Aplicação e Análise de Teoria de Controle Realimentado no Tratamento de Faltas de Páginas em Sistemas de Gerenciamento de Memória

Ivo Santos Cavalcante Carneiro[1] **Luciano Porto Barreto**[1] **Alirio Santos de Sá**[1]

ivo.cavalcante@gmail.com, {lportoba,aliriosa}@ufba.br

[1]Departamento de Ciência da Computação
Universidade Federal da Bahia
Campus de Ondina, 40170-110, Salvador-BA, Brasil

Abstract. *The static nature of traditional page replacement algorithms make them unsuitable to handle sudden workload changes or variations in memory access patterns. Several adaptive algorithms have been designed in order to cope with this inability, however, they still lack flexibility to handle unpredicted scenarios. This paper presents a new approach, based on Control Theory, which is able to both properly manage OS paging as well as to adapt to disturbances introduced into the system.*

Resumo. *O comportamento estático dos algoritmos tradicionais de substituição de páginas os tornam inapropriados em lidar com mudanças súbitas na carga de trabalho ou variações nos padrões de acesso à memória. Diversos algoritmos adaptativos foram projetados com o intuito de suplantar essa deficiência. Entretanto, tais algoritmos possuem limitada flexibilidade quando da necessidade de lidar com cenários imprevisíveis. Este trabalho apresenta uma nova abordagem, baseada na Teoria de Controle, capaz de gerenciar de forma apropriada o processo de paginação, bem como adaptar-se aos distúrbios introduzidos no sistema.*

1. Introdução

É prática comum, nos sistemas operacionais modernos, a abstração ou virtualização da memória como amplificador da memória física disponível aos processos, através da utilização temporária de outros dispositivos de armazenamento. Reduzir a quantidade de faltas de página geradas – responsabilidade do *gerenciador de memória* – é tarefa fundamental para garantir desempenho adequado desse tipo de ambiente.

As abordagens clássicas para substituição de páginas de memória – LFU, LRU, MRU, etc. – utilizam critérios estáticos na escolha de uma página a ser substituída: remover sempre a página com menor freqüência de acessos ou aquela acessada há mais tempo, por exemplo. A principal desvantagem dessas técnicas reside na sua baixa capacidade de se adaptar a mudanças no padrão de acessos de memória dos processos. Os algoritmos adaptativos – LRU-WAR [Cassettari 2004], EELRU [Smaragdakis et al. 2003], entre outros – representam uma abordagem mais recente, modificando seu comportamento de forma a reagir mediante alterações na carga de trabalho. Tenta-se evitar, dessa forma, a indesejável degradação no desempenho do sistema quando um ou mais processos apresentam comportamento para o qual algoritmos tradicionais seriam ineficientes. Embora

mais flexíveis, os algoritmos adaptativos ainda apresentam características estáticas, evidenciadas pela definição de parâmetros de controle necessários ao seu funcionamento. Boa parte das novas soluções exige que os valores de tais parâmetros sejam informados na etapa de projeto, reduzindo a capacidade do algoritmo de adaptar-se a novas situações.

Tradicionalmente, os problemas relacionados à área de controle envolvem a gestão de características físicas de plantas reais tais como: a velocidade de rotação de um motor e a pressão em caldeiras industriais [Ogata 1990]. No âmbito de sistemas computacionais, a teoria de controle tem sido empregada com sucesso em diversas áreas, principalmente, com o intuito de obter o desempenho desejado face às alterações na dinâmica do sistema. Alguns exemplos incluem sua aplicação na gestão da carga de servidores web, escalonamento adaptativo de processos e controle de congestionamento em redes de comunicação[1]. O uso de teoria de controle em sistemas computacionais é ainda emergente. Entretanto, o emprego dessa técnica na gerência de memória pode trazer consigo diversos benefícios: capacidade de adaptação a mudanças no ambiente, robustez a imprecisões no modelo matemático usado na descrição do sistema e ferramental para o estudo da estabilidade do algoritmo de adaptação.

Nesse contexto, este trabalho tem como objetivo verificar a viabilidade do uso de controladores na gestão de páginas de memória. Para tanto, nosso objetivo e contribuição principal consiste no desenvolvimento de um modelo baseado na Teoria de Controle que demonstre ser capaz de lidar com a substituição de páginas de forma eficiente.

O restante desse artigo está estruturado da seguinte maneira. A seção 2 apresenta uma breve introdução sobre o problema de gerenciamento de memória e descreve o algoritmo *Page Fault Frequency* (PFF), utilizado no decorrer desse trabalho. As seções 3 e 4 apresentam uma breve introdução ao estudo da Teoria de Controle e a proposta fundamental desse trabalho: o projeto e desenvolvimento do algoritmo PFF controlado. Na seção 5 são descritas a metodologia utilizada na condução do experimento e a discussão dos resultados obtidos. A seção 6 apresenta outras propostas adaptativas correlatas a nossa abordagem. Por fim, a seção 7 conclui o trabalho, apresentando considerações acerca do emprego de Teoria de Controle nos sistemas computacionais.

