Uso de Técnicas e Informações para Potencializar Políticas de Substituição em Sistemas de Memória Virtual

Ricardo L. Piantola, Edson T. Midorikawa

Departamento de Engenharia de Computação e Sistemas Digitais
Escola Politécnica da Universidade de São Paulo
05508-900 – São Paulo – SP – Brasil

piantola@uol.com.br, edson.midorikawa@poli.usp.br

Abstract. *The virtual memory system performance depends directly on the quality of the memory management policy. Basically two strategies can be developed to improve such performance: the first one is creating new memory management policies that present, at the same time, simplicity and good performance; the second one is developing techniques and include information that will aid the policies that already exist. This paper aims to show a strategy that will aid replacement policies in order to obtain a good performance in a memory management system without changing the replacement policy behavior. To do so, a page prefetching technique along with information about access frequency, obtained through a method used in a statistical natural language processing, was used. The results show, besides the good performance, that the same strategy can be adopted in other algorithms.*

Resumo. *O desempenho do sistema de memória virtual depende diretamente da qualidade da política de gerência de memória. Basicamente duas estratégias podem ser desenvolvidas para melhorar tal desempenho: a primeira é criar novas políticas de gerência de memória que tenham, ao mesmo tempo, bom desempenho e simplicidade; a segunda é desenvolver técnicas e incluir informações para auxiliar as políticas já existentes. Este artigo procura mostrar uma estratégia para auxiliar políticas de substituição com a finalidade de obter bom desempenho em um sistema de gerência de memória, sem a necessidade de alterar o comportamento da política de substituição. Para isso, foi utilizada a técnica de busca antecipada de páginas em conjunto com a informação de frequência de acessos, obtida por meio de um método usado em processamento estatístico de linguagem natural. Os resultados mostram, além do bom desempenho, que a mesma estratégia pode ser adotada em outros algoritmos.*

1. Introdução

A busca por novas alternativas para a obtenção de um gerenciamento eficiente de recursos é alvo de pesquisas em várias partes do mundo. O advento de processadores *multi-core* levou aos computadores pessoais a questão do gerenciamento simultâneo de múltiplos processadores, anteriormente tratada apenas em sistemas operacionais para multiprocessadores das décadas de 70 e 80. A difusão dos monitores de máquinas virtuais fez com que a tecnologia de virtualização, desenvolvida em *mainframes* na

década de 60, tivesse aplicação em servidores de *data centers* ou em *desktops* de classes em cursos de graduação. Dessa forma, sistemas operacionais modernos devem adotar políticas mais eficientes para a gerência de seus recursos, dentre os quais a memória figura como um dos mais escassos.

Diversas estratégias avançadas de gerência de memória têm sido desenvolvidas na última década. Entre elas, destaca-se a dos algoritmos adaptativos de gerência de memória. Cada algoritmo desenvolvido busca aplicar alguma estratégia ou técnica e utiliza certo tipo de informação sobre o padrão de acessos à memória para buscar maior desempenho. Por exemplo, o algoritmo EELRU [Smaragdakis et al. 1999] faz uso de informações de recência nos acessos à memória. No caso do algoritmo LRU-WARlock [Piantola and Midorikawa 2008], parte da memória é reservada para páginas com alta frequência de acesso. Tais páginas são identificadas com a aplicação da técnica de *profiling*. Todos estes algoritmos adotam a estratégia de paginação sob demanda.

O objetivo principal deste artigo é apresentar uma forma de obter maior desempenho para uma política de gerência de memória por meio da adoção de técnicas complementares e da inclusão de informações, sem precisar modificar a política e seu comportamento original. Isto levou à criação de um novo algoritmo chamado LRU+ng.

Este artigo está organizado da seguinte forma: a seção 2 discute o uso do modelo n-grama, descrevendo sua importância em processamento estatístico de linguagem natural e sua adequação em sistemas de gerência de memória; a seção 3 descreve a técnica de busca antecipada e o novo algoritmo proposto. A avaliação de desempenho é realizada sobre o algoritmo LRU+ng e é apresentada na seção 4. Por fim, a seção 5 finaliza o artigo e traz as principais conclusões, indicando alguns trabalhos futuros.

2. O Uso de modelos n-gramas

Os n-gramas são tipos de modelos probabilísticos para prever o próximo elemento em uma sequência. Eles são usados em várias aplicações de processamento estatístico de linguagem natural e em Biologia, especificamente, em Genética [Manning and Schütze 1999]. Dado uma sequência, um n-grama é uma sub-sequência de n elementos. Os elementos podem ser letras, palavras, fonemas, nucleotídeos ou, como ocorre neste trabalho, páginas do sistema de memória virtual. Um n-grama de tamanho 1 é chamado de unigrama, de tamanho 2 é conhecido como bigrama, tamanho 3 é trigrama e tamanho 4 ou mais é nomeado n-grama.

