Influência das Técnicas e Informações no Auxílio de Políticas Adaptativas de Substituição de Páginas

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 including information that will aid the policies that already exist. This paper aims to show a strategy that will aid adaptive replacement policies without changing the replacement policy behavior. To do so, a page prefetching technique and information on the relationship between pages were used. The results show, besides the good performance, that the same strategy can be adopted in other adaptive 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 adaptativas de substituição, 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 relação entre páginas. Os resultados mostram, além do bom desempenho, que a mesma estratégia pode ser adotada em outros algoritmos adaptativos.*

1. Introdução

Com os atuais sistemas operacionais com novos recursos, a procura por alternativas para se obter um gerenciamento de memória eficaz e eficiente é alvo de pesquisas na sociedade científica. Para os desenvolvedores de sistemas operacionais, é um grande desafio melhorar as políticas de gerência de memória para obter bom desempenho, independentemente do uso do sistema computacional e sua configuração.

Várias 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 técnica e utiliza certo tipo de informação sobre o padrão de acessos à memória para buscar maior desempenho. Por exemplo, o algoritmo LIRS [Jiang and Zhang 2002] tem como objetivo minimizar as deficiências apresentadas pelo LRU utilizando um critério adicional interessante: a chamada IRR (Inter-Reference Recency),

que representa o número de páginas referenciadas entre os dois últimos acessos consecutivos a uma mesma página, e tenta, assim, prever o reuso das páginas. Já o algoritmo FPR [Sabeghil and Yaghmaee 2006] faz uso da lógica nebulosa para decidir qual página será substituída. 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 esses algoritmos adotam a estratégia de paginação sob demanda.

O objetivo principal deste trabalho é apresentar uma forma de obter maior desempenho para uma política adaptativa de gerência de memória, por meio da adoção de técnicas complementares e da inclusão de informações, sem a necessidade de modificar a política e seu comportamento original. Isso levou à criação de um novo algoritmo chamado LRU-WARlock+ng, que utiliza a mesmas técnicas do LRU+ng [Piantola and Midorikawa 2009].

Este artigo está organizado da seguinte forma: a seção 2 discute as técnicas no auxílio às políticas, descrevendo sua importância e adequação em sistemas de gerência de memória. A seção 3 descreve o novo algoritmo adaptativo proposto. A avaliação de desempenho realizada sobre o algoritmo LRU-WARlock+ng é apresentada na seção 4. A seção 5 finaliza o artigo e traz as principais conclusões, indicando alguns trabalhos futuros.

2. Técnicas de *profiling*, *prefetching* e n-gramas aplicados a gerência de memória

Grande parte dos sistemas de gerência de memória utiliza a técnica de paginação por demanda, onde somente a página responsável pela falta é trazida para a memória. No caso da busca antecipada de páginas ou pré-paginação (*prefetching*), 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 contexto dos algoritmos adaptativos de substituição de páginas, toda vez que um programa é executado, é necessário que o algoritmo se adapte as referências à memória na execução atual do programa. Isso tem um custo relativo alto, até que a adaptação ocorra. No meio da execução do programa, podem ocorrer outros padrões de acessos, e o algoritmo levará um tempo até se adaptar ao novo padrão encontrado. Uma alternativa para evitar o tempo de adaptação e faltas de páginas desnecessárias, seria utilizar a técnica de *profiling*. Essa técnica possibilita capturar informações em tempo de execução do programa, e usar esse conhecimento em suas próximas execuções.

No sistema operacional Windows Vista, da Microsoft, foi criada uma tecnologia denominada por eles *SuperFetch*. O *SuperFetch* utiliza, ao mesmo tempo, *prefetching* e *profiling*; ele reconhece o programa de uma execução anterior e, através da técnica de *profiling*, grava o padrão de referencias as páginas. Nesse momento, o sistema operacional tenta pré-paginar a maior parte das páginas necessárias.

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 de Biologia, especificamente, genética [Manning and Schütze 1999].

Dado uma sequência, um n-grama é uma sub-sequência de n elementos. Um modelo de trigrama foi usado no sistema de reconhecimento de voz chamado IBM TANGORA já na década de 70 [Jurafsky and Martin 2008].