2. Problemática do Gerenciamento de Memória Virtual

No contexto do gerenciamento de memória virtual, os algoritmos de substituição são utilizados pelo gerenciador de memória de um sistema paginado, no momento em que uma página de memória precisa ser descartada para dar lugar a uma outra. Quando o sistema se encontra sob alta carga de trabalho, esta tarefa pode ocorrer muitas vezes em um único segundo. Um algoritmo desse tipo precisa, portanto, ser eficiente na sua implementação, para que o sistema operacional não ocupe tempo do processador que poderia ser destinado a processos que fornecem serviços aos usuários.

Um critério básico utilizado na classificação dos algoritmos de substituição é o espaço de memória sobre o qual eles trabalham: fixo ou variável [Cassettari 2004]. Algoritmos de espaço fixo assumem que a quantidade de memória disponível para o processo não muda. Os de espaço variável, por sua vez, precisam considerar a possibilidade de

[1] Alguns exemplos podem ser encontrados em [Hollot et al. 2001],[Hellerstein et al. 2004],[Diao et al. 2005] e [Shah et al. 2008].

aumento ou diminuição na quantidade de páginas alocadas, tornando a computação mais complexa.

O principal problema encontrado na elaboração dos algoritmos é escolher a "melhor" página candidata para descarte. No contexto da memória virtual, a melhor candidata é, geralmente, aquela que mais tempo levará para ser novamente referenciada — *principle of optimality*, exposto por [Denning 1970]. O algortimo OPT, proposto por [Belady et al. 1969], segue esse princípio, embora não seja implementável, pois parte da premissa que a sequência de acessos à memória de um processo é conhecida antecipadamente, o que é impraticável. Os algoritmos mais eficientes são aqueles que tentam especular qual página candidata obedece à regra acima.

Alguns algoritmos, sugeridos mais recentemente, têm capacidade de modificar seu comportamento em função de alterações no ambiente de trabalho. Tais, enquadram-se na classe dos *algoritmos adaptativos* [Cassettari 2004]. Em termos básicos, eles tentam ajustar-se aos diferentes estados de um sistema, buscando obter sempre a melhor solução para cada caso. Aqueles que não se enquadram na categoria acima serão denominados, neste estudo, *tradicionais*.

A análise da seqüência de páginas referenciadas por um processo durante sua execução costuma revelar um determinado padrão de acesso. Dois importantes conceitos a serem considerados na análise e projeto de algoritmos de substituição são o *Princípio da Localidade* e o modelo de *Conjunto Funcional* — (*Working Set*). Informalmente, *localidade* significa que, durante qualquer intervalo de execução, um programa prioriza um subconjunto de suas páginas e esse conjunto priorizado modifica-se lentamente. Esse comportamento é explicado pela forma com que os programas são construídos [Denning 1970]. O conjunto funcional, por sua vez, engloba a menor coleção de páginas que precisa estar na memória para que um processo possa executar. Como não há informação, por parte do programador ou compilador, acerca de tal conjunto, o sistema operacional o define como o conjunto de páginas referenciadas mais recentemente [Denning 1968].

2.1. Algoritmo PFF

O algoritmo PFF — *Page-Fault-Frequency* — proposto por [Chu and Opderbeck 1976] pertence à classe dos que atuam sobre o tamanho do conjunto de páginas residentes na memória. Seu principal objetivo é garantir uma adequada relação espaço-tempo nos processos, aumentando a quantidade de molduras alocadas para aqueles que apresentam altas taxas de falta de página (T.F.), e reduzindo para aqueles em que a taxa encontra-se muito baixa. Intuitivamente, pode-se dizer que há uma tentativa de evitar desperdício de recursos, já que, a partir de um determinado estado, não há redução considerável na quantidade de faltas associada a um aumento do conjunto residente.

O PFF utiliza a frequência de faltas de página medida como parâmetro básico do processo de decisão sobre alocação de memória. Geralmente, uma frequência alta de faltas de página indica que um processo está executando de maneira ineficiente porque possui poucas molduras de memória [Chu and Opderbeck 1976]. Uma taxa muito baixa indica exatamente o oposto: excesso de molduras alocadas. Alternando aumentos e reduções no conjunto residente, o método tenta associar ao processo apenas a quantidade de molduras necessária para assegurar sua execução de forma eficiente.

Para orientar seu funcionamento, o algoritmo define um parâmetro $P = \frac{1}{T}$ que representa a frequência de faltas por unidade de amostragem — 1000 referências, no estudo original. O parâmetro T é chamado *interpage fault time* e representa o tempo decorrido (medido em referências) entre duas faltas de página consecutivas. Caso um processo esteja operando acima do parâmetro P, as novas páginas referenciadas (as que geraram faltas) são *acrescentadas* às que já se encontram em memória, sem haver descarte de nenhuma das anteriores (aumento do conjunto residente). Quando a frequência de faltas encontra-se abaixo de P, todas as páginas não referenciadas durante as duas últimas faltas são descartadas, reduzindo o tamanho do conjunto residente. Essa decisão ocorre sempre que um processo falta, causando uma variação constante da quantidade de páginas alocadas.

