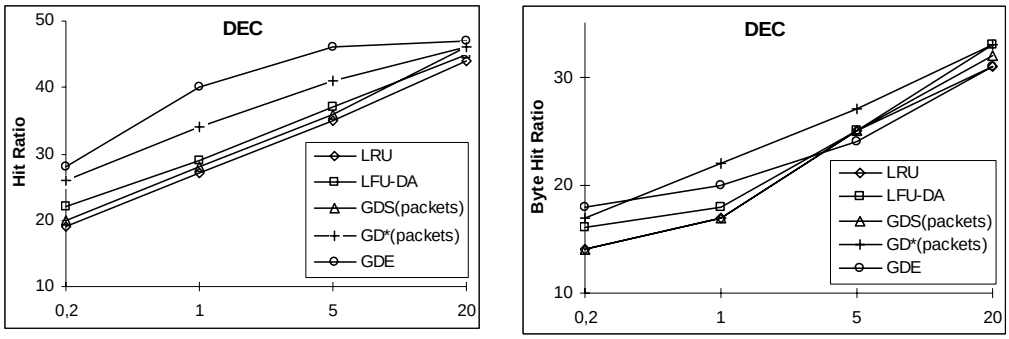

Figura 4.2a. Políticas LRU, LFU-DA, GDS(packets), GD*(packets) e GDE – Trace DEC

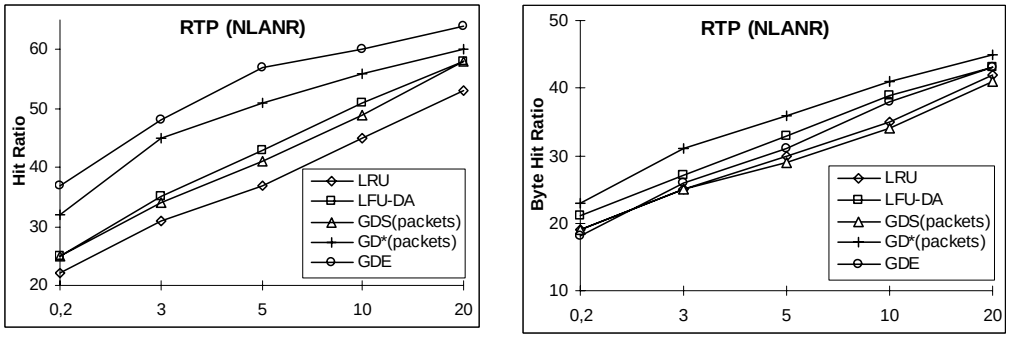

Figura 4.2b. Políticas LRU, LFU-DA, GDS(packets), GD*(packets) e GDE – Trace RTP

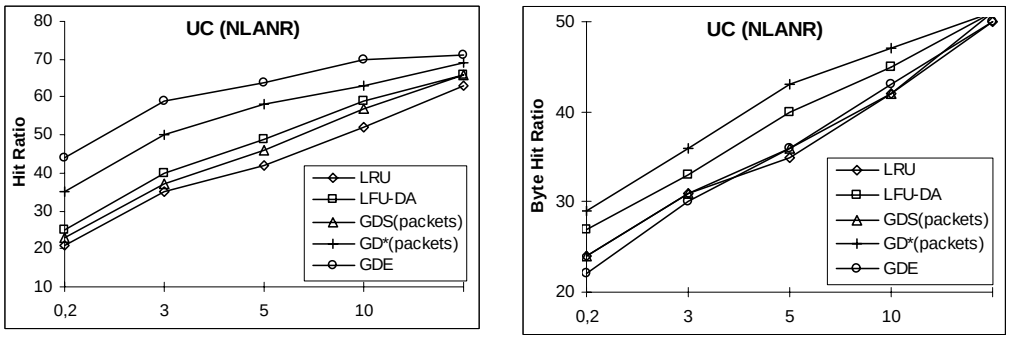

Figura 4.2c. Políticas LRU, LFU-DA, GDS(packets), GD*(packets) e GDE – Trace UC

Foram realizadas simulações para verificar o impacto da variação dos pesos das políticas básicas no desempenho em *Hit Ratio* e *Byte Hit Ratio*. Os resultados das simulações são apresentados na figuras 4.3a, 4.3b e 4.3c. Os valores nas legendas representam os pesos das políticas LRU, SIZE e LFU, respectivamente.