A base matemática dos n-gramas foi proposta por Markov em 1913. O problema de como prever o próximo elemento foi muito estudado e é conhecido como *Shannon Game* [Shannon 1955]. O jogo é simples: dada uma sequência de letras, qual seria provavelmente a próxima letra? Exemplo: “*prova de geo...*”. De uma coleção de dados para treinamento de um programa que dará a resposta, o próprio programa pode retirar a distribuição probabilística para a próxima letra onde as probabilidades de todas as letras somadas seja 1,0. (exemplo: a=0,0001, b=0,0002, ..., g=0,8, ...).

A Tabela 1 mostra os bigramas mais comuns encontrados no jornal *The New York Times* de agosto a novembro de 1990 [Manning and Schütze 1999]. É possível, por meio deste *corpus* obtido com o jornal, aplicar os bigramas a um programa de reconhecimento de escrita para melhorar sua precisão. Um modelo de trigrama foi usado no sistema de reconhecimento de voz chamado IBM TANGORA já na década de 70 [Jurafsky and Martin 2008].

Este trabalho pretende utilizar o modelo de bigrama para trazer à memória principal, além da página responsável pela falta, a possível próxima página a ser referenciada.

Tabela 1. Bigramas mais comuns no *New York Times*.

Frequência	Palavra 1	Palavra 2
80871	of	the
58841	in	the
26430	to	the
21842	on	the
21839	for	the
18568	and	the
16121	that	the
15630	at	the

3. Proposta de auxilio a política LRU

O objetivo principal da proposta aqui apresentada é criar uma estratégia para auxiliar políticas de substituição de páginas com a finalidade de obter bom desempenho em um sistema de gerência de memória virtual, sem a necessidade de alterar o seu comportamento. Para conquistar este objetivo, foi desenvolvido o algoritmo LRU+ng, que auxilia a política de substituição LRU (*Least Recently Used*), sem modificá-la, somente utilizando informações de frequência de acessos e adotando a técnica de busca antecipada de páginas (*prefetching*).

O princípio do funcionamento do LRU+ng surgiu da ideia da arquitetura funcional do algoritmo LRU-WARlock que, sem mudar a política LRU-WAR e utilizando a tecnologia de *profiling*, explora a informação de frequência de acessos as páginas. Desta forma, o algoritmo base LRU-WAR consegue detectar padrões mais facilmente, melhorando o desempenho da gerência da memória [Midorikawa, Piantola and Cassettari 2008]. A questão crucial foi descobrir o que mais atrapalhava a detecção de referências sequenciais. A política LRU foi escolhida por ter um desempenho médio para a maioria de padrões de acesso exibidos pelas aplicações.

O mecanismo utilizado pelo LRU+ng é a busca antecipada de páginas [Silberschatz et al. 2005]. A busca antecipada trabalha de maneira diferente da paginação por demanda, onde somente a página responsável pela falta é trazida para a memória. No caso da busca antecipada de páginas, não somente a página que sofreu a

falta pode ser movida para a memória, mas outras quaisquer de acordo com a estratégia escolhida, tentando prever quais poderão ser referenciadas em um futuro próximo. No LRU+ng, as páginas são selecionadas de acordo com o modelo de n-gramas que é utilizado para simultaneamente separar frequencia e tentar prever qual página será acessada imediatamente após a atual.

Com relação à determinação das páginas selecionadas para a busca antecipada, foi escolhido o modelo de bigramas. Em nosso trabalho, queríamos trazer ferramentas de outras áreas do conhecimento, e testar sua aplicabilidade na gerência de memória virtual. Esse modelo possibilita capturar informações de frequência de acesso adicionado à probabilidade de previsão da próxima página a ser acessada. A Google, por exemplo, utiliza modelos de n-gramas em vários de seus projetos, tais como reconhecimento de voz, mineração de dados, tradução de texto, e verificação de pronúncia [Google 2008].

3.1. Algoritmo LRU+ng

O algoritmo LRU+ng adiciona uma técnica e uma informação à política LRU: a técnica de busca antecipada de páginas e a informação de frequência de acesso. Mesmo com esses mecanismos de auxílio, a política de substituição de páginas não foi modificada, ou seja, a vítima continua sendo a página menos recentemente acessada na memória.

A estrutura de dados do algoritmo mantém uma fila em ordem de acesso. No fim da fila, encontra-se a última página acessada, isto é, a mais recente. No outro extremo, está a menos recentemente acessada, em outras palavras, a vítima da substituição.

A operação do LRU+ng é a mesma do algoritmo LRU, pois quando ocorrer uma falta de página, a página referenciada é movida para a memória e entra na fila na posição MRU. Já a página na posição LRU é descartada da memória. Essa é a descrição de um sistema de paginação por demanda. No caso em que se adota a busca antecipada no momento da falta de página, tem-se, então, que as páginas representadas pelas letras “y” e “z” serão as vítimas da substituição. A página “a” será aquela da busca antecipada e a página “b”, a responsável pela falta (Figura 1).