3. A abordagem Baseada em Controle Realimentado

A solução apresentada neste trabalho — denominada *Algoritmo PFF controlado* — baseia-se no PFF original incluindo um controlador, utilizado para ajustar o funcionamento do sistema durante sua execução. O objetivo da estratégia é ajustar o tamanho do conjunto residente de modo a manter o desempenho médio de tal algoritmo dentro de um limiar esperado. Assim, dado um valor de referência (r_k) representando a taxa de faltas de página desejada, o PFF controlado atua sobre o tamanho (u_k) do conjunto residente do processo, de modo que a taxa efetiva de faltas (y_k) mantenha-se o mais próximo possível desse valor (figura 1).

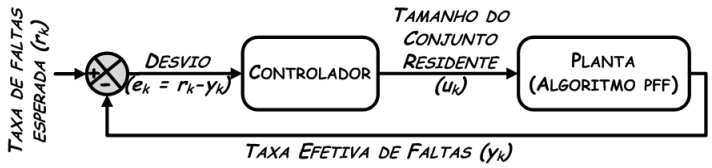

Figura 1. Representação da Malha de controle para o algoritmo PFF Controlado

A cada período de tempo, medido em quantidade de referências a memória, o PFF controlado calcula o desvio (e_k), entre as taxas médias de faltas obtida (y_k) e a desejada (r_k), e determina o tamanho do conjunto residente a ser usado durante o próximo período.

3.1. Lógica do Laço de Controle

A lógica do laço de controle envolve a regulação da taxa média de faltas obtidas para a planta (algoritmo PFF): mantendo $e_k \approx 0$ para todo k. Assim, se e_k é negativo então $y_k > r_k$, logo u_k deve ser aumentado, significando um aumento no conjunto residente. Por outro lado, se e_k é positivo, u_k deve ser diminuído, significando, por sua vez, uma diminuição no conjunto residente.

3.2. Modelo da Planta

Para a representação da planta, optou-se por utilizar por um modelo ARX (***A**uto-**R**egressive e**X**ogenous*) de primeira ordem[2], cuja forma é:

[2]A ordem do modelo está relacionada com a quantidades de amostras passadas de y ou u. Assim, primeira ordem significa que apenas o último valor de y ou u é usado na estrutura do modelo.

$$y_{k+1} = a \cdot y_k + b \cdot u_k \tag{1}$$

Um modelo ARX é uma equação de diferenças que relaciona funções lineares dos históricos de duas variáveis [Hellerstein et al. 2004]. Na equação 1, y representa a saída da planta (taxa de faltas de página), enquanto u corresponde à entrada (tamanho do conjunto residente). O índice k define os instantes, no tempo, aos quais cada valor de y e u estão associados. Os coeficientes a e b, por sua vez, são chamados *parâmetros do modelo*. Analisando-se a expressão apresentada, percebe-se que relaciona a saída da planta no instante $k + 1$ à saída e entrada no instante anterior, k.

De posse do modelo apresentado na equação 1, é possível derivar a função de transferência da planta, usando a transformada Z [Ogata 1995]:

$$P(z) = \frac{Y(z)}{U(z)} = \frac{b}{z - a} \tag{2}$$

A equação 2 é a representação, no domínio Z, do modelo definido pela equação 1. Essa notação é bastante utilizada no estudo de sinais porque facilita a manipulação algébrica dos mesmos. Posteriormente, aplicada-se a trasformada Z inversa para converter as expressões encontradas de volta para o domínio do tempo.

A estrutura do modelo, bem como os valores dos parâmetros a e b foram estudadas e validadas usando traces de freqüência de páginas de modo a encontrar aqueles que minimizassem o erro quadrático entre os dados no trace e os dados sugeridos pelo modelo[3].

3.3. Definição do Controlador

Controladores *PID* (*Proporcional-Integral-Derivativo*, ver [Ogata 1990]) e suas variações (*P*, *PD* e *PI*) têm encontrado grande aceitação na automação de processos industriais e nas mais diversas aplicações das mais diversas áreas[4]. Neste trabalho, considera-se um controlador PI, cujo modelo ARX e a função de transferência em Z podem ser descritos por [Hellerstein et al. 2004]:

$$u_k = K_p \cdot e_k + K_i \cdot \sum_{i=0}^{k} e_i \tag{3}$$

e

$$C(z) = K_p + K_i \cdot \left(\frac{z}{z - 1} \right) \tag{4}$$

Os parâmetros u_k, e_k, K_p e K_i representam, respectivamente, a saída do controlador, o desvio entre a resposta desejada e a obtida após a ação de controle e os ganhos proporcional e integral do controlador.

[3]Uma descrição mais detalhada do processo de identificação do modelo pode ser encontrado em [Draper and Smith 1998].