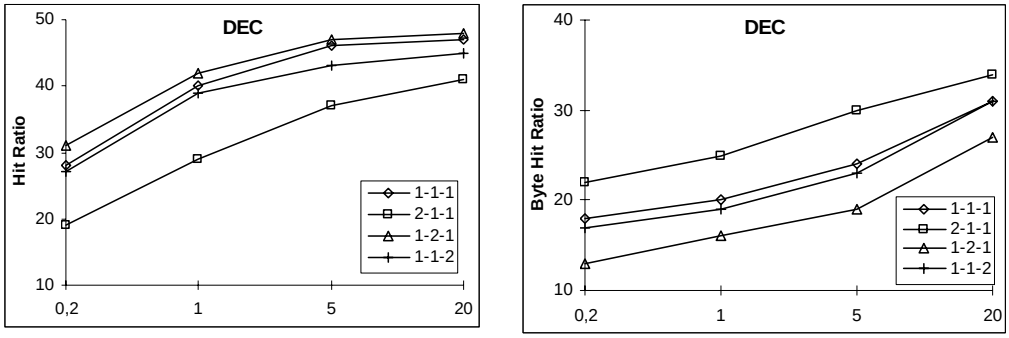

Figura 4.3a. Desempenho do GDE com variação de pesos – Trace DEC

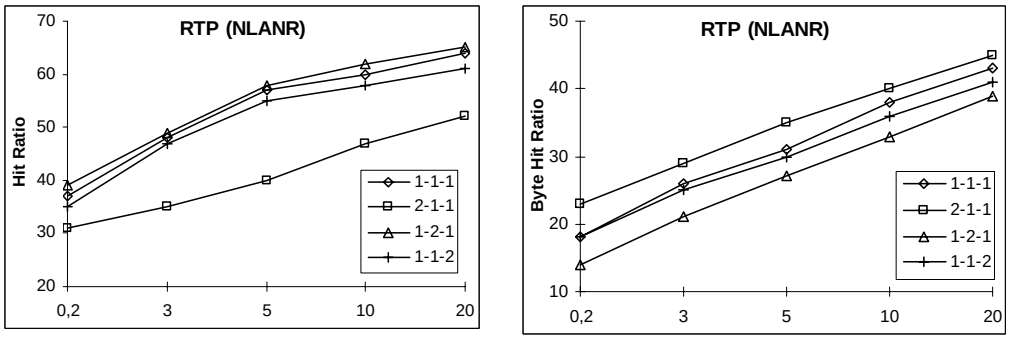

Figura 4.3b. Desempenho do GDE com variação de pesos – Trace RTP (NLANR)

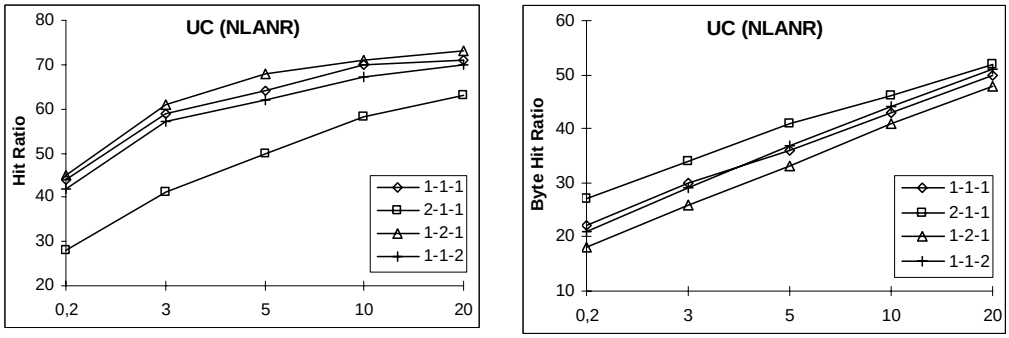

Figura 4.3c. Desempenho do GDE com variação de pesos – Trace UC (NLANR)