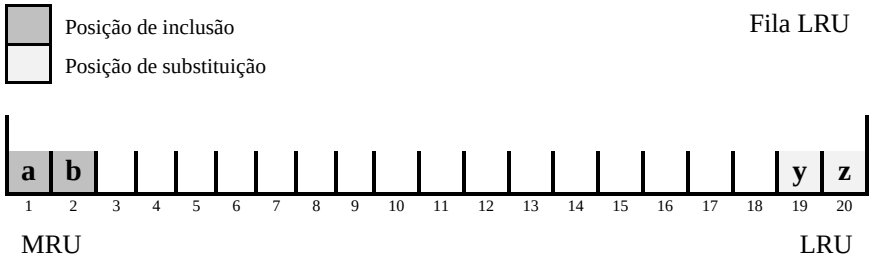


Figura 1. Estrutura do algoritmo LRU+ng.

A busca antecipada de páginas pode acontecer ou não quando ocorre uma falta de página, dependendo da configuração atual do sistema de memória. Se a página escolhida para a busca antecipada já estiver na memória, não haverá busca antecipada e o algoritmo funcionará como o LRU. A escolha da página da busca antecipada é feita através do método de bigramas. Para cada duas referências consecutivas é guardada a

frequência com que essas duas páginas são referenciadas novamente em sequência. A página que aparece com maior frequência após a referenciada é a escolhida para a busca antecipada. Por exemplo, se a sequência de acesso para a página 725 é a 730 e isto já ocorreu 20 vezes, e nenhuma outra página a supera, então a página 730 é a candidata da busca antecipada quando ocorre uma falta para a página 725. A estrutura escolhida para guardar esta relação foi um *hash*, pela velocidade da busca nesta estrutura, onde a chave é a página responsável pela falta e o valor é a página da busca antecipada.

A Figura 2 apresenta o pseudocódigo do algoritmo, onde são destacados os acréscimos ao código original do LRU. As novas condições adicionadas ao código são referenciadas nas linhas 6 até 10 e 12. A linha 6 verifica se a página para a busca antecipada encontra-se na memória. Em caso negativo, as linhas seguintes tratam de trazê-la para a memória e remover a página na fila LRU (caso não tenha mais memória disponível); no pseudocódigo, a linha 12 cria, exclui e atualiza a estrutura a cada referência para a busca antecipada.

```
1. Se a página referenciada está na memória:
2.     Reordena a página na primeira posição da fila.
3. Senão, se não está na memória:
4.     Se a memória está cheia:
5.           Remove a página na posição LRU da fila.
6.     Se página do hash para busca antecipada existe e não está na memória:
7.           Se a memória está cheia:
8.                 Remove a página na posição LRU da fila.
9.           Carrega a página referenciada e a página da busca antecipada.
10.    Senão, se a página da busca antecipada está na memória:
11.          Carrega a página referenciada no início da fila.
12. Manutenção do hash para busca antecipada.
```

Figura 2. Pseudocódigo do algoritmo LRU+ng.

É possível remover a função de manutenção do *hash* na linha 12 (Figura 2). Para isso é necessário incluir a técnica de *profiling* [Midorikawa, Piantola and Cassettari 2008], deixando assim o código mais rápido e com maior precisão desde o início de sua execução.

4. Análise do Desempenho

Nesta seção apresentamos a análise de desempenho efetuada sobre o algoritmo de substituição de páginas LRU+ng. Começamos com a descrição dos *traces* de aplicações e ferramentas utilizadas para a avaliação e, na seqüência, a análise dos resultados obtidos através dos testes.

4.1. Caracterização dos *Traces* Utilizados

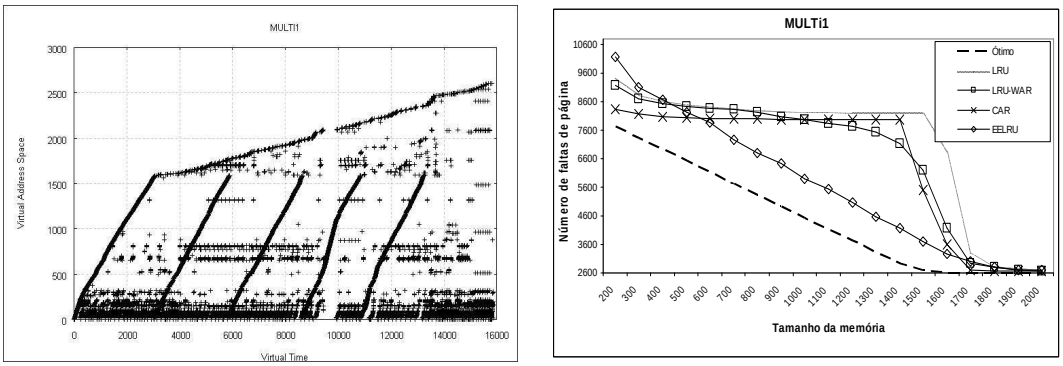
A avaliação do LRU+ng foi efetuada usando três diferentes *traces* de aplicações, são eles: multi1, multi2 e multi3 (Tabela 2). Os três *traces* [Kim et al 2000] foram selecionados pelo fato de conterem acessos simultâneos de aplicações, simulando um ambiente multiprogramado. As ferramentas utilizadas nas simulações fazem parte do ambiente Elephantools [Cassettari and Midorikawa 2004a].