[4]Como exemplo, podem ser citados: [Hollot et al. 2001], [Fengyuan and Chuang 2003], [Majumdar et al. 2003], [Majumdar et al. 2004] e [Ang et al. 2005], entre outros.

3.3.1. Sintonia do Controlador

O processo de sintonia do controlador deve considerar critérios objetivos para o desempenho do mesmo. Durante o projeto de controladores para sistemas computacionais, focase em quatro propriedades para o laço de controle [Hellerstein et al. 2004]: Estabilidade (Stability), Precisão (Accuracy), Tempo de convergência (Settling time) e Variação máxima (Overshoot).

Um sistema é dito estável se, para uma entrada limitada, produz uma saída limitada (ver [Hellerstein et al. 2004] e [Simon 2002]). Por sua vez, um sistema é dito preciso se consegue miminizar o erro entre a resposta desejada e a obtida (Steady-state error). O tempo de convergência (K_s) representa o tempo que o sistema leva para atingir o estado desejado. A variação máxima (M_p) é a diferença máxima entre o estado desejado e a resposta do sistema.

Para analisar cada uma dessas propriedades, é necessário avaliar a função de transferência ($F(z)$) em malha fechada do sistema, a qual é dada por:

$$F(z) = \frac{P(z)C(z)}{1 + P(z)C(z)} \tag{5}$$

A estabilidade do sistema é determinada pela localização dos pólos do laço de controle. Os pólos do sistema são a solução para a equação característica:

$$1 + P(z)C(z) = 0 \tag{6}$$

O sistema será estável se todos os pólos do sistema tiverem magnitudes menores que 1 [Simon 2002].

4. Projeto e desenvolvimento

O projeto de um algoritmo controlado é uma atividade que compreende diversas etapas. No caso de modelos caixa-preta, onde as leis matemáticas que definem o sistema não são conhecidas, o esforço dispendido é ainda maior, pois as relações entre as variáveis precisam ser escritas e quantificadas. Assim, para a identificação dos parâmetros do modelo, é necessária a obtenção de *cadeias de referências à memória*, denominadas *traces*. Essas cadeias representam os endereços lógicos das páginas de memória acessadas por um processo durante sua execução, e têm a seguinte forma: *1, 2, 4, 3, 8, 11, 8, 10, 11, 4, . . . , 51, 11, 3, 1*. Cada número representa uma página e a ordem na sequência representa o *tempo* em que se deu o acesso. No exemplo, o processo hipotético acessou a página 1 no tempo $t = 1$, 2 no tempo $t = 2$, 4 no tempo $t = 3$ e assim sucessivamente, até terminar na página 1 no tempo $t = n$, onde n representa o total de referências à memória.

Os traces utilizados no experimento foram obtidos a partir da execução de um programa, *Memory_Access_Generator* (*mag*), desenvolvido pelos autores, que utiliza critérios pseudo-aleatórios para gerar a cadeia. Seu funcionamento baseia-se nos valores dos seguintes parâmetros de entrada: *tamanho da amostra, endereço máximo acessado (limite superior), probabilidade do processo iniciar um laço, probabilidade de sair de um laço*

(quando em um), "tamanho do loop" (quantidade de endereços acessados em cada iteração), probabilidade de ocorrer alocação de memória (aumento na faixa de endereços utilizados, até o limite dado pelo endereço máximo), probabilidade de ocorrer desalocação de memória, fator de localidade de referências, probabilidade de mudança da localidade de referência. Os traces gerados contêm $1.000.000$ (um milhão) de referências. Os valores dos parâmetros de entrada foram variados entre cada trace, para tentar simular diferentes padrões de acessos. Embora não representem fielmente padrões de acesso reais, as cadeias utilizadas mostraram-se satisfatórias para os objetivos do experimento.

4.1. Descrição do experimento

Uma vez definidas as variáveis que comporiam o modelo, foi preciso verificar se havia relação entre as mesmas — ou seja, se mudanças no conjunto residente influenciam a taxa de faltas de página. De fato, essa relação já foi demonstrada por [Chu and Opderbeck 1976]. Ainda assim, foram realizados testes para comprovar a validade do modelo. Com o intuito de verificar a influência de uma variável sobre a outra, foram produzidos gráficos mostrando a evolução de ambas ao longo da execução de um processo hipotético representado, nesse caso, pelos traces gerados anteriormente. O algoritmo PFF controlado foi, então, implementado e utilizado para processar as cadeias.

Uma questão bastante importante na identificação de relacionamentos entre propriedades é a maneira como se modifica a variável independente, ao longo do experimento. O padrão de variação escolhido deve ser tal que as mudanças causadas na variável dependente possam ser percebidas graficamente. Idealmente, o padrão de variação pode ser suficiente não apenas para provar que existe influência, mas também permitir uma caracterização prévia do relacionamento. Seguindo-se recomendações de [Hellerstein et al. 2004], optou-se por impor um padrão senoidal de baixa frequência ao sinal de entrada, variando o tamanho do C.R. entre 1 e 1000 páginas. O período de amostragem utilizado para calcular a taxa de faltas foi de 500 referências à memória, e o passo de variação da senóide, em cada amostragem, foi definido como 0.002π. Para um trace com um milhão de referências, o período de amostragem gerou uma saída com 2000 pares (x, y), em que x representa o tamanho do C.R. e y a taxa de faltas de página correspondente. Uma vez comprovada a relação entre as duas propriedades, utilizou-se um dos traces gerados para calcular, pelo método dos mínimos quadrados, os valores dos parâmetros do modelo, obtendo-se $a = 0.3611$ e $b = 0.0008077$.