3. Proposta de auxilio a política LRU

O objetivo principal da proposta apresentada neste artigo é criar uma estratégia para auxiliar políticas adaptativas 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-WARlock+ng, que auxilia a política de substituição LRU-WARlock, sem modificá-la, somente utilizando informações de relação entre páginas e adotando a técnica de busca antecipada de páginas (*prefetching*).

O princípio do funcionamento do LRU-WARlock+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.

O mecanismo utilizado pelo LRU-WARlock+ng é a busca antecipada de páginas [Silberschatz et al. 2005]. No LRU-WARlock+ng, as páginas são selecionadas de acordo com o modelo de n-gramas que é utilizado para, ao mesmo tempo separar frequencia e tentar prever qual página será acessada imediatamente após a atual. Este modelo foi testado com sucesso no algoritmo não adaptativo LRU+ng. [Piantola and Midorikawa 2009]

3.1. Algoritmo LRU-WARlock+ng

O algoritmo LRU-WARlock+ng adiciona uma técnica e uma informação à política utilizada no LRU-WARlock: a técnica de busca antecipada de páginas e a informação de relacionamento entre páginas. Mesmo com esses mecanismos de auxílio, a política de substituição de páginas não foi modificada, ou seja, a vítima continua sendo a página escolhida pelo algoritmo adaptativo LRU-WAR.

Antes de iniciar a descrição do algoritmo desenvolvido para esse trabalho, se faz necessário a descrição do algoritmo em que é baseado. O algoritmo LRU-WARlock está dividido logicamente em duas partes: o LRU-WAR sem modificação e uma nova parte que utiliza a técnica de profiling como auxílio, mais o novo parâmetro de controle K [Piantola and Midorikawa 2008]. A parte referente ao LRU-WAR continua monitorando os acessos à memória, entre duas faltas de páginas consecutivas, utilizando a dimensão máxima da área de trabalho como fator decisivo de adaptabilidade. Existem três estados possíveis de execução definidos: tendência LRU, tendência sequencial e operação sequencial. Na operação padrão, tendência LRU, o critério adotado para a substituição de páginas é o LRU, o mesmo se aplica à tendência sequencial. Quando são detectados acessos sequenciais, o algoritmo entra em operação sequencial, sendo adotado para a substituição de páginas MRU-n. A Região Reservada, é composta pelas páginas referenciadas com maior frequência, e está fora de todo

o processamento descrito na parte do LRU-WAR. O tamanho da Região Reservada é definido pelo novo parâmetro de controle K, em porcentagem do total de páginas. A Região reservada é construída utilizando a técnica de *profiling*.

A operação do LRU-WARlock+ng é a mesma do algoritmo LRU-WARlock. No caso em que se adota a busca antecipada no momento da falta de página, tem-se então que as páginas representadas pelas letras “t” e “u” ou “y” e “z” (respectivamente, operação sequencial e tendência LRU) serão as vítimas da substituição, a página “a” será aquela da busca antecipada e a página “b”, a responsável pela falta (Figura 1).

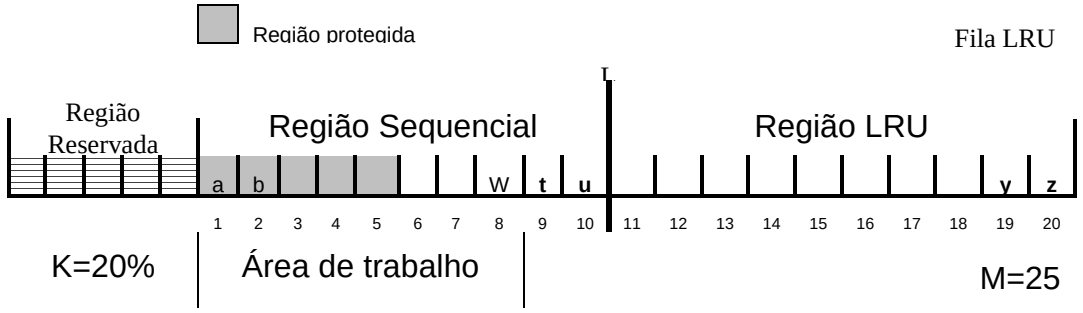


Figura 1. Estrutura do algoritmo LRU-WARlock+ng.