Apresentamos a composição dos traces utilizados nos testes (Tabela 2), além de uma breve descrição dos padrões de acesso e seu comportamento com alguns algoritmos de substituição apresentados na literatura como o LRU e o LRU-WAR, entre outros.

Tabela 2. Descrição dos *traces* utilizados.

Trace	Descrição	Origem	Páginas
multi1	Execução simultânea das aplicações cscope e cpp.	LIRS	2606
multi2	Execução simultânea das aplicações cscope, cpp e postgres.	LIRS	5684
multi3	Execução simultânea das aplicações cpp, gnuplot, glimpse e postgres.	LIRS	7454

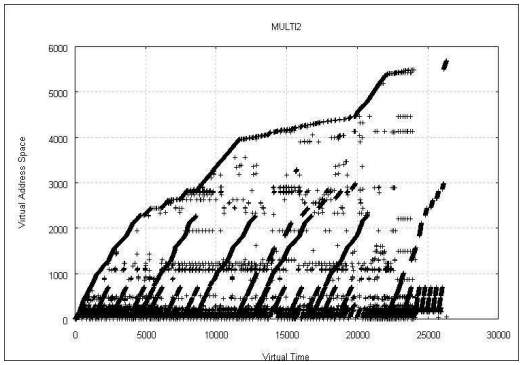
O *trace* **multi1** é composto pelas aplicações cscope e cpp. O cscope é uma ferramenta interativa de verificação de programa fonte escrita em linguagem C. Seu padrão de acesso à memória faz referências a *looping* com forte localidade temporal e outras referências de padrão diverso (Figura 3.a). O cpp é o pré-processador do GNU C, durante cuja execução podem ser observados blocos de referências sequenciais à memória em conjunto com outras referências. O *trace* multi1 intercala acessos dessas duas aplicações, uma com referências a *looping* e a outra com acesso sequencial. Esse padrão de acesso prejudica muito o desempenho do algoritmo LRU [Jiang and Zhang 2002] e isso acontece até que o espaço de memória disponível seja maior ou igual ao tamanho total das duas aplicações (Figura 3.b). Os programas individuais apresentam um padrão de acessos adequado ao LRU-WAR, porém como estão intercalados pela multiprogramação, diminuem consideravelmente o desempenho do algoritmo.



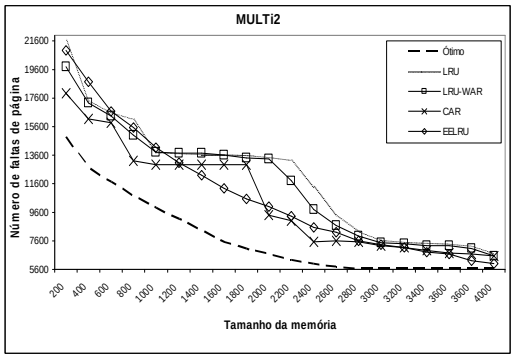
(a) Padrão de acessos à memória. (b) Desempenho de alguns algoritmos.

Figura 3. Características do *trace* multi1.

O *trace* **multi2** tem a mesma composição do *trace* **multi1** com a adição do programa postgres. O postgres é um sistema de banco de dados relacional da Universidade da Califórnia, que apresenta um padrão de acessos sequencial e *looping* com períodos não- constantes. A Figura 4.a mostra o padrão de acessos do *trace* multi2. A Figura 4.b mostra o desempenho de vários algoritmos e, de modo geral, a maioria dos algoritmos apresenta um desempenho muito próximo ao LRU.



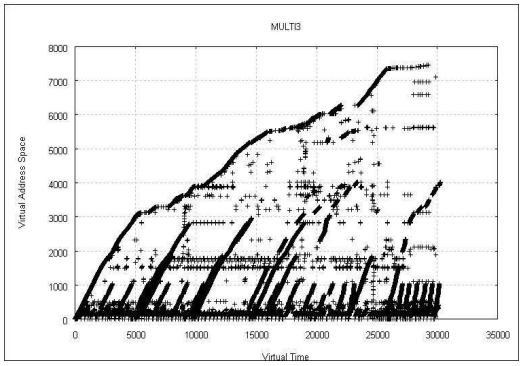
(a) Padrão de acessos à memória.



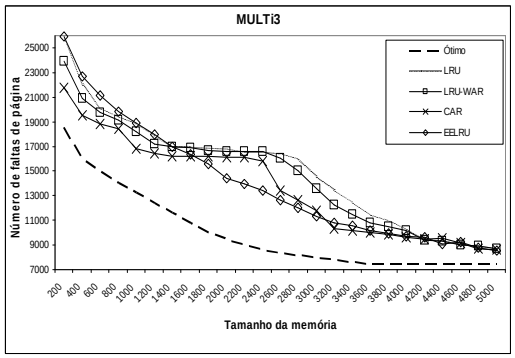
(b) Desempenho de alguns algoritmos.