4.1.1. Projeto do controlador

O cálculo dos valores dos coeficientes K_p e K_i do controlador foi feito utilizando-se o processo descrito por [Hellerstein et al. 2004], de ajuste de parâmetros por *posicionamento dos pólos*. No método, são definidos valores máximos para o tempo de estabilização K_s e erro máximo M_p. Em seguida, calculam-se valores máximos para os coeficientes r e θ dos pólos da função de transferência de realimentação, que são utilizados — integralmente ou reduzidos, como margem de segurança — para calcular os valores K_p e K_i do controlador que satisfazem os requisitos definidos por K_s e M_p. Conjuntos diferentes de pares (K_p, K_i) foram calculados. Os gráficos gerados pelos controladores resultantes foram utilizados para definir, através de observação, os parâmetros que seriam utilizados na validação da solução. A tabela 1 apresenta o conjunto de valores obtidos para K_p e K_i.

Tabela 1. Cálculo dos parâmetros do controlador. As colunas K_s e M_p apresentam os requisitos *tempo de estabilização* e *erro máximo*, respectivamente. As colunas 4 a 7 apresentam os valores máximos obtidos para r e θ, e os valores dos mesmos utilizados no cálculo de K_p e K_i, cujos resultados são mostrados nas duas últimas colunas.

Conj.	K_s	M_p	r ***max***	θ ***max***	r ***usado***	θ ***usado***	K_p	K_i
1	13	0,1	0,7351	0,4198	0,65	0,35	76.0537	-249.2577
2	11	0,1	0,6951	0,4961	0,6	0,4	-1.3303	-315.3887
3	9	0,1	0,6412	0,6064	0,6	0,5	-1.3303	-379.9878
4	7	0,1	0,5647	0,7796	0,5	0,7	-137.5263	-600.6957
5	5	0,1	0,4493	1,0915	0,4	0,9	-248.9593	-820.5335
6	9	0,3	0,6412	1,1597	0,6	1	-1.3303	-881.1100
7	9	0,5	0,6412	2,0144	0,6	2	-0.0013	-2.3022

O controlador utilizado na validação foi ajustado com os parâmetros apresentados no conjunto 3 da tabela 1, escolhido a partir da observação de gráficos gerados nas simulações com cada par de valores (K_p, K_i). Para verificar sua estabilidade e precisão, o sistema foi submetido à T.F. referencial de 0,3 (*ponto de operação*). Em seguida, acrescentou-se distúrbio à saída do controlador na forma de um impulso com intensidade aleatória variando entre -70 e 70 páginas[5] e período de 15 amostras, atuando a partir do instante $t = 20$. O comportamento do sistema nas duas situações está representado nas figuras 2a e 2b. Os gráficos demonstram que o modelo resultante é estável e capaz de rejeitar distúrbios externos. A etapa seguinte consistiu na realização do mesmo teste na planta real, o algoritmo PFF controlado, processando uma cadeia de referências qualquer; as figuras 2c e 2d apresentam os resultados. A taxa de faltas média do algoritmo, operando sem distúrbio, foi de $0{,}2999$, aproximadamente igual ao valor de referência.

Nos testes seguintes, o controlador foi ajustado com o conjunto 5 de parâmetros da tabela 1, que definem um comportamento mais "agressivo" — o tempo de estabilização máximo é reduzido de 9 para 5 amostras. As figuras 3a e 3b mostram a atuação do controlador no modelo matemático (sob ação de distúrbio) e planta, respectivamente. Comparando-se as figuras 2b e 3a pode-se perceber que, na segunda, o sistema reage mais rapidamente às mudanças no ambiente, conforme evidenciado pelo formato mais agudo dos picos de distúrbio. Por outro lado, os efeitos do controlador sobre a planta degradaram o desempenho do sistema quando comparado ao teste anterior.

5. Análise e discussão dos resultados

O primeiro fator analisado após a conclusão dos experimentos foi a eficácia do algoritmo controlado, ou seja, sua capacidade de ajustar-se de modo a manter a taxa média de faltas de página próxima ao valor de referência. A tabela 2 apresenta os resultados do processamento, pelo PFF original e controlado, de arquivos de amostras gerados. A simulação, coleta e análise dos dados foi feita no ambiente MATLAB/Simulink. O controlador utilizado foi ajustado com os parâmetros do conjunto 3 da tabela 1, e a T.F. de referência foi definida como 0,3.

[5]Esse valor é adicionado ao ponto de operação do sistema, 140 páginas.