A busca antecipada de páginas pode acontecer ou não quando ocorre uma falta de página, isso depende 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-WARlock. A escolha da página da busca antecipada é feita através do método de bigramas. Para cada duas referências consecutivas é guardada a frequência em que essas duas páginas são referenciadas novamente em sequência. A página que aparece com maior frequência após a referênciada é a escolhida para a busca antecipada. Exemplo: Se a sequência de acesso para a página 803 é a 813 e isto já ocorreu 20 vezes, e nenhuma outra página a supera, então a página 813 é a candidata a busca antecipada quando ocorre uma falta para a página 803. 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-WARlock. As novas condições adicionadas ao código são referenciadas nas linhas 21 até 24 e 29: a linha 21 verifica se a página para a busca antecipada encontra-se na memória. Caso negativo as linhas seguintes tratam de trazê-la para a memória e remover a página na fila (caso não tenha mais memória disponível); no pseudocódigo a linha 29 cria, exclui e atualiza a estrutura para busca antecipada a cada referência.

```
1. Se a página referenciada está na memória:
2.     Se o bit “page locked” for igual a zero:
3.           Se está em Operação Sequencial:
4.                 Termina a Operação Sequencial
5.                 Aumenta o tempo de carência
6.           Aumenta Área de Trabalho
7.           Reordena a página na primeira posição da fila.
8. Senão, se não está na memória e não está nas páginas do profile:
```

```
9.     Se a memória está cheia (memória total – memória reservada):
10.          Se estiver em Operação Sequencial
11.                Remove a página na posição (Área de Trabalho + 1) da fila;
12.          Senão, se estiver em Tendência Sequencial:
13.                Se a Área de Trabalho for menor que L e excedeu a carência
14.                      Entrar em Operação Sequencial
15.                Remove a página na posição LRU da fila;
16.          Senão (Está em Tendência LRU):
17.                Se Área de Trabalho for menor que L
18.                Entrar em Tendência sequencial
19.                Diminui Área de Trabalho e ajusta carência.
20.                Remove a página na posição LRU da fila;
21.   Se página do hash para busca antecipada existe e não está na memória:
22.                Remove a página na posição LRU ou MRU-n da fila.(LRU-WARlock)
23.          Carrega a página referenciada e a página da busca antecipada.
24.    Senão, se a página da busca antecipada está na memória:
25.          Carrega a página referenciada no início da fila.
26. Senão, se não está na memória e está no profile:
27.    Carrega a página referenciada
28.    Adiciona o bit "page locked" a página
29. Manutenção do hash para busca antecipada.
```

Figura 2. Pseudocódigo do algoritmo LRU-WARlock+ng.

É possível remover a função de manutenção do *hash* na linha 29 (Figura 2). Para isso é necessário incluir a técnica de *profiling* [Piantola and Midorikawa 2008], deixando assim o código mais rápido e com maior precisão desde o início de sua segunda execução.