Figura 4. Características do *trace* multi2.

O terceiro *trace* escolhido, o **multi3**, tem a configuração um pouco diferente dos dois primeiros. É formado pelo cpp, prostgres, glimpse e gnuplot. O cpp está contido nos dois primeiros *traces* e o prostgres no multi2. O glimpse é um utilitário usado na busca de informações em textos e seu padrão de referência à memória é bem diverso. Já o gnuplot tem um padrão de acessos sequenciais bem definido. O gnuplot é um programa interativo de plotagem gráfica. É possível observar (Figura 5.b) que com o aumento da quantidade de programas contidos no multi3, a diferença de desempenho entre o algoritmo LRU e LRU-WAR diminui.



(a) Padrão de acessos à memória.



(b) Desempenho de alguns algoritmos.

Figura 5. Características do *trace* multi3.

4.2. Resultados Obtidos e Análises

Nesta seção apresentamos, segundo os objetivos definidos neste trabalho, as análises e os resultados do estudo do desempenho do algoritmo LRU+ng, que utiliza o modelo de bigramas e com busca antecipada de páginas.

Multi1

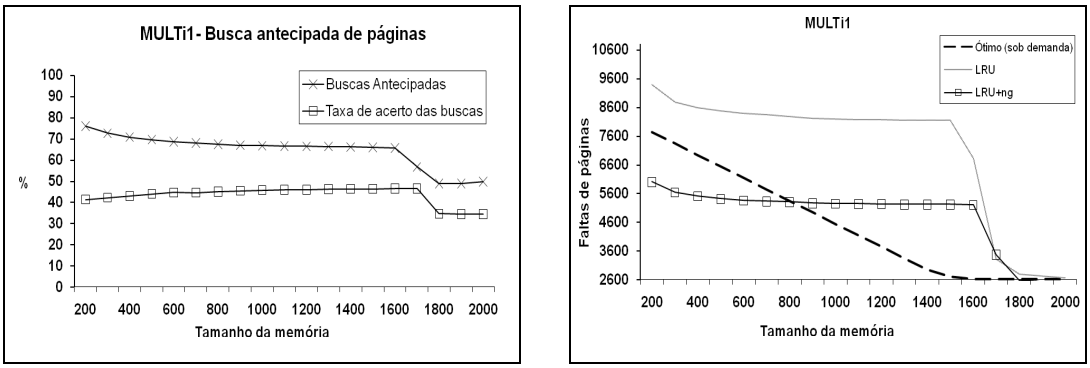
O algoritmo LRU tornou-se o mais amplamente utilizado como política de substituição de páginas porque ele consegue prever razoavelmente o comportamento geral dos programas, atingindo, assim, bom desempenho na maioria dos padrões de acesso à memória. Entretanto, em um programa com padrões de acessos sequenciais e que contém grandes *loops*, o algoritmo LRU não atinge um bom desempenho. Já o *trace* multi1 apresenta acessos de duas aplicações intercaladas em um sistema de gerência de

memória global que têm padrões de acessos sequenciais, fato que causa o mau desempenho. (Figura 6.b).

No multi1, a análise do perfil de frequência de acessos às páginas mostra que somente 7,5% das páginas são responsáveis por 50% dos acessos à memória [Piantola and Midorikawa 2008]. Podemos, então, prever que a informação de frequência de acessos, obtida pelo modelo dos n-gramas, irá proporcionar menos faltas de páginas. O LRU+ng supera o LRU em todos os tamanhos de memória, com ganho médio de quase 36% (Figura 6.b), o que confirma a nossa previsão.

O gráfico da Figura 6.a representa duas grandezas de eficiência sobre a busca antecipada no algoritmo testado. A primeira, com o tracejado em "x", representa em porcentagem de quantas vezes LRU+ng fez a busca antecipada sobre todos as faltas de páginas ocorridas em um tamanho de memória especificado pela abscissa. A segunda, com o tracejado em forma de quadrado, representa o total de acertos sobre a página buscada. Um acerto só é mensurado nas páginas que foram buscadas e imediatamente acessadas, ou seja, acessadas na próxima referência.

Uma marca de extrema importância sobre o algoritmo LRU+ng é que ele supera o algoritmo Ótimo em todos os tamanhos de memória menores a 800 páginas. O algoritmo Ótimo, também conhecido como OPT ou MIN, é o algoritmo teórico que apresenta o melhor desempenho possível entre os algoritmos de paginação sob demanda [Silberschatz et al. 2009]. No caso do LRU+ng, a razão para seu melhor desempenho em relação ao algoritmo Ótimo é o fato de que ele não é um algoritmo de paginação sob demanda, pois usa uma técnica de busca antecipada de páginas. Portanto, concluímos que, como nenhum algoritmo de paginação sob demanda pode ser melhor que o Ótimo, nenhum pode alcançar o desempenho obtido pelo LRU+ng para memórias de tamanho pequeno com o *trace* Multi1. Toda referência ao algoritmo Ótimo neste artigo se trata da substituição ótima sob demanda e não a busca antecipada ótima.