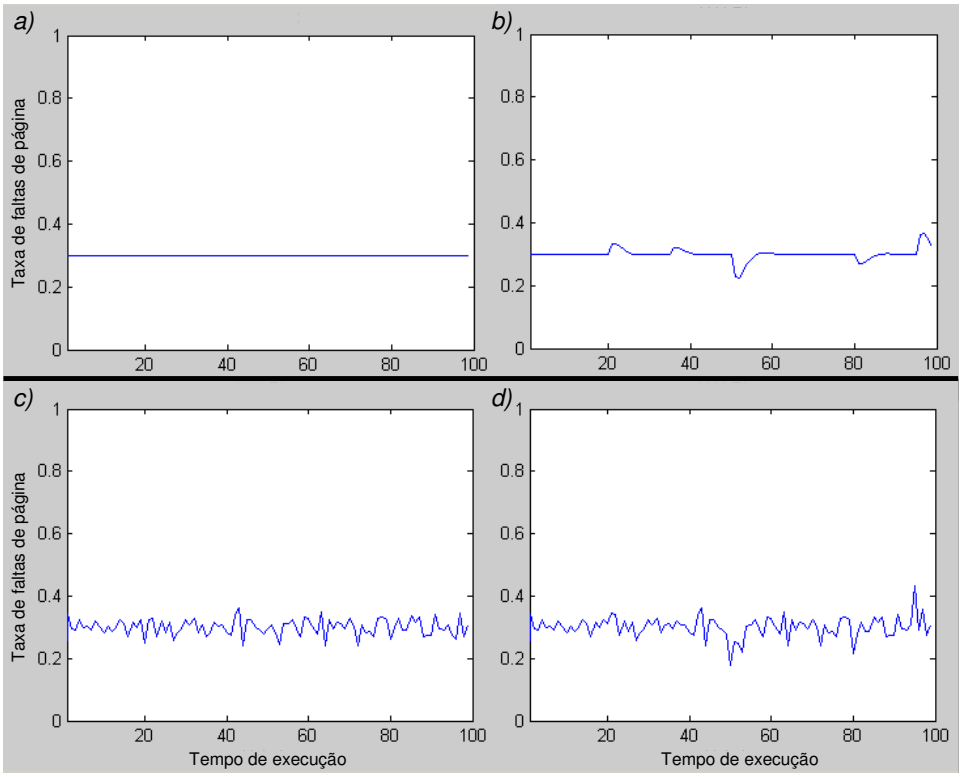

Figura 2. Controlador atuando sobre o modelo matemático e a planta real. Os gráficos *a* e *b* foram obtidos a partir do modelo matemático; o *c* e o *d* são oriundos do processamento, pelo PFF controlado, de uma cadeia de referências. Nos quadros *b* e *d*, o sistema foi submetido a distúrbios. Em todos os quadros, o eixo x representa o tempo virtual de execução e o y, a T.F. .

Observou-se que a T.F. do algoritmo original apresentou comportamento mais estável ao longo do experimento, conforme evidenciado pelos valores das colunas 4 e 8 (desvios padrão). Os valores mais altos obtidos pela versão controlada podem ser consequência do intervalo entre as atuações (10000 referências), mas tal hipótese precisa ser testada. Com relação à capacidade de atingir o valor de referência $(0{,}3)$, o algoritmo controlado mostrou-se mais preciso, conseguindo obter erros menores. É importante ressaltar que o parâmetro *P* do algoritmo original foi ajustado empiricamente de modo a obter taxa média em torno do valor de referência.

Além do controle da T.F., é importante analisar o desempenho das propostas em relação à quantidade total de faltas, visto que a redução deste fator é o principal objetivo das políticas de gestão de páginas. Também neste quesito os algoritmos obtiveram desempenho equiparado, fato que, mais uma vez, reforça a aplicabilidade de controladores para a tarefa proposta.

6. Trabalhos Correlatos

Até a escrita desse artigo, inexistem, em nosso conhecimento, algoritmos que apliquem a técnica de teoria de controle no algoritmo de paginação. A seguir descrevemos os trabalhos que mais se aproximam da nossa abordagem. De fato, tais algoritmos, assim como outros algoritmos adaptativos, poderiam valer-se da teoria de controle no cálculo

