UM ESTUDO SOBRE O MECANISMO DE PAGINAÇÃO DE MEMÓRIA E OS ALGORITMOS DE SUBSTITUIÇÃO DE PÁGINAS FIFO E LRU

Fernando Sales Ferreira, fernandobrabat@hotmail.com
William Antônio Faria Da Silva, William_8716@hotmail.com
Eduardo Machado Real, eduardomreal@uems.br
Universidade Estadual de Mato Grosso do Sul/Nova Andradina

1. Introdução

Este trabalho aborda uma das principais abstrações realizadas por um Sistema Operacional (SO), o gerenciamento de memória e, mais especificamente a paginação. A ideia básica de um SO é controlar o funcionamento de um sistema computacional, gerenciando a utilização e o compartilhamento dos seus diversos recursos, como processadores, memórias e dispositivos de entrada e saída.

Todo programa precisa estar na memória para ser executado e com a multiprogramação, foi dado maior ênfase ao gerenciamento de memória em um SO, principalmente quando surge possibilidades de problemas como a fragmentação interna e externa. É função da gerência de memória prover mecanismos necessários para que os diversos processos compartilhem a memória de forma segura e eficiente.

O sistema de gerenciamento de memória está relacionado com memória lógica e física, sendo consideradas uma das partes mais críticas e difíceis de qualquer SO. Isso é motivado pelo aumento da necessidade do uso de mais memória pelos diversos sistemas atuais, limitada fisicamente. Devido a esta necessidade foi desenvolvido um método para melhorar este problema, foi criado a memória virtual, que realiza a tradução de endereços lógicos em endereços físicos, realizada por mecanismos implementados no próprio hardware do processador, que são as técnicas de Paginação e Segmentação.

2. Revisão teórica

Durante a evolução das arquiteturas de computadores e principalmente dos Sistemas Operacionais, muitas tecnologias tiveram que ser aprimoradas para

acompanhar esta evolução e permitir com que ela aconteça. A gerência de memória nos sistemas multiprogramáveis se torna mais crítica comparada aos sistemas monoprogramáveis, devido à necessidade de se aumentar o número de usuários e aplicações utilizando eficientemente o espaço da memória principal.

Quando se trata exclusivamente do gerenciamento de memória, Machado (2007) descreve que:

> "A gerência de memória deve tentar manter na memória principal o maior número de processos residentes, permitindo maximizar o compartilhamento do processador e demais recursos computacionais. Mesmo na ausência de espaço livre, o sistema deve permitir que novos processos sejam aceitos e executados. Isso é possível através da transferência temporária de processos residentes na memória principal para a memória secundária, liberando espaço para novos processos. Este mecanismo é conhecido como swapping".

Dois problemas podem surgir com algum tipo de alocação de memória e que o gerenciamento de memória procura resolver: fragmentação interna e externa. Quando um programa é carregado em uma área de memória maior que o necessário, isso resulta em um desperdício de memória que é chamado de fragmentação interna, isto é, memória perdida ou subutilizada dentro da área alocada para um processo. No entanto, se o programa não pode ser executado devido à forma como a memória é gerenciada, o problema é chamado de fragmentação externa, isto é, memória perdida fora da área ocupada por um processo. O desperdício de memória em função da fragmentação externa é um grande problema. A origem da fragmentação externa está no fato de cada programa necessitar ocupar uma única área contígua de memória. Se essa necessidade for eliminada, ou seja, se cada programa puder ser espalhado por áreas não contíguas de memória, a fragmentação externa é eliminada. Esse efeito é obtido com a paginação.

Segundo Machado (2007),

> "A memória virtual por paginação é a técnica de gerência de memória onde o espaço de endereçamento real são divididos em blocos de mesmo tamanho chamado páginas. As páginas no espaço virtual são denominadas páginas virtuais, enquanto as páginas no espaço real são chamadas de páginas reais ou frames. Todo mapeamento de endereço virtual real é realizado através de tabela de páginas. Cada processo possui sua tabela de página e cada página possui uma entrada na tabela com as informações que permitem ao sistema localizar a página real correspondente".

O endereço lógico gerado por um processo é inicialmente dividido em duas partes: um número de página lógica e um deslocamento dentro da página. O número da página é usado como índice no acesso à tabela de páginas. Cada entrada da tabela de páginas possui o mapeamento de página lógica para a página física. Já o deslocamento do *byte* dentro da página lógica é carregado exatamente em uma página física. Basta juntar o número de página física obtido na tabela de páginas com o deslocamento já presente no endereço lógico para obter-se o endereço físico do *byte* em questão.