(a) Buscas sobre total de Faltas (b) Comparação entre LRU, LRU+ng e Ótimo

Figura 6. Gráficos do *trace* multi1.

Podemos observar também que, como preservamos a política LRU no algoritmo LRU+ng, o comportamento do novo algoritmo permanece o mesmo. No gráfico da figura 6.b as linhas representadas pelos algoritmos permanecem paralelas. É possível, assim, melhorar os algoritmos sem a necessidade de modificar a política vigente.

O que marca a grande eficiência do LRU+ng em memórias de pequeno tamanho é que quando temos pouca memória e um número grande de páginas do programa, a probabilidade das páginas acessadas estarem fora do memória principal é alta. Por esse motivo, do total de faltas de páginas, em 76% dos casos é utilizada a busca antecipada de página, ou seja, a página mais acessada após aquela que ocasionou a falta também é colocada na memória. Já em tamanhos grandes de memórias isso ocorre em apenas 49% das faltas.

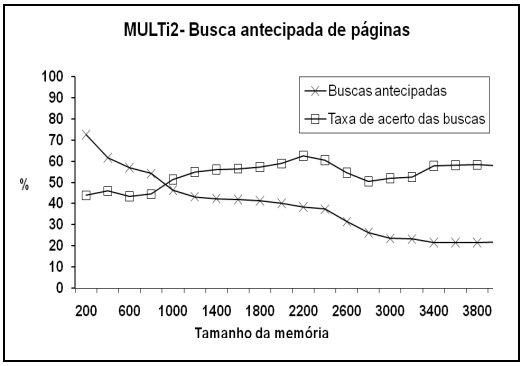
Multi2

A principal característica do *trace* multi2 para a análise surge do fato de ele ser semelhante ao multi1. O que difere os dois *traces* é a adição do programa postgres. O desempenho do LRU+ng também é muito bom, apesar de ser mais difícil distinguir padrões de acessos com a inserção de novas referências intercaladas (Figura 4.a).

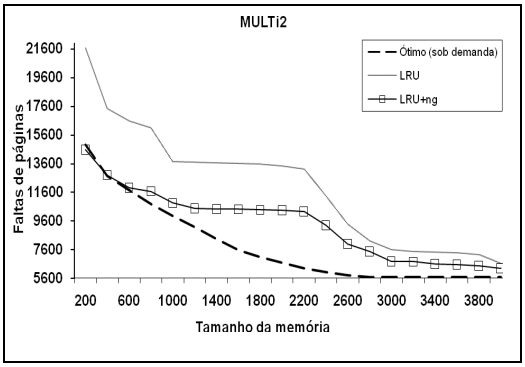
A análise do perfil de frequência de acessos às páginas mostra que metade das referências é feita por apenas 6% do total de páginas [Piantola and Midorikawa 2008]. Conforme analisado no *trace* multi1 a informação de frequência, obtida pelo modelo dos n-gramas, proporciona menos faltas de páginas também neste caso.

O LRU é superado pelo LRU+ng em todos os tamanhos de memória. O mesmo acontece com o algoritmo Ótimo, que é superado pelo LRU+ng em tamanhos de memórias pequenas. A eficiência do LRU+ng em memórias de pequeno tamanho no *trace* multi2, é determinado pelo fato do algoritmo não encontrar a página da busca antecipada na memória e trazê-la para a memória em quase 75% das vezes, contra 22% das vezes em memórias de tamanho grande. Esse comportamento pode trazer muitos benefícios a sistemas que utilizam controle mais rígido de memória para aplicações, ou sistemas de gerenciamento de memória que compartilhem um conjunto mínimo de páginas por processo. Por exemplo, sistemas que adotam uma política de gerenciamento de memória local.

A curva representada no gráfico do LRU+ng se alinha paralelamente com a curva do algoritmo LRU, aumentando ainda mais sua identidade com a política LRU preservada no LRU+ng.



(a) Buscas sobre total de Faltas



(b) Comparação entre LRU, LRU+ng e Ótimo

Figura 7. Gráficos do *trace* multi2.