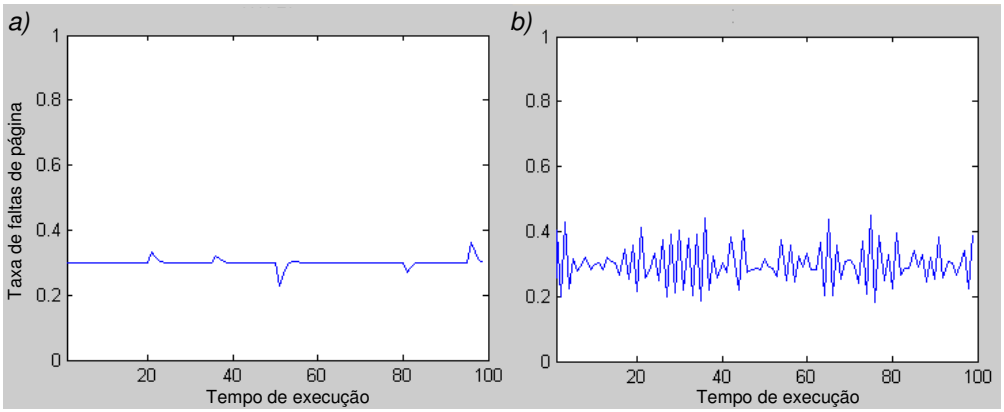

Figura 3. Efeito do uso de um controlador mais "agressivo". Os gráficos *a* e *b* demonstram o resultado da atuação sobre o modelo matemático (com presença de distúrbio) e a planta real (sem distúrbios), respectivamente. O eixo x representa o *tempo virtual de execução* e o y, a T.F..

Tabela 2. Comparativo entre desempenho dos algoritmos PFF original e controlado. As colunas para cada algoritmo são, respectivamente: total de faltas, média da T.F., desvio padrão e diferença (erro) entre a média de T.F. (colunas 3 e 7) em relação ao valor de referência ($0{,}3$).

Trace	PFF original				PFF controlado			
	T. faltas	Média T.F.	SD	Erro	T. faltas	Média T.F.	SD	Erro
1	287577	0,2876	0,0157	0,0124	299925	0,2999	0.0239	0,0001
2	295821	0,2958	0,0125	0,0042	300130	0,3001	0.0255	$-0{,}0001$
3	290654	0,2907	0,0161	0,0093	299338	0,2993	0.0299	0,0007
4	305041	0,3050	0,0165	$-0{,}0050$	299933	0,2999	0.0262	0,0001
5	312980	0,3130	0,0151	$-0{,}0130$	300411	0,3004	0,0254	$-0{,}0004$
6	361063	0,3611	0,0208	$-0{,}0611$	352000	0,3520	0,0257	$-0{,}0520$

automatizado de seus parâmetros de atuação.

O algoritmo EELRU — *Early Eviction* LRU*Early Eviction* LRU — pertence ao grupo dos que utilizam o LRU como base para seu funcionamento. No entanto, o EELRU procura lidar de maneira mais eficiente com as situações em que o seu predecessor é reconhecidamente ineficiente. Em seu trabalho, [Smaragdakis et al. 2003] observaram que padrões de acesso cíclicos — laços, por exemplo — em que o tamanho dos dados processados em cada iteração ultrapassa o da memória disponível, degradam o desempenho do LRU. Notaram, ainda, que dentro desses laços, as referências à memória não são, necessariamente, a endereços lógicos adjacentes. Dessa forma, métodos que identificam padrões baseando-se em análise dos endereços teriam sua capacidade de adaptação reduzida.

O funcionamento do EELRU baseia-se em uma abstração chamada *eixo LRU*, onde as páginas de memória são ordenadas segundo suas *recências de referência*. Essencialmente, esse é o mesmo modelo utilizado no algortimo LRU, com uma diferença: a fila de páginas compreende aquelas carregadas na memória e as recentemente descartadas. Cada posição no eixo corresponde a um valor de recência; a primeira posição, de valor 1, representa a página mais recente. Para cada posição no eixo, o algoritmo mantém

um contador de referências. O histograma de referências criado descreve o comportamento do algoritmo em relação à recência das páginas acessadas. Dessa forma, em um dado instante, é possível visualizar como estiveram distribuídas, em termos de recência, as referências às páginas de um determinado processo.

Experimentos realizados pelos autores mostraram que o EELRU apresenta-se, na maioria dos casos, mais eficiente que o LRU. Além disso, verificou-se que, nas poucas ocasiões em que é superado pelo LRU, o EELRU mantém essa diferença limitada a um fator constante, de no máximo 3. Uma outra vantagem do EELRU é sua possibilidade de detectar padrões baseados na recência das páginas acessadas, o que o torna capaz de adaptar-se a uma grande variedade de situações.

O algoritmo LRU-WAR [Cassettari 2004, Cassettari and Midorikawa 2004] — *LRU with Working Area Restriction*[6] — pertence à classe dos que emprega política LRU até que sejam detectados padrões de acesso sequenciais; nesse momento, modifica sua política para um MRU-n variável, que reconhecidamente trata melhor esses padrões. Fortemente influenciado pelos algoritmos SEQ [Glass and Cao 1997] e EELRU, o LRU-WAR distingue-se pelo método utilizado para identificar tais padrões: o SEQ busca por sequências nos endereços lógicos das páginas, enquanto o EELRU realiza análises na distribuição das referências sobre as recências das mesmas. O LRU-WAR, por outro lado, opta por utilizar o domínio das recências nas suas análises, e traduz o padrão de acessos sequenciais para esse novo domínio.