Dessa forma, a paginação é um esquema de gerenciamento de memória em que o espaço de endereçamento físico de um processo não é contíguo. Segundo Silbertchatz (2004),

> "A paginação evita o grande problema de ajustar pedaços de memória de tamanho variável à memória secundária (*backing store*), do qual sofre a maioria dos esquemas de gerenciamento de memória abordados. Quando é preciso desocupar a memória principal é necessário encontrar espaço nesta *backing store*. Os problemas de fragmentação discutidos em conexão com a memória principal são também prevalentes com a *backing store*, exceto o acesso, que é muito mais lento, tornando a compactação impossível. A paginação em suas diversas forma é comumente utilizada na maioria dos sistemas operacionais".

Em computadores com memória virtual o endereço lógico não é alocado diretamente no barramento de memória, para isso é necessário utilizar uma unidade de gerenciamento de memória (MMU). Tanenbaum (2010) define que a MMU tem como objetivo mapear endereços virtuais em endereços físicos. Sabe-se que programas quando são executados fazem referências a endereços virtuais, e é função da MMU mapear esses endereços virtuais para endereços físicos dentro do barramento de memória.

Sendo assim, até o momento pode-se notar que a memória virtual na teoria é um bom recurso utilizado pelo SO, contudo deve ser devidamente implementada para de fato trazer desempenho através de mecanismos que permitem as substituições das páginas da memória principal para o disco e vice versa. Os algoritmos responsáveis pela substituição de páginas na memória física são responsáveis pela velocidade e eficiência no mecanismo de paginação. Alguns exemplos desses algoritmos são elencados em Machado (2007) e Tanenbaum (2010), tais como: FIFO (primeiro a entrar, primeiro a sair), LRU (menos recentemente utilizada), algoritmo Ótimo, NRU (não usada

recentemente), algoritmo de Segunda Chance, algoritmo do Relógio, algoritmo do Conjunto de Trabalho, WSClock, entre outros. Praticamente todos esses algoritmos de paginação funcionam com base em três políticas básicas: uma política de busca, uma política de substituição e uma política de locação.

Com isso, surge a motivação para estabelecer um estudo de caso que realmente nos possibilite obter resultados e que proporcione uma boa análise comparativa de dois algoritmos, nesse caso o FIFO e o LRU. Para isso foram elaborados dois ambientes para realizar uma simulação que compara duas sequências de referências de páginas na memória utilizando dos dois algoritmos. Esta simulação foi realizada no simulador SimulaRSO (http://www.simula-rso.appspot.com) e em uma aplicação de testes (desenvolvida na linguagem C).

3. Objetivos

Elaborar um estudo sobre a paginação de memória e os algoritmos de substituição de páginas FIFO (primeiro a entrar, primeiro a sair) e LRU (usada menos recentemente). Para isso, serão realizadas simulações que possibilitem analisar comparativamente o desempenho de cada algoritmo e principalmente o mecanismo de funcionamento de cada um.

4. Resultados Parciais

Este trabalho faz parte de um estudo parcial da disciplina de Sistemas Operacionais. Inicialmente foi definido o levantamento teórico do tema, com base no que foi visto em aula e, em seguida, um estudo dos algoritmos a serem utilizados na simulação comparativa, além da elaboração dos ambientes de testes. O experimento leva em consideração dois ambientes que permitem observar o desempenho dos algoritmos FIFO e LRU. Para isso, considere um sistema de memória virtual que implementa paginação de memória, onde o limite de *frames* por processo é igual a três. A simulação retornará para as sequências “a” e “b” o número total de *page faults* (falhas de páginas) e *page hits* (acertos) para as estratégias de realocação. Seguem as sequências de referências às páginas pelo processo(s):

a) ***1 / 2 / 3 / 1 / 4 / 2 / 5 / 3 / 4 / 3*** e b) ***1 / 2 / 3 / 1 / 4 / 1 / 3 / 2 / 3 / 3***

Segue abaixo os *resultados para as sequências "a"* e "b":

Referência	Status	Substituição	Memória
1	*page fault*	vazio	1
2	*page fault*	vazio	1 2
3	*page fault*	vazio	1 2 3
1	*page hit*	-	1 2 3
4	*page fault*	1	4 2 3
2	*page hit*	-	4 2 3
5	*page fault*	2	4 5 3
3	*page hit*	-	4 5 3
4	*page hit*	-	4 5 3
3	*page hit*	-	4 5 3

Tabela 1 – FIFO para a sequência de referência "a"
Total de page fault: 5
Total de page hit: 5

Referência	Status	Substituição	Memória
1	*page fault*	vazio	1
2	*page fault*	vazio	1 2
3	*page fault*	vazio	1 2 3
1	*page hit*	-	1 2 3
4	*page fault*	2	1 4 3
2	*page fault*	3	1 4 2
5	*page fault*	1	5 4 2
3	*page fault*	4	5 3 2
4	*page fault*	2	5 3 4
3	*page hit*	-	5 3 4

