

Relatório Técnico: Simulação de Algoritmos de Substituição de Páginas

Universidade Federal do Ceará (UFC) - Sistemas Operacionais

Equipe: Antônio Gabriel, Erik Bayerlein, Julia Naomi

1. Introdução e Objetivo

Este relatório apresenta a implementação e análise de um simulador de gerência de memória focado em algoritmos de substituição de páginas. O objetivo principal foi calcular métricas de desempenho (faltas de página, taxa de faltas e evicções) e observar o comportamento de diferentes estratégias sob variações na quantidade de molduras (frames) de memória física.

O projeto permitiu a análise experimental da **Anomalia de Belady** (especialmente no FIFO) e do efeito da **localidade de referência** em algoritmos baseados em recência como o LRU.

Divisão:

Gabriel: Configuração inicial do Projeto, implementação do MemorySimulator, algoritmos Ótimo, LRU.

Erik: Implementação módulo de leitura do arquivo de rastreamento, cálculo e coleta das métricas, lógica de simulação, algoritmos NRU, LFU, MFU e SecondChance (Clock).

Julia: Implementação da saída, tratamento de erro, algoritmo FIFO e docs.

2. Decisões de Implementação e Arquitetura

2.1 Estruturas de Dados e Classes

- **Classe Page:** Representa a unidade fundamental de memória. Possui um `id` (identificador único) conforme sugerido e bits de controle (`modified`, `referenced`) para suporte a algoritmos como NRU e Segunda Chance.
- **Classe PageFrame:** Representa as molduras físicas na RAM. Contém metadados para controle de substituição:
 - `loadTime`: Para identificar a ordem de chegada.
 - `lastAccessTime`: Usado para determinar a recência.
 - `frequency`: Contador de acessos.
 - `referenceBit` (bit R),
 - `modifyBit` (bit M)
- **MemorySimulator:** O "core" do sistema, responsável por iterar sobre a cadeia de referências e contabilizar os resultados.
- **TraceReader:** Componente de I/O que valida e lê o arquivo `.trace`, garantindo que o simulador processe uma referência por linha.

2.2 Padrões de Projeto (Design Patterns)

Para garantir a extensibilidade exigida (suporte a múltiplos algoritmos como FIFO, LRU, Ótimo, etc.), aplicamos:

- **Strategy Pattern:** Cada algoritmo foi implementado como uma classe independente que herda de uma interface comum. Isso isola a lógica de decisão de "qual página remover" do restante do simulador.
- **Factory Method:** Uma fábrica de algoritmos que instancia a estratégia correta baseada no argumento `--algo` passado via linha de comando.

3. Algoritmos Implementados

1. **FIFO:** Substitui a página que entrou primeiro (maior tempo em memória).
2. **LRU:** Substitui a página não utilizada há mais tempo (menor `lastAccessTime`).
3. **Ótimo:** Analisa o futuro da cadeia de referências e remove a página que demorará mais tempo a ser usada.
4. **Segunda Chance / Clock:** Evolução do FIFO que usa o `referenceBit`. Se o bit for 1, a página ganha uma "segunda chance" e o bit é zerado.
5. **NRU (Not Recently Used):** Classifica as páginas em 4 classes baseadas nos bits R e M, removendo aleatoriamente uma página da classe de menor prioridade.
6. **LFU / MFU:** Algoritmos baseados em frequência de uso (Menos/Mais Frequentemente Usada).

4. Metodologia de Teste

A validação foi realizada utilizando a **Cadeia de Referências Oficial** de Silberschatz et al. (2001):

7, 0, 1, 2, 0, 3, 0, 4, 2, 3, 0, 3, 2, 1, 2, 0, 1, 7, 0, 1

O simulador processou este traço considerando dois cenários de memória física: **3 molduras** e **4 molduras**.