Analisando os resultados das figuras 4.1a, 4.1b e 4.1c, pode-se notar que o GDE apresenta desempenho compatível com as melhores políticas analisadas. As políticas LRU e LFU-DA apresentam os piores resultados em *Hit Ratio*. Isso ocorre porque essas políticas não levam em consideração os tamanhos dos documentos [Jin00].

As figuras 4.2a, 4.2b e 4.2c mostraram que o GDE obteve os melhores resultados em *Hit Ratio*. Em *Byte Hit Ratio*, obteve desempenho próximo dos melhores resultados obtidos pelas demais políticas.

A variação de desempenho em *Hit Ratio* e *Byte Hit Ratio* pela alteração dos pesos das políticas básicas o GDE mostrou que é possível otimizá-la de acordo com a métrica desejada. Assim, é possível utilizar apenas uma estratégia de gerenciamento de espaço em diferentes casos de utilização de caches para Web.

5. Aplicações do GDE

Nessa seção são apresentados exemplos de possíveis aplicações para o GDE. Essas aplicações tomam como princípio que são mantidas pelo GDE políticas capazes de favorecer os arquivos cujas QoS oferecidas deseja-se alterar. A alteração das características de favorecimento do cache aos arquivos é feita através da alteração dos pesos das políticas que interferem na probabilidade da manutenção desses arquivos no cache.

5.1. Favorecimento de arquivos em horas específicas

A popularidade dos arquivos requisitados varia com o tempo. Em alguns casos, arquivos têm sua popularidade aumentada muito rapidamente. Por exemplo, se um programa de televisão de grande audiência mencionar um Web site em sua programação, existe uma grande chance do número de acessos ao Web site mencionado aumentar muito. Muitas vezes, pode acontecer do aumento de popularidade daquele Web site ser tão grande a ponto de fazer com que o servidor responsável seja incapaz de responder a tantas requisições. Esse seria um caso em que o favorecimento do cache aos arquivos daquele Web site diminuiriam o número de requisições ao servidor.

Uma questão que pode surgir nesse ponto é quanto ao impacto real da utilização do GDE no favorecimento dos arquivos do Web site, pois se esses arquivos são tão populares a ponto de sobrecarregar o servidor, eles terão grande chance de serem mantidos em um cache que não utilize o GDE. Porém, no fluxo de requisições que passa pelo cache podem existir muitos arquivos muito mais populares que os arquivos do Web site que se deseja privilegiar.

A figura 5.1 apresenta um gráfico com os números de acessos aos arquivos do *trace* RTP, ordenados de acordo com sua popularidade. Quanto maior o número de acessos, mais popular é o arquivo e mais à esquerda esse arquivo é representado no gráfico. Os valores entre parênteses no eixo X representam a soma total dos tamanhos dos arquivos.

