UNIVERSIDADE DE FORTALEZA
CENTRO DE CIÊNCIAS TECNOLÓGICAS
CURSO: CIÊNCIA DA COMPUTAÇÃO

Readme sobre Simulador de Algoritmos de Substituição de Páginas

Autor 1: Matheus Diógenes Amorim - 2310277
Autor 2: Luiz Vitor Dantas Freitas - 2320410

Link do github: https://github.com/MatheuzinDev/Simulador-Algoritmos-de-Paginacao

Palavras-chave: Memória virtual. Algoritmos de substituição. Faltas de página. Simulação de algoritmos. Desempenho de sistemas.

Resumo

A gerência de memória virtual é um componente fundamental dos sistemas operacionais modernos, permitindo a execução de processos que excedem o tamanho da memória física disponível. Este processo depende de técnicas de paginação e, crucialmente, de algoritmos de substituição de página para decidir qual página residente na memória deve ser removida quando ocorre uma *falta de página* (page fault). Este projeto apresenta a implementação e a simulação de quatro algoritmos de substituição de página: FIFO (First-In, First-Out), LRU (Least Recently Used), Clock (Relógio) e NFU (Not Frequently Used).

Desenvolvido em Java, o simulador recebe como entrada uma sequência de referências de páginas e o número de quadros de memória disponíveis. O objetivo é comparar o desempenho de cada algoritmo, medido pelo número total de faltas de página, oferecendo uma análise prática de suas eficiências e complexidades.

Introdução

Em sistemas computacionais, a memória principal (RAM) é um recurso limitado e de alto custo. Para contornar essa limitação, os sistemas operacionais implementam o conceito de memória virtual, que utiliza espaço em disco (swap) para estender a capacidade da RAM. Quando um processo tenta acessar uma página que não está na memória principal, ocorre uma interrupção de hardware chamada falta de página. O sistema operacional deve, então, buscar a página no disco e carregá-la na memória. Se a memória já estiver cheia, o sistema precisa escolher uma página para ser removida e dar lugar à nova.

A escolha da página a ser retirada é realizada por um algoritmo de substituição de página. A eficiência deste algoritmo é crítica, pois um grande número de faltas de página pode degradar severamente o desempenho do sistema, um fenômeno conhecido como *thrashing*.

Este trabalho foca na implementação de quatro desses algoritmos para fins de simulação e comparação. O objetivo é demonstrar, na prática, como diferentes estratégias impactam diretamente a contagem de faltas de página para uma mesma carga de trabalho.

Agora, explicaremos como rodar o projeto pelo terminal, lembrando que é possível clicar no ícone de “Play” no Intellij, ou “Run Without Debugging” no VSCODE.

Antes de compilar e executar, entre na pasta src onde estão os arquivos .java:

Comando no terminal:

- cd .\src\

A classe Main.java é o ponto de entrada para a execução via terminal.
Para rodar:

1. Compile o projeto:

- javac Main.java FIFO.java LRU.java Clock.java NFU.java

2. Execute:

- java Main

A interface gráfica foi implementada na classe SimuladorSubstituicao.java, utilizando a biblioteca Java Swing.

1. Compile a interface:

- javac SimuladorSubstituicao.java

2. Execute:

- java SimuladorSubstituicao

Metodologia

O projeto foi desenvolvido na linguagem de programação Java e é composto por cinco classes principais:

- Main.java: Serve como o ponto de entrada da simulação. Esta classe utiliza a biblioteca Scanner para coletar duas entradas do usuário: a sequência de referências de página (uma string de números separados por espaço) e o número total de quadros de memória

(um inteiro). Em seguida, invoca o método simular de cada um dos seis algoritmos, imprimindo o resultado (número de faltas de página) de cada um.

- FIFO.java: Implementa o algoritmo First-In, First-Out. Utiliza uma lista encadeada para gerenciar as páginas na memória. Quando uma falta de página ocorre e a memória está cheia, a página na frente da fila (a mais antiga) é removida.

- LRU.java: Implementa o algoritmo Least Recently Used. Utiliza um LinkedHashSet, uma estrutura de dados que mantém a ordem de inserção. Como o nome diz, ele pega a página mais recentemente usada, que já está na memória, e a coloca no final da fila, assim, a primeira página da fila é a menos recentemente usada, que será removida no caso de uma falta de página.