4. Análise do Desempenho

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

4.1. Caracterização dos *Traces* Utilizados

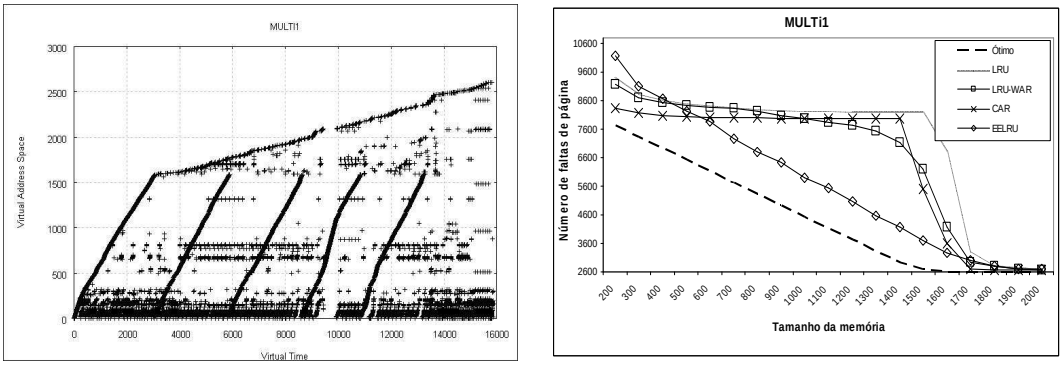
A avaliação do LRU-WARlock+ng foi efetuada usando três diferentes *traces* de programas, são eles: multi1, multi2 e multi3 (Tabela 1). Os três *traces* [Kim et al 2000] foram selecionados pelo fato de conterem acessos simultâneos de aplicações, simulando um ambiente multiprogramado. As ferramentas utilizadas nas simulações fazem parte do ambiente Elephantools [Cassettari and Midorikawa 2004a].

Tabela 1. Descrição dos *traces* utilizados.

Trace	Descrição	Origem	Total de 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

A composição dos traces utilizados nos testes encontra-se na Tabela 1, além de uma breve descrição dos padrões de acesso e seu comportamento com alguns algoritmos de substituição apresentados na literaturam como o LRU e o LRU-WAR, entre outros.

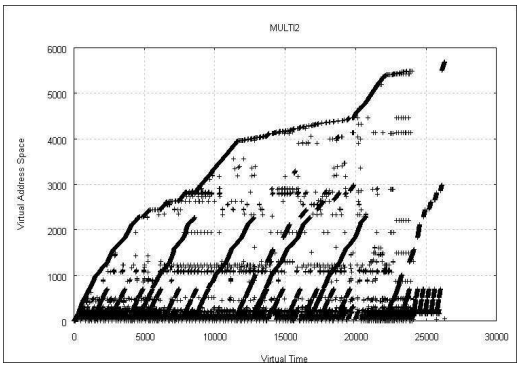
O *trace* **multi1** é composto pelas aplicações cscope e cpp. O cscope é uma ferramenta interativa de verificação de programa fonte escrita em linguagem C. Seu padrão de acesso à memória faz referências a *looping* com forte localidade temporal e outras referências de padrão diverso (Figura 3.a). O cpp é o pré-processador do GNU C, onde durante sua execução pode ser observado 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 seqüencial. Este padrão de acesso prejudica muito o desempenho do algoritmo LRU [Jiang and Zhang 2002], 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.



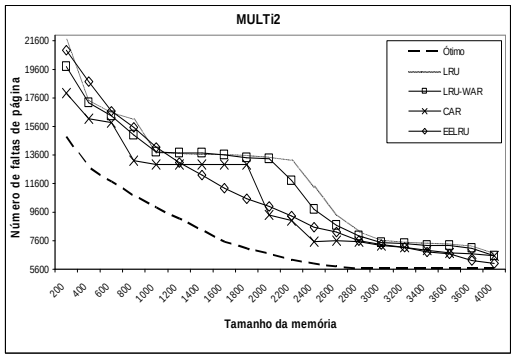
(a) Padrão de acessos à memória. (b) Desempenho de alguns algoritmos.

Figura 3. Características do *trace* multi1.

O *trace* **multi2** é composto pelas mesmas aplicações do *trace* multi1 com a adição do programa postgres. O postgres é um sistema de banco de dados relacional da Universidade da Califórnia. 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 uma maneira geral, a maioria dos algoritmos apresenta um desempenho muito próximo ao LRU.



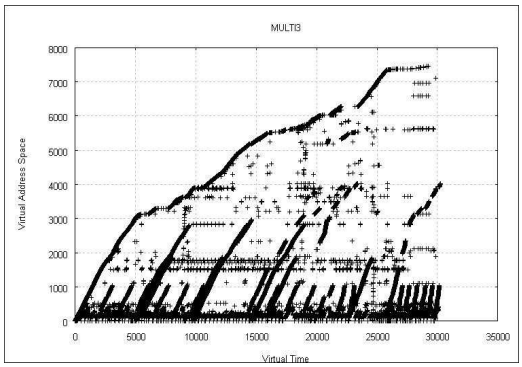
(a) Padrão de acessos à memória.