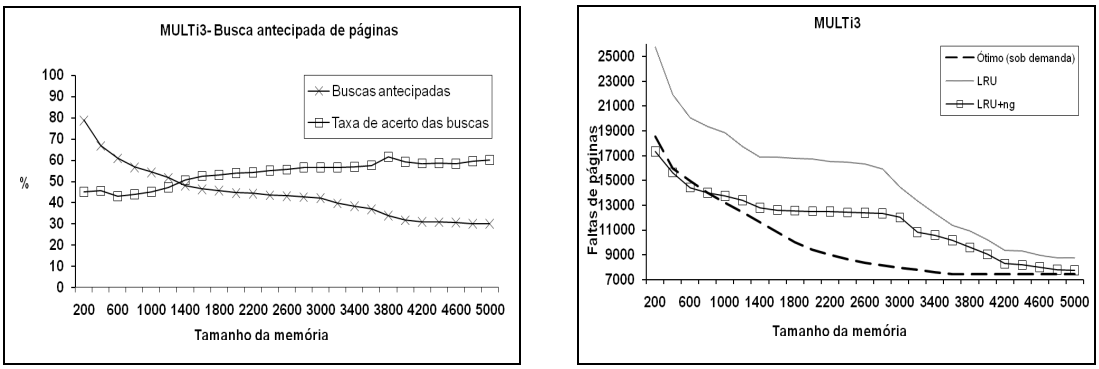
É possível notar que na Figura 7.a quanto maior é o tamanho da memória, menos buscas antecipadas são feitas. Entretanto, a precisão aumenta (Figura 7.b). Isso ocorre porque a maioria das páginas já está na memória. Então, nas poucas vezes em que é executada a busca antecipada, ela ocorre em uma mudança de *working set*. Estes resultados mostram que o LRU+ng é bem adequado para sistemas com política de memória global.

Multi3

Dos *traces* utilizados na análise, o que contém o maior nível de multiprogramação é o *trace* multi3. São diversos padrões de acessos intercalados entre os quatro programas. Como não existe uma uniformidade da localidade temporal, páginas que serão acessadas em um futuro próximo são descartadas prematuramente. Para esta seqüência de referências o LRU não é bom, porém o LRU+ng recupera essas páginas que foram descartadas prematuramente e supera o desempenho do LRU (Figura 8.b).

Assim como os *traces* multi1 e multi2, o LRU+ng também supera o algoritmo Ótimo nas memórias de pequeno tamanho no *trace* multi3. Outro fator em comum é o padrão da curva de desempenho do LRU que foi mantida no LRU+ng. Os ganhos obtidos chegam em média a 26% em comparação com o LRU.

Na Figura 8.a, quanto menor o tamanho da memória, mais buscas antecipadas são feitas, mas a precisão no acerto imediato da página buscada é baixo. No entanto, o desempenho é muito melhor, comparando com as memórias de tamanho médio e grande. O oposto acontece com as memórias de maior tamanho. Apesar da precisão no acerto das páginas das buscas antecipadas ser baixa, os pontos de maior desempenho são encontrados nas memórias de menor tamanho. Um hipótese a considerar é que as páginas da busca antecipada, apesar de não serem acessadas imediatamente após a busca, são acessadas pouco tempo depois, e assim, não entrando na contagem de taxa de acerto no gráfico (Figura 8.a). Dois motivos enriquecem essa hipótese. O primeiro é que o *trace* intercala acessos de quatro aplicações, ou seja, uma página qualquer pode estar entre duas páginas da mesma aplicação que gerealmente são acessadas consecutivamente. O segundo motivo é que a página da busca antecipada tem grande possibilidade de pertencer ao *working set* atual.



(a) Buscas sobre total de Faltas (b) Comparação entre LRU, LRU+ng e Ótimo

Figura 8. Gráficos do *trace* multi3.

Como nos outros dois *traces* estudados, o multi3 também atinge excelente desempenho quando se tem disponíveis quantidades mínimas de memória. Deste modo é uma estratégia que pode beneficiar sistemas dedicados com recursos restritos. Uma possibilidade de aumentar a taxa de busca antecipada e melhorar o desempenho é, além de utilizar os bigramas, adotar os trigramas.

5. Conclusões e Trabalhos Futuros

Apresentamos um estudo sobre estratégias para auxiliar políticas de substituição páginas em um sistema de gerenciamento de memória global. Este estudo foi conduzido sobre o algoritmo LRU modificado, chamado LRU+ng, que aplicou a técnica de busca antecipada de páginas junto com a informação de frequência de acessos, obtida por meio de um método usado em processamento estatístico de linguagem natural. Foram utilizados três *traces* com características de multiprogramação para os estudos de desempenho com políticas de gerência de memória global.

A contribuição mais importante foi mostrar que, apesar de vários estudos anteriores apontarem novas políticas sofisticadas para solucionar o problema da substituição de páginas ou a junção de várias políticas [Smaragdakis 2004], é possível, sem modificar a política vigente, melhorar seu desempenho com o auxílio de técnicas e informações. A política deixa de ser o objeto único dos esforços para a solução e novas técnicas complementares ganham valor. As análises revelaram que é imprescindível tratar o aspecto da frequência nos acessos à memória, quando o sistema utilizar uma política de gerência de memória global. A prova dessa tese está no fato observado em que, nos três *traces* estudados, menos de 9% do número total de páginas é responsável por 50% das referências à memória. Outro resultado interessante é que, nos três *traces* estudados, o algoritmo LRU+ng supera o algoritmo Ótimo em memórias de tamanho pequeno. Podemos tirar duas conclusões: a primeira é que todo algoritmo de substituição sob demanda é limitado ao Ótimo, ou seja, o Ótimo é o melhor algoritmo de substituição sob demanda, mas impossível de implementar. Algoritmos com busca antecipada podem ultrapassar esse limite imposto pelos sob demanda. A segunda conclusão gira em torno da boa atuação do algoritmo testado em memórias de pequeno tamanho. Isso faz com que o algoritmo proposto obtenha vantagens em sistemas de memória com gerência local, que distribuem poucas páginas para cada um dos processos.