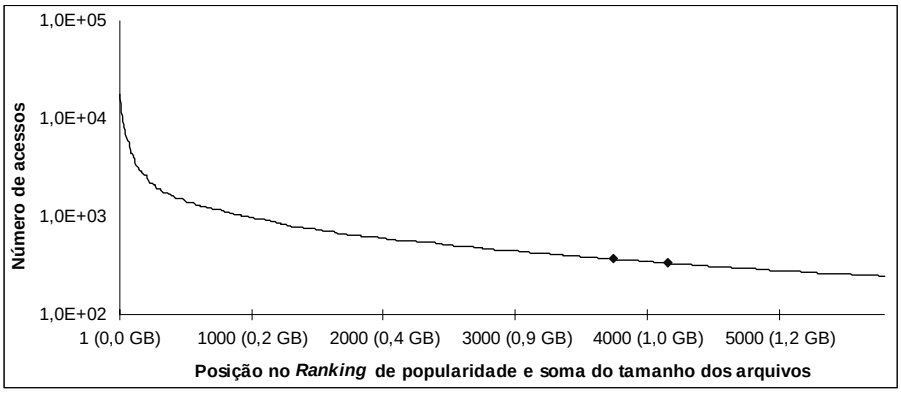

Figura 5.1. Distribuição de popularidade de arquivos

Um cache que não utilize o GDE terá uma grande probabilidade de favorecer apenas a manutenção dos arquivos mais populares. Por exemplo, na distribuição da figura 5.1, uma cache de 0,4 GB terá uma grande probabilidade de manter no cache os 2000 artigos mais populares. No caso de manter no cache arquivos com valores de popularidade entre os dois pontos destacados no gráfico, de forma a melhorar a QoS

oferecida ao servidor desses arquivos, é necessário usar uma política que privilegie esses arquivos. Porém, essa política não pode ser usada em todo o tempo de operação do cache, pois prejudicaria outros arquivos em momentos que não é necessário favorecer os arquivos específicos. Nesse caso, a solução usando o GDE seria manter uma política que favorecesse os arquivos mais populares e outra que permitisse favorecer arquivos específicos. Na maioria do tempo, a primeira teria um grande peso na decisão de quais arquivos remover. Para privilegiar arquivos específicos, a segunda política deveria ter seu peso aumentado.

Dessa forma, é possível usar o GDE para garantir uma alta QoS a arquivos específicos, mesmo que eles não estejam dentre os mais populares.

Para demonstrar a utilização do GDE no favorecimento de arquivos específicos, foi realizada uma simulação usando um cache de 100 MB. O GDE usou duas políticas: LRU e LRU com favorecimento, criada para essa experiência. A política LRU com favorecimento multiplica pelo grau de favorecimento a importância dos arquivos favorecidos em serem mantidos no cache.

Foram escolhidos aleatoriamente 1000 arquivos do *trace* RTP, que foi aplicado ao cache. Esses arquivos representavam arquivos que deveriam ser favorecidos pela LRU com favorecimento.

O valor do peso da política LRU foi 1 em todas as simulações. Foram experimentados pesos para a política LRU com favorecimento entre 1 e 10. Os resultados do desempenho em *hit ratio* dos arquivos escolhidos é apresentado na figura 5.2.

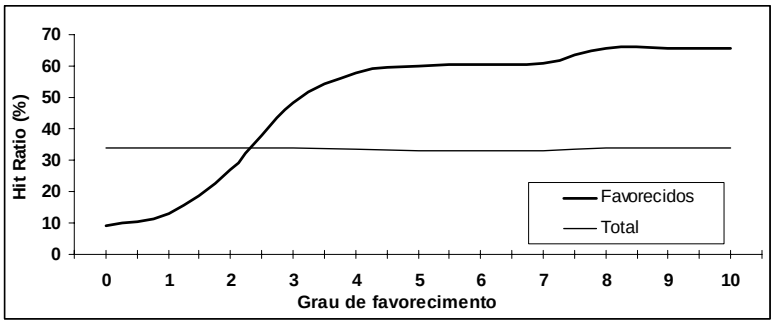

Figura 5.2. Impacto do favorecimento de arquivos

Através da figura 5.2, observa-se que a variação do peso da política LRU com favorecimento permitiu que as requisições aos arquivos escolhidos oferecessem uma QoS variável de acordo com o grau de favorecimento. O impacto do favorecimento no desempenho total do cache foi muito pequeno. Considera-se que isso ocorreu devido ao pequeno número de arquivos favorecidos.