7. Considerações Finais

Este trabalho consiste num estudo preliminar do uso de teoria de controle aplicada ao gerenciamento de memória. O critério definido para comprovação da eficácia da abordagem baseou-se na capacidade de manutenção da média das faltas próxima ao valor determinado. A análise dos resultados experimentais obtidos reforça a crença na viabilidade de utilização da técnica de Teoria de Controle no contexto de gerência de memória, ao menos no campo teórico. Faz-se necessário um estudo da viabilidade prática da abordagem, através, por exemplo, da implementação do modelo apresentado em um sistema operacional. Tal realização permitiria uma avaliação da relação custo-benefício associada à utilização do controlador.

Um aspecto percebido no decorrer do trabalho residiu na grande dificuldade em descrever o comportamento de sistemas computacionais em termos de padrões de acesso à memória. Até o nosso conhecimento, inexistem leis definidas que expliquem seu funcionamento com precisão, o que pode ser justificado, ao menos em parte, pela natureza estocástica dos processos envolvidos. Nesse contexto, uma das grandes vantagens da aplicação de Teoria de Controle na gerência desses sistemas deve-se justamente à sua capacidade de lidar com ambientes cujas leis não são totalmente conhecidas. Por essa razão, acreditamos haver ainda muito espaço para a aplicação da técnica não apenas no campo da gestão de memória, mas em diversas áreas ligadas aos sistemas computacionais.

Referências

Ang, K., Chong, G., Li, Y., Ltd, Y., and Singapore, S. (2005). PID control system analysis, design, and technology. *IEEE Transactions on Control Systems Technology*,

[6]LRU com Confinamento da Área de Trabalho

13(4):559–576.

Belady, L., Nelson, R. A., and Shedler, G. S. (1969). An anomaly in space-time characteristics of certain programs running in a paging machine. *Communications of the ACM*, 12.

Cassettari, H. H. (2004). Análise da localidade de programas e desenvolvimento de algoritmos adaptativos para substituição de página. Master's thesis, Escola Politécnica da Universidade de São Paulo.

Cassettari, H. H. and Midorikawa, E. T. (2004). Algoritmo adaptativo de substituição de páginas LRU-WAR: Exploração do modelo LRU com detecção de acessos sequenciais. *Anais do XXIV Congresso da Sociedade Brasileira de Computação (SBC 2004)*, 1.

Chu, W. W. and Opderbeck, H. (1976). Program behavior and the Page-Fault-Frequency replacement algorithm. *IEEE Computer*.

Denning, P. J. (1968). The working set model for program behavior. *Communications of the ACM*, 11(5).

Denning, P. J. (1970). Virtual memory. *ACM Computing Surveys*, 2(3).

Diao, Y., Hellerstein, J., Parekh, S., Griffith, R., Kaiser, G., Phung, D., Center, I., and Hawthorne, N. (2005). A control theory foundation for self-managing computing systems. *IEEE journal on selected areas in communications*, 23(12):2213–2222.

Draper, N. and Smith, H. (1998). *Applied regression analysis*. Wiley Series in Probability and Statistics. Wiley-Interscience, 3 edition.

Fengyuan, R. and Chuang, L. (2003). Speed up the responsiveness of active queue management system. *IEICE Transactions on CommunicationsE86-B (2)*, pages 630–636.

Glass, G. and Cao, P. (1997). Adaptive page replacement based on memory reference behavior. *Proceedings of the 1997 ACM SIGMETRICS international conference on Measurement and modeling of computer systems*, pages 115–126.

Hellerstein, J. L., Diao, Y., Parekh, S., and Tilbury, D. M. (2004). *Feedback Control of Computing Systems*. Wiley-IEEE.

Hollot, C., Misra, V., Towsley, D., and Gong, W. (2001). A control theoretic analysis of RED. In *IEEE INFOCOM 2001. Twentieth Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings*, volume 3.

Majumdar, R., Moudgalya, K., and Ramamritham, K. (2003). Adaptive coherency maintenance techniques for time-varying data. *Real-Time Systems Symposium, 2003. RTSS 2003. 24th IEEE*, pages 98–107.

Majumdar, R., Ramamritham, K., Banavar, R., and Moudgalya, K. (2004). Disseminating dynamic data with qos guarantee in a wide area network: a practical control theoretic approach. *Real-Time and Embedded Technology and Applications Symposium, 2004. Proceedings. RTAS 2004. 10th IEEE*, pages 510–517.

Ogata, K. (1990). *Modern Control engineering*. Prentice Hall, Englewood Cliffs, 2nd edition.

Ogata, K. (1995). *Discrete-Time Control Systems*. Prentice-Hall, Upper Saddle River, NJ 07458, USA, 2nd edition.

Shah, S. B., Moudgalva, K. M., and Ramamritham, K. (2008). Feedback control of internet applications involving the tracking of dynamic data. In *Proc. 17th IFAC World Congress*, Seoul, Korea.

Simon, D. (2002). Analyzing control system robustness. *Potentials, IEEE*, 21(1):16–19.

Smaragdakis, Y., Kaplan, S., and Wilson, P. (2003). The EELRU adaptive replacement algorithm. *Performance Evaluation*, 53(2):93–123.