Alguns estudos complementares podem ser desenvolvidos para dar continuidade a este trabalho. Uma sugestão a considerar é incluir, além dos bigramas, trigramas ou n-gramas, para que aumentem as buscas antecipadas em memórias de maior tamanho, diminuindo, assim, faltas de páginas posteriores. Uma segunda sugestão é repetir o estudo realizado neste artigo para políticas adaptativas, como o LRU-WAR, de forma que informações mais detalhadas e técnicas mais robustas sejam disponibilizadas para a política. Contudo, além da informação de frequência de acessos, outras informações poderiam ser disponibilizadas indiretamente à política no auxílio da gerência de

memória, como por exemplo, a composição do *working set*. Estudos sobre qual informação é mais relevante e outras técnicas de auxílio estão sendo conduzidos.

Referências

Cassettari, H. H. (2004). "*Análise da Localidade de Programas e Desenvolvimento de Algoritmos Adaptativos para Substituição de Páginas*." Dissertação de Mestrado. Escola Politécnica da Universidade de São Paulo, 2004.

Cassettari, H.H. and Midorikawa, E.T. (2004a) "*Caracterização de Cargas de Trabalho em Estudos sobre Gerência de Memória Virtual*", In Anais do III Workshop em Desempenho de Sistemas Computacionais e de Comunicação (WPerformance 2004), Salvador, BA.

Cassettari, H.H. and Midorikawa, E.T. (2004b) "*Algoritmo Adaptativo de Substituição de Páginas LRU-WAR: Exploração do Modelo LRU com Detecção de Acessos Seqüenciais*". In: Anais do I Workshop de Sistemas Operacionais (WSO 2004), Salvador, BA.

Google. (2009). "*Papers Written by Googlers"*. Disponível em http://research.google.com/pubs/papers.html. Acesso em 10/03/2009.

Jiang, S. and Zhang, X. (2002) "*LIRS: An Efficient Low Inter-Reference Recency Set Replacement Policy to Improve Buffer Cache Performance*", In Proc. of the ACM International Conference on Measurement and Modeling of Computer Systems (SIGMETRICS'02), Marina Del Rey, pp.31-42.

Jurafsky, D. and Martin, J. H. (2008) "*Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition*". Prentice Hall, New Jersey.

Kim, J.M. et al. "*A low-overhead high-performance unified buffer management scheme that exploit sequential and looping references",* In: Symposium on Operating System Design and Implementation, 4., San Diego, 2000. *OSDI' 2000*: Proceedings. San Diego: USENIX, 2000 pp.119-134.

Manning, C. D. and Schütze, H. (1999) "*Foundations of Statistical Natural Language Processing*". The MIT Press, Cambridge, MA.

Midorikawa, E.T. (1997) "*Uma nova estratégia para a gerência de memória para sistemas de computação de alto desempenho*". 193p. Tese (Doutorado) – Escola Politécnica, Universidade de São Paulo. São Paulo, 1997.

Midorikawa, E.T., Piantola, R.L., Cassettari, H.H. (2007) "*Influência dos Parâmetros de Controle no Desempenho de Algoritmos Adaptativos de Substituição de Páginas*". In: Anais do IV Workshop de Sistemas Operacionais (WSO 2007), Rio de Janeiro, RJ.

Midorikawa, E.T., Piantola, R.L., Cassettari, H.H. (2008) "*On adaptive replacement based on LRU with working area restriction algorithm*". ACM SIGOPS Operating Systems Review (OSR), vol.42, n.6, Oct 2008, New York, pp 81 – 92.

Piantola, R.L. and Midorikawa, E.T. (2008) "*Ajustando o LRU-WAR para uma Política de Gerência de Memória Global*". In: Anais do V Workshop de Sistemas Operacionais (WSO 2008), Belém, PA.

Silberschatz, A., Gagne, G. and Galvin, P.B. (2009) *Operating System Concepts*. 8th Edition, Wiley.

Smaragdakis, Y., Kaplan, S., and Wilson, P. (1999) "*EELRU: Simple and Effective Adaptive Page Replacement*", In Proc. of the ACM International Conference on Measurement and Modeling of Computer Systems (SIGMETRICS'99), Atlanta, pp.122-133.

Smaragdakis, Y. (2004) "*General Adaptive Replacement Policies*", In Proc. of the 2004 International

Symposium on Memory Management (ISMM'04), Vancouver, British Columbia, pp.108-119.