5.2. Favorecimento de tipos de documentos

O GDE permite também favorecer tipos de arquivos. Por exemplo, pode-se instalar um cache cujo principal objetivo é diminuir o tempo de espera de usuários na busca por arquivos de música. Porém, o uso do GDE permite que o cache não seja exclusivo de arquivos de música, podendo também ser usado para outros tipos de arquivos quando não for necessário privilegiar especificamente os arquivos de música. Isso pode ser feito

variando-se os pesos de duas políticas básicas: uma que privilegie arquivos de música e outra que apresente um alto desempenho para arquivos em geral.

5.3. Favorecimento de usuários

Apesar de exigir uma troca de informações mais complexa, o cache pode usar uma política que favoreça arquivos requisitados por usuários específicos. Para tanto, é necessário que o cache mantenha uma tabela de prioridades de clientes, o que pode fazer com que o processo seja mais demorado. Além disso, é necessário que o cache mantenha dados que indiquem o cliente responsável pela requisição a cada arquivo armazenado, exigindo um maior consumo de memória e processamento.

O favorecimento de usuários permite que o cache concentre seus esforços em usuários especiais, mas sem deixar de permitir que usuários comuns utilizem-se da capacidade não utilizada pelos usuários favorecidos.

5.4. Variação dinâmica dos pesos das políticas básicas

A grande vantagem do GDE é permitir que seja possível utilizar tantas políticas de substituição quantas sejam necessárias, variando dinamicamente o peso de decisão de cada política para proporcionar a variação dinâmica da qualidade de serviço oferecida pelo cache a diferentes classes de clientes. Dessa forma, é possível programar o cache para favorecer determinados conjuntos de arquivos apenas em determinadas horas.

A variação automática dos pesos das políticas básicas do GDE pode também ser feita durante o decorrer do dia. A variação de desempenho em *Byte Hit Ratio* e *Hit Ratio* é importante devido às diferentes taxas de utilização da Web durante o dia. A análise do tráfego na Internet mostrou que existe um ciclo de 24 horas que apresenta tráfegos maiores em determinadas horas do dia [Morris00]. Considerando as diferenças de fusos horários pelo mundo, existirão momentos em que uma parte da Web mundial está sobrecarregada enquanto que sobra capacidade de transmissão em outras.

Por exemplo, quando são 4 da tarde na costa oeste dos EUA e os servidores estão sobrecarregados, será 1 da manhã em grande parte da Europa. Nesse momento, pode ser mais importante que os caches europeus economizem requisições aos servidores americanos do que a banda de transmissão dos *links* EUA-Europa. Esse seria então o momento de aumentar a taxa desejada de *Hit Ratio*. O GDE pode ser programada para variar os pesos das políticas básicas conforme a hora e conforme o desempenho do cache.

5.5. O GDE e o *push-caching*

Os caches para Web normalmente trabalham em um modelo onde as requisições dos clientes determinam o conteúdo do cache. No modelo alternativo chamado *push-caching*, os servidores replicam seu conteúdo no cache. Isso ajuda a prevenir que o servidor fique sobrecarregado de requisições e, desde que o servidor envie para o cache seus documentos mais populares, diminui o tempo de espera dos clientes e a carga na rede.

O GDE pode ser usado independentemente do *push-caching* pois a busca e armazenamento de arquivos podem ser programados para aceitar “recomendações” de servidores. Nesse caso, é necessário informar ao GDE que um novo arquivo foi incluído, sendo então necessário atribuir valores de utilidade futura para o arquivo em

todas as ordenações utilizadas, para que o GDE decida corretamente em que posição o arquivo se encontra em todas as ordenações.