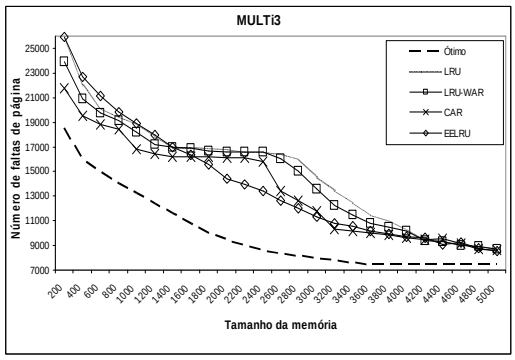
(b) Desempenho de alguns algoritmos.

Figura 4. Características do *trace* multi2.

O último *trace*, o **multi3**, tem a composição 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. Seu padrão de referência a memória é bem diverso. Já o gnuplot tem um padrão de acessos seqüencial 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

Na atual seção apresentamos, de acordo com os objetivos definidos neste artigo, as análises e os resultados do estudo do desempenho do algoritmo LRU-WARlock+ng, que utiliza *profiling*, o modelo de bigramas e busca antecipada de páginas.

Multi1

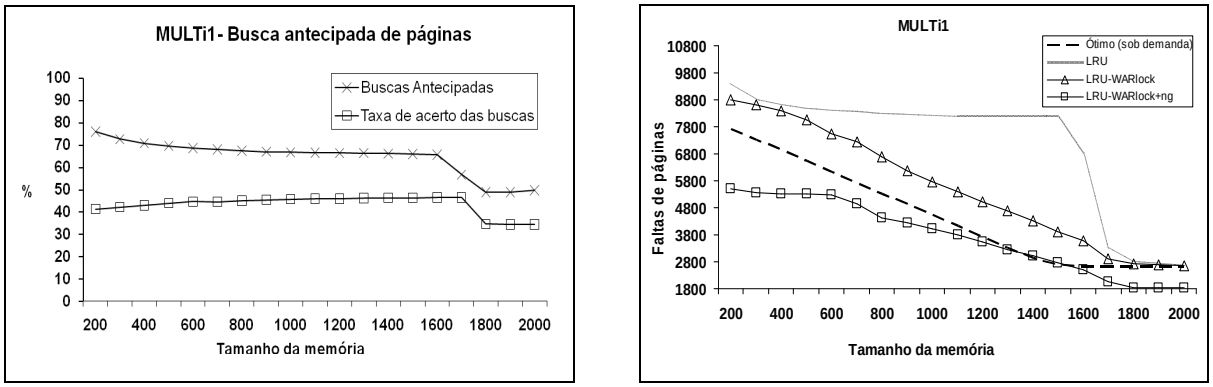
O algoritmo LRU-WAR, o qual é a base do LRU-WARlock+ng, tem a característica do LRU, que consegue prever rasuavelmente o comportamento geral dos programas, assim, atingindo bom desempenho na maioria dos padrões de acesso a memória. Em um programa com padrões de acessos sequenciais e os que contem grandes *loops*, o LRU não atinge um desempenho bom, o LRU-WAR trata essa deficiência. Como o *trace* multi1 apresenta acessos de duas aplicações intercaladas em um sistema de gerência de memória global, o LRU

apresentou desempenho não muito bom, mesmo porque os dois programas que compõem o multi1 apresentam padrões de acessos seqüenciais (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 *profiling* e modelo dos n-gramas, irá proporcionar menos faltas de páginas. O LRU-WARlock+ng supera o LRU-WARlock em todos os tamanhos de memória, com ganhos chegando até 37% (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-WARlock+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-WARlock+ng é que ele supera o algoritmo Ótimo em todos os tamanhos de memória menores a 1200 páginas. O algoritmo Ótimo, também conhecido como OPT ou MIN, se refere ao algoritmo teórico que apresenta o melhor desempenho possível entre os algoritmos de paginação sob demanda [Silberschatz et al. 2005]. A razão para seu melhor desempenho em relação ao algoritmo Ótimo é o fato do LRU-WARlock+ng não ser um algoritmo de paginação sob demanda, pois ele 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-WARlock+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 os algoritmos

Figura 6. Gráficos do *trace* multi1.