- Clock.java: Implementa o algoritmo Relógio (ou Segunda Chance). Utiliza um array para as páginas e um array booleano paralelo para simular o "bit de uso". Um ponteiro circular percorre os quadros. Ao procurar uma página para remover, se o bit de uso for true, ele é alterado para false, dando uma segunda chance para a página; se for false, a página é substituída.

- NFU.java: Implementa o algoritmo Not Frequently Used. Utiliza um Map para manter um contador de acessos para cada página. Em uma falta, a página presente na memória com o menor valor no contador de frequência é escolhida para ser substituída.

- Desenvolvimento Front-End: Além da versão executada via terminal, foi implementada uma interface gráfica utilizando a biblioteca Java Swing, com o propósito de proporcionar uma experiência mais intuitiva e visual ao usuário. Essa interface foi estruturada por meio da classe SimuladorSubstituicao, que organiza os componentes principais do sistema em um layout dividido entre entrada de dados, exibição textual dos resultados e um painel gráfico comparativo.

Resultados e Discussão

A execução do programa Main fornece uma saída comparativa direta do número de faltas de página para cada algoritmo, dado um conjunto de entradas. Por exemplo, para uma sequência de teste e um número de quadros, o programa exibe qual algoritmo foi mais eficiente. Os resultados dos algoritmos foram os seguintes:

- FIFO: É o mais simples de implementar, mas geralmente apresenta o pior desempenho, pois pode remover páginas frequentemente usadas que apenas tiveram a má sorte de chegar cedo.

- LRU: Geralmente tem um desempenho excelente, muito próximo do Ótimo. Sua lógica baseia-se no princípio da localidade temporal (se uma página foi usada recentemente, é provável que seja usada novamente). Sua implementação, no entanto, pode ser custosa, exigindo atualização da estrutura de dados a cada acesso.

- Clock: Apresenta um desempenho muito bom, geralmente melhor que o FIFO e ligeiramente inferior ao LRU. É uma implementação de "compromisso", oferecendo uma boa aproximação do LRU com uma sobrecarga de sistema muito menor, sendo amplamente utilizado na prática.

- NFU: É uma aproximação simples do LRU, mas seu contador não "esquece" o histórico. Uma página muito usada no passado, mas não mais relevante, pode parecer mais importante do que uma página usada poucas vezes, mas recentemente, o que é uma falha em sua lógica.

Conclusão

Este projeto demonstrou com sucesso a implementação de quatro algoritmos fundamentais de substituição de página. Através da ferramenta de simulação desenvolvida em Java, foi possível analisar e comparar empiricamente o desempenho de cada estratégia.

Conclui-se que a escolha do algoritmo tem um impacto direto e significativo na eficiência da gerência de memória virtual, medido pela contagem de faltas de página. Algoritmos simples como o FIFO, embora fáceis de implementar, são ineficientes. O LRU oferece um desempenho quase ótimo, mas a um custo de implementação elevado. Finalmente, algoritmos de aproximação como o Clock apresentam um desempenho próximo do LRU com uma complexidade de implementação e sobrecarga de sistema consideravelmente menores, justificando sua popularidade em sistemas operacionais reais.

Referências

JOSHI, S.; GOSWAMI, P.; KUMAR, R. *Comparative Study of Page Replacement Algorithms in Operating System*. International Journal of Innovative Technology and Exploring Engineering, v. 9, n. 10, p. 1-7, 2020. Disponível em: https://www.ijitee.org/wp-content/uploads/papers/v9i10/J75530891020.pdf. Acesso em: 31 out. 2025.

WIKIPEDIA. *Page replacement algorithm*. Wikipédia, a enciclopédia livre. Disponível em: https://en.wikipedia.org/wiki/Page_replacement_algorithm. Acesso em: 31 out. 2025.

INSTITUTO DE CIÊNCIAS MATEMÁTICAS E DE COMPUTAÇÃO. *Aula 12 – Algoritmos de Substituição de Páginas*. São Carlos: ICMC-USP, s.d. Disponível em: http://wiki.icmc.usp.br/images/d/dc/Aula12.pdf. Acesso em: 31 out. 2025.