6. Aspectos de implementação do GDE

Um dos grandes limites dos sistemas de gerenciamento de espaço em caches para Web é a quantidade de recursos computacionais exigidos para a sua execução. No caso do GDE, ela depende do número de políticas que podem ser usadas pelo cache. Supondo que as informações sobre os arquivos no cache sejam armazenados em uma lista duplamente ligada, para cada política é necessário que seja alocado espaço em memória para mais 2 ponteiros para cada registro de arquivo armazenado. Esses ponteiros são necessários na ligação do registro de cada arquivo armazenado à lista ordenada pela política. Dessa forma, é possível manter apenas um registro com informações sobre o arquivo e várias ordenações de prioridade de exclusão.

Entretanto, o maior problema é o consumo de ciclos do processador necessários para manter muitas diferentes ordenações para os arquivos no cache. Caso as informações dos arquivos fossem mantidas em uma lista duplamente ligada, a demanda por processamento causada pelas repetidas reordenações poderia tornar o custo do GDE muito alto.

Para diminuir o custo de execução, o GDE usa um sistema com listas de vetores de ponteiros para diminuir o número de ciclos de processador necessários para a manutenção das listas de arquivos. O esquema na figura 6.1 representa as estruturas necessárias para procura e remoção de arquivos em uma única política.

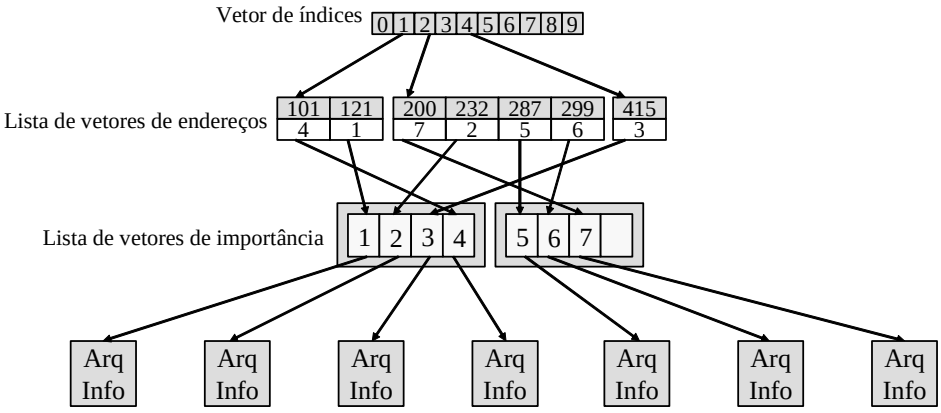

Figura 6.1. Estruturas necessárias para aumentar o desempenho do GDE

As listas de vetores de endereço e índice são utilizadas apenas para a localização rápida dos arquivos. Os vetores de índice são usados para endereçar rapidamente o início da lista dos arquivos cujos algarismos mais significativos do valor numérico gerado através da aplicação de uma função de *hash* no nome do arquivo são os mesmos. A lista de vetores de endereços é utilizada para armazenar a localização das informações sobre o arquivo nas listas de vetores de importância. O tamanho do vetor de índice depende do número de arquivos armazenados. Quanto maior o vetor, menor o número de posições do vetor de endereços que devem ser avaliadas. Por outro lado, o aumento exagerado do tamanho do vetor de índice causa um atraso maior na procura dentro deste vetor.

Para cada política básica do GDE, são mantidos o vetor de índices, a lista de vetores de endereços e a lista de vetores de importância. Uma representação de parte das estruturas exigidas pelo GDE quando da utilização de 3 políticas básicas é apresentada na figura 6.2.