É possível dessa forma melhorar os algoritmos sem a necessidade de modificar a política vigente.

O que marca a grande eficiência do LRU-WARlock+ng em memórias de pequeno tamanho, é que quando temos pouca memória e um número grande de páginas do programa, a

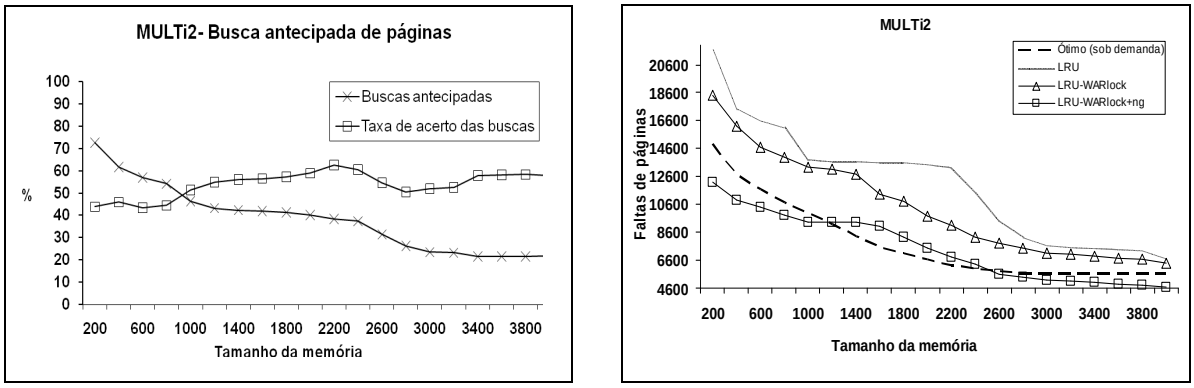
probabilidade das páginas acessadas estarem fora do memória principal é alta. Por esse motivo, do total de faltas de páginas, em 74% dos casos é utilizada a busca antecipada de página, ou seja, a página mais acessada após a página que ocasionou a falta também é colocada na memória. Já em tamanhos de memórias grandes isso ocorre em apenas 17% das faltas.

Multi2

A principal característica do *trace* multi2 para a análise surge do fato de ser semelhante ao multi1. O que difere os dois *traces* é a adição do programa postgres. O desempenho do LRU-WARlock+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 *profiling* e modelo dos n-gramas, proporciona menos faltas de páginas também neste caso.

O LRU-WARlock é superado pelo LRU-WARlock+ng em todos os tamanhos de memória. O mesmo acontece com o algoritmo Ótimo, que é superado pelo LRU-WARlock+ng em tamanhos de memórias pequenas. Essa eficiência em memórias de pequeno tamanho é determinada pelo fato do algoritmo não encontrar a página da busca antecipada na memória e traze-la para a memória em quase 80% das vezes, contra 20% 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) Buscas sobre total de Faltas (b) Comparação entre os algoritmos

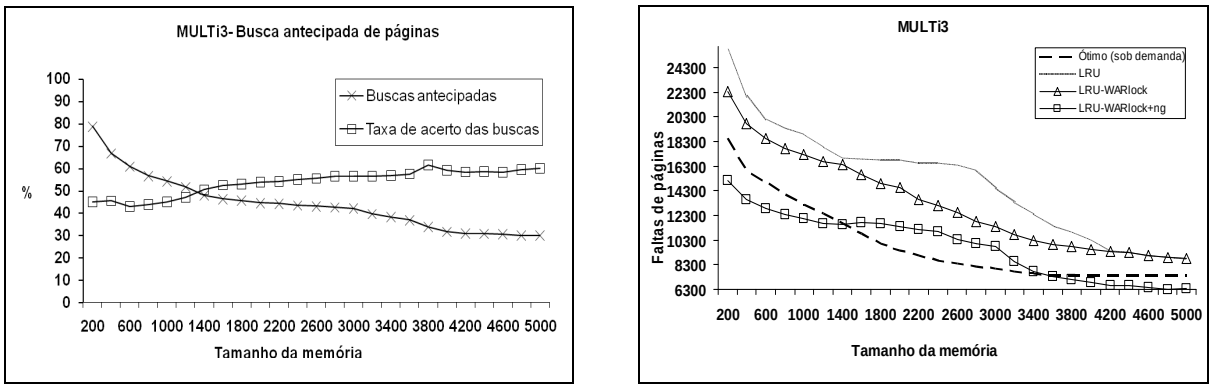
Figura 7. Gráficos do *trace* multi2.