Tabela 2 – LRU para a sequência de referência "a"
Total de page fault: 8
Total de page hit: 2

Referência	Status	Substituição	Memória
1	*page fault*	vazio	1
2	*page fault*	vazio	1 2
3	*page fault*	vazio	1 2 3
1	*page hit*	-	1 2 3
4	*page fault*	1	4 2 3
1	*page fault*	2	4 1 3
3	*page hit*		4 1 3
2	*page fault*	3	4 1 2
3	*page fault*	4	3 1 2
3	*page hit*	-	3 1 2

Tabela 4 – FIFO para a sequência de referência "b"
Total de page fault: 7
Total de page hit: 3

Referência	Status	Substituição	Memória
1	*page fault*	vazio	1
2	*page fault*	vazio	1 2
3	*page fault*	vazio	1 2 3
1	*page hit*	-	1 2 3
4	*page fault*	2	1 4 3
1	*page hit*	-	1 4 3
3	*page hit*	-	1 4 3
2	*page fault*	4	1 2 3
3	*page hit*	-	1 2 3
3	*page hit*	-	1 2 3

Tabela 5 – LRU para a sequência de referência "b"
Total de page fault: 5
Total de page hit: 5

A simulação das sequências "a" e "b" com os algoritmos FIFO e LRU proporcionaram resultados iguais nas duas aplicações utilizadas, apenas confirmando os resultados obtidos. Em relação ao desempenho comparativo de cada algoritmo, o FIFO obteve um melhor desempenho com a sequência "a", gerando 5 *page faults* e 5 *page hits*. Já o LRU para esta mesma sequência obteve 8 *page faults* e 2 *page hits*. Para a sequência "b", o LRU obteve um melhor desempenho, gerando 5 *page faults* e 5 *page hits*. Já o FIFO para esta mesma sequência obteve 7 *page faults* e 3 *page hits*. Esses resultados estabelecem uma média geral de 4 *page hits* para o FIFO e de 3,5 *page hits* para o LRU. É importante ressaltar que é possível testar outras sequências, no intuito de buscar resultados que apontem diversos resultados de cada algoritmo. O simulador SimulaRSO é mais simples de utilizar, porém permite apenas definir sequências de no máximo dez referências, ao contrário da outra aplicação.

5. Considerações Finais

Sistemas que utilizam paginação são fundamentais para as necessidades atuais de computação, exceto para uso de tecnologia embarcada. A busca por algoritmos eficientes se mantém necessária considerando que ainda há uma relação oposta entre desempenho e custo computacional envolvido.

Compreende-se que a simulação efetuada neste estudo de caso nos permite visualizar melhor os conceitos de gerenciamento de memória. A amostragem do estudo da simulação não permite definir de forma conclusiva qual o melhor algoritmo de substituição de páginas utilizado.

Contudo, estudos revelam o LRU tendo melhor desempenho que o FIFO, e isto só pode ser visto em um ambiente com mais referências à páginas e principalmente com um maior número de *frames*. Portanto, considera-se até o momento que: a escolha no FIFO não leva em consideração se a página está sendo muito utilizada ou não, o que não é muito adequado, pois pode prejudicar o desempenho do sistema. Por este motivo, o FIFO apresenta uma deficiência denominada *anomalia de Belady*: a quantidade de falta de páginas pode aumentar quando o tamanho da memória também aumenta. Por estas razões, o algoritmo FIFO puro é pouco utilizado. A sua principal vantagem é a facilidade de implementação: uma lista de páginas ordenada pela “idade”. Na ocorrência de uma falta de página a primeira página da lista será substituída e a nova será acrescentada ao final da lista.

Já em relação ao LRU, pode-se apontar a sua eficiência principalmente em um ambiente real de multiprogramação. O seu desempenho independe da situação tanto física quanto lógica do sistema, ou seja, independente da quantidade de frames ou páginas que serão referenciadas, na pior situação tem um desempenho próximo ao FIFO. Por vezes, o LRU é definido como um algoritmo que garante eficiência em relação ao problema do aumento de falhas de página quando o tamanho da memória também aumenta (*anomalia de Belady*). A principal desvantagem de um algoritmo LRU é seu custo computacional, pois a cada acesso a memória, é necessário atualizar o tempo de acesso da página e/ou reordenar a lista de páginas, dependendo da implementação.

6. Referências

MACHADO, Francis B. Arquitetura de Sistemas Operacionais. 4 ed. Rio de Janeiro: LTC, 2007.

SILBERTCHATZ, A., GALVIN, P. Fundamentos de Sistemas Operacionais. 6 ed. Rio de Janeiro: LTC, 2004.

TANENBAUM, Andrew S. Sistemas Operacionais Modernos. 3 ed. São Paulo: Pearson Prentice Hall, 2010.