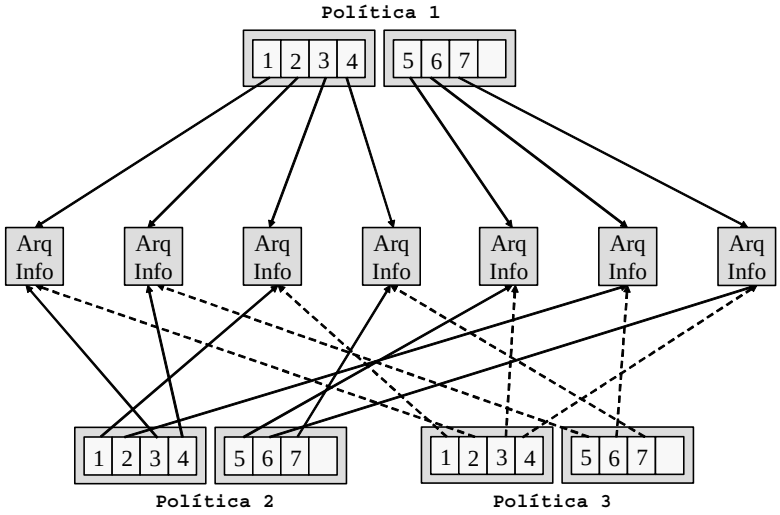

Figura 6.2. Listas de vetores de ponteiros

No exemplo da figura 6.2, é importante mencionar que, usando o sistema de listas de ponteiros, o crescimento da demanda por processamento é linear com o número de políticas básicas desejadas, pois é como se o cache tivesse que manter várias políticas independentes. Já no caso do crescimento do consumo de memória, o consumo é menor que o de várias políticas independentes, pois apenas um registro de informações sobre cada arquivo (Arq Info) precisa ser mantido no cache.

O sistema de listas de ponteiros foi utilizado em todos os experimentos apresentados nesse artigo. No restante dessa seção é sugerida outra estratégia para diminuir ainda mais a demanda por capacidade de processamento para o controle da prioridade de exclusão dos arquivos. Essa estratégia consistem em manter uma única fila de documentos, ordenada de acordo com um valor de importância de manutenção do arquivo, calculado utilizado os pesos das políticas básicas. Inicialmente, considera-se o caso em que se os pesos das políticas forem alterados, é necessário reordenar toda a lista de arquivos. A figura 6.3 apresenta um exemplo onde um cache usando um GDE altera os pesos das políticas básicas e, por isso, precisa reordenar toda a lista de arquivos.

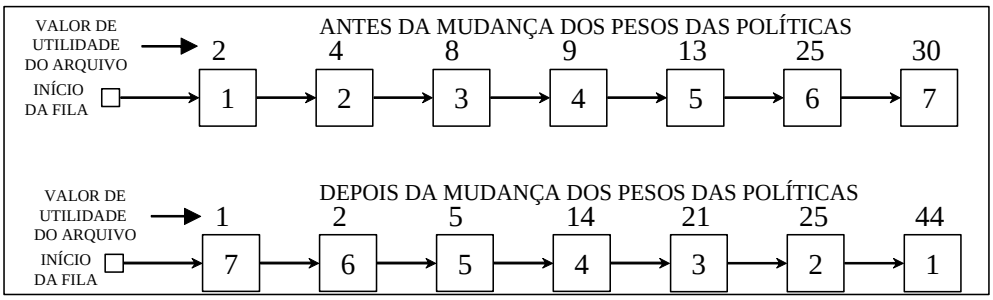

Figura 6.3. Antes e depois de uma reordenação da lista de arquivos

Entretanto, em muitos casos, a reordenação pode ser computacionalmente muito cara. Por isso, pode-se fazer com que o valor usado para ordenar a lista de arquivos não seja recalculado para todos os arquivos da lista. Nesse caso, apenas os arquivo inseridos a partir da mudança de pesos das políticas básicas terão seu valor de ordenação correto. Dessa forma, a orientação do cache para privilegiar determinados grupos de arquivos será feita gradativamente, a medida em que os arquivos antigos forem removidos.

7. Conclusões e trabalhos futuros

Em simulações de uso, o Gerenciador Dinâmico de Espaço (GDE) apresentou um desempenho comparável ao desempenho das melhores políticas de substituição existentes. Além disso, sua capacidade de ser configurada através da variação dos pesos das políticas básicas permite obter do cache o desempenho desejado, na métrica escolhida. Isso permite que o GDE seja uma excelente opção para possibilitar que o cache ofereça diferentes QoS tanto a servidores Web quanto a clientes específicos.