É possível notar que na Figura 7.a, quanto maior é o tamanho da memória, menos buscas antecipadas são feitas, porém a precisão aumenta (Figura 7.b). Isto 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-WARlock+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 sequência de referências o LRU-WARlock não é bom, porém o LRU-WARlock+ng recupera essas páginas que foram descartadas prematuramente e obtendo melhor desempenho (Figura 8.b).

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 baixa. Porém o desempenho é muito melhor, comparando com as memórias de tamanho médio. O oposto acontece com 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. Uma 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. Primeiro, o *trace* intercala acessos de quatro aplicações, 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 os algoritmos

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. Duas outras possibilidades de aumentar a taxa de busca antecipada e melhorar o desempenho é, além de utilizar os bigramas, adotar-se os trigramas. A segunda seria diminuir o parâmetro k do LRU-WARlock+ng, que por sua vez diminuiria o tratamento de frequência pelo *profiling*, deixando esse tratamento ao encargo da técnica de n-gramas.

5. Conclusões e Trabalhos Futuros

Este trabalho apresentou um estudo sobre estratégias para auxiliar políticas adaptativas de substituição páginas em um sistema de gerenciamento de memória global. Este estudo foi conduzido sobre o algoritmo LRU-WARlock modificado, que foi chamado LRU-

WARlock+ng, o qual aplicou as técnicas de *profiling*, 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. Foram utilizados três *traces* com características de multiprogramação para os estudos de desempenho com políticas de gerência de memória global.

Neste trabalho, 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 adaptativa vigente, melhorar seu desempenho através 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. A prova desta 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-WARlock+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.

Diferente dos resultados sobre o LRU+ng, que não usou uma política adaptativa como base, o LRU-WARlock+ng não manteve a mesma característica nas faltas de páginas comparando com seu algoritmo base LRU-WARlock. A adição de técnicas e informações altera o comportamento dos algoritmos adaptativos de substituição de páginas, no que diz respeito a suas escolhas de substituição.

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 aumente as buscas antecipadas em memórias de maior tamanho, assim, diminuindo faltas de páginas posteriores.

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*.

Referências

Bansal, S. and Modha, D. S. (2004) "*CAR: Clock with Adaptive Replacement*", In Proc. of the USENIX Conference on File and Storage Technologies (FAST'04), San Francisco, pp.187-200.

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.

Glass, G. and Cao, P. (1997) "*Adaptive Page Replacement Based on Memory Reference Behavior*", In Proc. of the ACM International Conference on Measurement and Modeling of Computer Systems (SIGMETRICS'97), Seattle, pp.115-126.

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.

Lee, D. et al. (2001) "*LRFU: a spectrum of policies that subsumes the Least Recently Used and Least Frequently Used policies"*. IEEE Transactions on Computers, vol.50, n.12, p.1352-1361.

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.

Piantola, R.L. and Midorikawa, E.T. (2009) "*Uso de Técnicas e Informações para Potencializar Políticas de Substituição em Sistemas de Memória Virtual*". In: Anais do VI Workshop de Sistemas Operacionais (WSO 2009), Bento Gonçalves, RS.

Sabeghil, M. and Yaghmaee (2006), M. H. "*Using fuzzy logic to improve cache replacement decisions*". IJCSNS International Journal of Computer Science and Network Security, Seoul, v.6, n.3A, pages182-188.

Silberschatz, A., Gagne, G. and Galvin, P.B. (2005) *Operating System Concepts*. 7th 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.