Relatório: Utilização do Algoritmo LRU no Gerenciador de Memória Aleatório

Alunos: Francisco Adrian Vinicius Chaves Sampaio, Gabriel De Menezes Sousa, Gustavo Kesley de Fontes Nunes, Jackson Renan Oliveira dos Santos, Francisco Deivid da Silva, Bruno Gomes da Silveira.

1. Introdução

O gerenciamento de memória virtual é um dos aspectos fundamentais dos sistemas operacionais modernos, permitindo a execução eficiente de múltiplos processos sem a necessidade de armazenar todo o conteúdo na memória RAM. Para otimizar a alocação da memória, são utilizados algoritmos de substituição de páginas, que determinam quais páginas devem ser removidas quando uma nova precisa ser carregada na RAM.

Neste projeto, foi implementado o algoritmo LRU (Least Recently Used) para gerenciar o processo de Page In e referência de páginas em um ambiente simulado de memória, utilizando Java.

2. Objetivo do Algoritmo LRU

O algoritmo LRU busca substituir a página menos recentemente utilizada quando a RAM atinge sua capacidade máxima. A premissa é que páginas acessadas recentemente provavelmente serão acessadas novamente em um curto período de tempo, enquanto páginas não utilizadas há muito tempo são menos prováveis de serem necessárias no futuro.

A implementação do LRU permite que o gerenciador de memória:

- **Armazene as páginas em uma memória RAM limitada.**
- **Realize substituições eficientes de páginas da RAM quando necessário.**
- **Priorize páginas que foram recentemente acessadas, evitando a remoção de páginas úteis.**
- **Simule um sistema de gerenciamento de memória similar ao usado em sistemas reais.**

---

3. Funcionamento do Sistema

O gerenciador de memória é dividido em dois espaços de armazenamento:

1. **RAM (Memória Principal) - Armazena páginas atualmente utilizadas pelos processos.**
2. **HD (Memória Secundária) - Contém páginas que não estão na RAM no momento.**

O sistema possui um menu interativo com três operações principais:

1. **Page In (Carregar uma página do HD para a RAM):**
    - **Se a RAM estiver cheia, a página menos recentemente utilizada (LRU) será removida e substituída pela nova página carregada do HD.**
    - **A página escolhida para remoção é a que possui o tempo de referência mais antigo.**
2. **Referenciar uma página na RAM:**
    - **Atualiza o tempo de acesso da página selecionada.**
    - **Essa ação mantém a página na RAM por mais tempo, evitando que seja substituída rapidamente.**
3. **Visualizar o estado atual da RAM e do HD:**
    - **Exibe as páginas atualmente na RAM e no HD, organizadas por índice.**

---

4. Passos da Utilização do Algoritmo LRU

4.1. Processo de Substituição de Página (Page In)

1. **O usuário seleciona um índice do HD que deseja carregar na RAM.**
2. **Se houver um espaço livre na RAM, a nova página é armazenada sem necessidade de substituição.**
3. **Se a RAM estiver cheia:**
    - **O sistema encontra a página com o menor tempo de acesso (página menos recentemente usada).**
    - **A página menos utilizada é removida e enviada de volta ao HD.**
    - **A nova página do HD substitui a página removida na RAM.**
    - **O tempo da nova página é atualizado para garantir que ela não seja removida imediatamente.**

4.2. Referência de Página

1. **O usuário escolhe um índice da RAM para "referenciar".**
2. **O sistema atualiza o tempo de acesso da página referenciada.**
3. **Com isso, essa página será mantida na RAM por mais tempo, pois não será considerada "menos utilizada".**

4.3. Exibição do Estado Atual da Memória

1. **O sistema exibe um mapa atualizado da RAM e do HD.**
2. **O usuário pode visualizar quais páginas estão armazenadas e sua organização.**

---

5. Benefícios da Implementação do LRU

- **Eficiência: Prioriza páginas que têm maior probabilidade de serem acessadas novamente, reduzindo a necessidade de carregamento constante de novas páginas.**
- **Aproximação do Comportamento Real: O LRU é utilizado em sistemas operacionais reais para gerenciar memória virtual.**
- **Gerenciamento Automático: O usuário não precisa decidir manualmente quais páginas remover, pois o algoritmo LRU faz essa escolha com base no histórico de acessos.**

6. Exemplo de Uso

Situação Inicial

RAM (Memória Principal)	HD (Memória Secundária)
PARTE 3 DE A	PARTE 1 DE A
PARTE 1 DE C	PARTE 2 DE A
PARTE 2 DE C	PARTE 1 DE B
PARTE 3 DE C	PARTE 2 DE B

Exemplo de Page In

1. **O usuário escolhe carregar a página PARTE 1 DE B da HD para a RAM.**
2. **Como a RAM está cheia, o algoritmo LRU escolhe a página menos recentemente usada para remoção.**
3. **Suponha que a página PARTE 3 DE A tenha sido a menos utilizada recentemente.**
4. **Substituição:**
    - **RAM antes: [PARTE 3 DE A, PARTE 1 DE C, PARTE 2 DE C, PARTE 3 DE C]**
    - **RAM depois: [PARTE 1 DE B, PARTE 1 DE C, PARTE 2 DE C, PARTE 3 DE C]**
    - **HD atualizado: [PARTE 3 DE A, PARTE 2 DE A, PARTE 1 DE B, PARTE 2 DE B]**

Agora, a página PARTE 1 DE B foi carregada para a RAM com sucesso.

---

7. Conclusão

O algoritmo Least Recently Used (LRU) provou ser eficiente na gestão de memória ao garantir que páginas recentemente acessadas permaneçam na RAM e que as menos utilizadas sejam substituídas. Ele é uma excelente estratégia para sistemas que precisam equilibrar o uso de memória limitada sem impactar drasticamente o desempenho.

Este projeto serve como uma ótima simulação educacional do gerenciamento de memória em sistemas operacionais modernos, demonstrando de forma prática como a substituição de páginas funciona.

Se implementado em um sistema real, o LRU poderia ser melhorado com estruturas mais eficientes, como listas duplamente encadeadas e tabelas hash, para otimizar a busca pela página menos recentemente utilizada.

---

8. Possíveis Melhorias

- **Uso de Estruturas Otimizadas: Implementação de uma lista duplamente encadeada para acelerar a remoção de páginas.**
- **Uso de HashMap para Acesso Direto: Em vez de percorrer toda a RAM, um HashMap<Page, Tempo> poderia armazenar tempos de acesso.**
- **Algoritmo Alternativo: Implementação de outro algoritmo como FIFO (First In, First Out) para comparação.**

Resumo:

Critério	Algoritmo LRU
Estratégia	Remove a página **menos recentemente usada**
Tempo de Execução	Requer busca pela página com menor tempo de acesso
Vantagem	Minimiza substituições desnecessárias
Desvantagem	Pode ter alto custo computacional se não for implementado com estruturas otimizadas