Foram ainda apresentadas estratégias que permitem diminuir a capacidade de processamento necessária para executar o GDE, possibilitando que a sua utilização seja feita mesmo em computadores que não possuam capacidade de processamento muito grande.

Como trabalhos futuros, pretende-se realizar novas experiências avaliar o impacto da variação de pesos das políticas básicas do GDE na QoS oferecida aos seus clientes. Essa QoS deve ser medida tanto em *Hit Ratio* quanto através do tempo de espera do usuário. Espera-se poder determinar uma relação entre os pesos atribuídos às políticas básicas e a diferenciação de QoS obtida. Para tanto, devem ser definidas formas de medir a QoS oferecida aos usuários.

Referências Bibliográficas

[Abrams95] M. Abrams, C. R. Standridge, G. Abdulla, S. Willians, E. A. Fox: “Caching proxies: Limitations and potentials”. *Proceedings of the 4th International World-wide Web Conference*, 119-133, 12/1995.

[Afonso98] M. Afonso, A. Santos, V. Freitas: “QoS in Web caching”. *Computer Networks And ISDN Systems*, 30(22-23):2093-2102, 11/1998.

[Almeida96] 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

[Almeida98] J. Almeida, M. Dabu, A. Manikutty, P. Cao. “Providing Differentiated Levels of Service in Web Hosting Services”. *Proceeding of the Workshop on Internet Server Performance*, 06/1998.

[Breslau99] 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.

[Cao97] P. Cao, S. Irani: “Cost-Aware WWW Proxy Caching Algorithms”. *Proceedings of the USENIX Symposium on Internet Technoloy and Systems*, 193-206, 12/1997.

[Cao99] P. Cao, J. Zhang, K. Beach. Active Cache: “Caching dynamic contents on the Web”. *Distributed Systems Engineering*, 6(1):43--50, 1999.

[Chan99] Y. Chan, J. Womer, S. Jamin, J. K. MacKie-Mason: “The Case for Market-based Push Caching”. *Proceedings of the Second International*

Conference on Telecommunications and Electronic Commerce, Nashville, TN, 11/1999.

[Crovella95] 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.

[Cunha95] 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.

[DEC00] Digital Equipament Coorporation ftp://ftp.digital/com/pub/DEC/traces/proxy - 2000

[Dingle96] A. Dingle: "Web cache coherence". *Computer Networks and ISDN systems*, vol. 28, No 7-11, 907-920, 1996.

[Jin00] 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.

[Kelly99] 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.

[Kelly99a] T. Kelly, S. Jamin, J.K. MacKie-Mason, "Variable QoS from shared Web caches: user-centered design and value-sensitive replacement". *Proceedings of the MIT Workshop on Internet Service Quality Economics (ISQE 99)*, Cambridge, MA, 12/1999

[Kim98] H. Kim, K. Chon: "Update policies for network caches". *Technical Memo, Department of Computer Science*, Korea Advanced Institute of Science and Technology, 07/1998.

[Malpani95] R. Malpani, J. Lorch, D. Berger: "Making World Wide Web caching servers cooperate". *Proceedings of the 4th International World-wide Web Conference*, 107-117, 12/1995.

[Morris00] R. Morris, D. Lin: "Variance of Aggregated Web Traffic". *IEEE INFOCOM 2000*, Tel Aviv, 360-366, 03/2000

[Murta98] C. D. Murta, V. A. F. Almeida, W. Meira Jr.: "Analyzing performance of partitioned caches for the WWW". *Proceedings of the Third International WWW Caching Workshop*, Manchester, England, 06/1998.

[NLANR00] National Laboratory For Applied Network Research - Examples of Squid Log Files. URL: ftp:// ftp.ircache.net/Traces/, 01/2000.

[Willians96] S. Williams, 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.