Análise Comparativa de Desempenho de Algoritmos de Substituição de Página em um Simulador de Memória Virtual

Abrahão Picanço Neres de Oliveira[1] **, Andreza Oliveira Gonçalves**[1]

[1] Departamento de Ciência da Computação – Universidade Federal de Roraima (UFRR)
Boa Vista – RR – Brazil

abksneres@gmail.com, andrezaolivego@gmail.com

Abstract. *This paper presents the development and analysis of a virtual memory simulator, created in C language to evaluate the performance of page replacement algorithms. The simulator processes memory access traces to compare three classic policies: FIFO, LRU, and Random. Performance is measured by the number of page faults and dirty page write-backs. The results consistently demonstrate the superiority of the LRU algorithm, which better explores the principle of locality, highlighting the critical impact of the chosen replacement policy on overall system efficiency.*

Resumo. *Este artigo apresenta o desenvolvimento e a análise de um simulador de memória virtual, criado em Linguagem C para avaliar o desempenho de algoritmos de substituição de página. O simulador processa traços de acesso à memória para comparar três políticas clássicas: FIFO, LRU e Random. O desempenho é medido pelo número de page faults e de escritas de páginas sujas. Os resultados demonstram consistentemente a superioridade do algoritmo LRU, que explora melhor o princípio da localidade, evidenciando o impacto crítico da política de substituição na eficiência geral do sistema.*

1. Introdução

O gerenciamento de memória é uma função crítica dos sistemas operacionais modernos. A técnica de memória virtual, implementada via paginação, permite a execução de processos cujas demandas por memória excedem a capacidade da RAM física. Um componente central deste mecanismo é o algoritmo de substituição de página, acionado quando uma página necessária não está na memória (*page fault*) e não há espaço livre para carregá-la. A eficiência deste algoritmo impacta diretamente o desempenho do sistema.

Este trabalho descreve o desenvolvimento de um simulador em C para modelar este ambiente e analisar quantitativamente o desempenho de três algoritmos fundamentais: First-In, First-Out (FIFO), Least Recently Used (LRU) e Aleatório (Random). O simulador processa traços de acesso à memória para coletar estatísticas de *page faults* e escritas de páginas ”sujas” (modificadas), permitindo uma comparação direta da eficiência de cada política.

2. Conceitos Fundamentais

A memória virtual cria para cada processo a ilusão de um espaço de endereçamento contíguo e vasto, dividindo-o em **páginas**. A memória RAM física é dividida em

quadros de página de mesmo tamanho. Uma **tabela de páginas** mapeia as páginas lógicas para os quadros físicos.

Um **page fault** ocorre quando um processo tenta acessar uma página que não está na RAM. O sistema operacional deve, então, buscar a página no armazenamento secundário. Se a RAM estiver cheia, uma página residente deve ser removida. Se esta página foi modificada (marcada como **suja** ou *dirty*), ela precisa ser salva de volta no disco antes da substituição, incorrendo em uma penalidade de desempenho adicional. A escolha da página a ser removida é a responsabilidade do algoritmo de substituição.

3. O Simulador Proposto

O simulador foi desenvolvido em Linguagem C (padrão C11) e compilado com GCC. Ele recebe quatro argumentos via linha de comando: o algoritmo a ser usado (‘fifo‘, ‘lru‘, ‘random‘), o caminho para o arquivo de log, o tamanho da página em KB e o tamanho da memória física em KB.

A memória física é modelada por um vetor dinâmico de estruturas ‘QuadroPagina‘. Cada estrutura armazena o status do quadro (ocupado/livre), o número da página que ele contém, um bit de sujeira (‘suja‘) e um contador de tempo para a implementação do LRU. A conversão de endereço para número de página é feita com uma operação de deslocamento de bits (‘page = addr ¿¿ s‘), onde ‘s‘ é derivado do tamanho da página.

O fluxo de execução processa cada linha do log, determina o número da página, e verifica se ocorre um *hit* (página na memória) ou *miss* (página fora da memória). Em caso de *miss* com a memória cheia, o algoritmo selecionado é invocado para escolher uma página vítima, e as estatísticas são atualizadas.

4. Avaliação Experimental

Para avaliar o simulador, foram realizados testes utilizando os três algoritmos com um arquivo de log (‘simulador.log‘) contendo 1.000.000 de acessos, com uma configuração de 500 KB de memória física e páginas de 10 KB.

Os resultados, sumarizados na Tabela 1, mostram o número de *page faults* (páginas lidas) e de escritas de páginas sujas em disco para cada algoritmo.

Table 1. Resultados dos testes com o arquivo simulador.log

Algoritmo	Page Faults	Páginas Escritas
FIFO	57.690	13.000
LRU	47.057	9.972
Random	56.413	13.851

Conforme os dados, o algoritmo LRU apresentou o melhor desempenho, com o menor número de *page faults* e de escritas em disco. O FIFO teve um desempenho intermediário, enquanto o Random se mostrou ineficiente, com um número de escritas superior até mesmo ao do FIFO. A superioridade do LRU se deve à sua capacidade de explorar a localidade temporal dos acessos, fazendo substituições mais inteligentes.

5. Conclusão

Este trabalho demonstrou com sucesso a implementação de um simulador de memória virtual para a análise de algoritmos de substituição de página. A avaliação experimental confirmou a teoria, evidenciando o desempenho superior do algoritmo LRU em comparação com as políticas FIFO e Random para a carga de trabalho testada. O projeto solidificou a compreensão prática dos complexos mecanismos de gerenciamento de memória, mostrando que a escolha da política de substituição é um fator crítico para a eficiência de um sistema operacional.

References

[1] TANENBAUM, Andrew S.; BOS, Herbert. Modern Operating Systems. 4. ed. Boston: Pearson, 2015.

[2] GEEKSFORGEEKS. Page Replacement Algorithms in Operating Systems. Disponível em: https://www.geeksforgeeks.org/operating-systems/page-replacement-algorithms-in-operating-systems/. Acesso em: 7 ago. 2025.