4.1 Resultados obtidos

== Simulation Results ==

Algorithm: FIFO
 Frames: 3
 References: 20
 Page Faults: 15
 Page Fault Rate: 75.00%
 Page Hits: 5
 Page Hit Rate: 25.00%
 Replacements: 12
 Final set:
 frame_ids: 0 1 2
 page_ids: 7 0 1

== Simulation Results ==

Algorithm: FIFO
 Frames: 4
 References: 20
 Page Faults: 10
 Page Fault Rate: 50.00%
 Page Hits: 10
 Page Hit Rate: 50.00%
 Replacements: 6
 Final set:
 frame_ids: 0 1 2 3
 page_ids: 2 7 0 1

== Simulation Results ==

Algorithm: LRU
 Frames: 3
 References: 20
 Page Faults: 12
 Page Fault Rate: 60.00%
 Page Hits: 8
 Page Hit Rate: 40.00%
 Replacements: 9
 Final set:
 frame_ids: 0 1 2
 page_ids: 1 0 7

== Simulation Results ==

Algorithm: LRU
 Frames: 4
 References: 20
 Page Faults: 8
 Page Fault Rate: 40.00%
 Page Hits: 12
 Page Hit Rate: 60.00%
 Replacements: 4
 Final set:
 frame_ids: 0 1 2 3
 page_ids: 7 0 1 2

== Simulation Results ==

Algorithm: OPT
 Frames: 3
 References: 20
 Page Faults: 9
 Page Fault Rate: 45.00%
 Page Hits: 11
 Page Hit Rate: 55.00%
 Replacements: 6
 Final set:
 frame_ids: 0 1 2
 page_ids: 7 0 1

== Simulation Results ==

Algorithm: OPT
 Frames: 4
 References: 20
 Page Faults: 8
 Page Fault Rate: 40.00%
 Page Hits: 12
 Page Hit Rate: 60.00%
 Replacements: 4
 Final set:
 frame_ids: 0 1 2 3
 page_ids: 7 0 1 2

== Simulation Results ==

Algorithm: CLOCK
 Frames: 3
 References: 20
 Page Faults: 16
 Page Fault Rate: 80.00%
 Page Hits: 4
 Page Hit Rate: 20.00%
 Replacements: 13
 Final set:
 frame_ids: 0 1 2
 page_ids: 0 1 3

== Simulation Results ==

Algorithm: CLOCK
 Frames: 4
 References: 20
 Page Faults: 13
 Page Fault Rate: 65.00%
 Page Hits: 7
 Page Hit Rate: 35.00%
 Replacements: 9
 Final set:
 frame_ids: 0 1 2 3
 page_ids: 3 1 0 7

== Simulation Results ==

Algorithm: NRU
 Frames: 3
 References: 20
 Page Faults: 10
 Page Fault Rate: 50.00%
 Page Hits: 10
 Page Hit Rate: 50.00%
 Replacements: 7
 Final set:
 frame_ids: 0 1 2
 page_ids: 1 0 2

== Simulation Results ==

Algorithm: NRU
 Frames: 4
 References: 20
 Page Faults: 9
 Page Fault Rate: 45.00%
 Page Hits: 11
 Page Hit Rate: 55.00%
 Replacements: 5
 Final set:
 frame_ids: 0 1 2 3
 page_ids: 1 0 2 7

<pre>== Simulation Results == Algorithm: LFU Frames: 4 References: 20 Page Faults: 8 Page Fault Rate: 40.00% Page Hits: 12 Page Hit Rate: 60.00% Replacements: 4 Final set: frame_ids: 0 1 2 3 page_ids: 7 0 1 2</pre>	<pre>== Simulation Results == Algorithm: LFU Frames: 3 References: 20 Page Faults: 14 Page Fault Rate: 70.00% Page Hits: 6 Page Hit Rate: 30.00% Replacements: 11 Final set: frame_ids: 0 1 2 page_ids: 3 0 1</pre>
<pre>== Simulation Results == Algorithm: MFU Frames: 3 References: 20 Page Faults: 13 Page Fault Rate: 65.00% Page Hits: 7 Page Hit Rate: 35.00% Replacements: 10 Final set: frame_ids: 0 1 2 page_ids: 2 1 7</pre>	<pre>== Simulation Results == Algorithm: MFU Frames: 4 References: 20 Page Faults: 12 Page Fault Rate: 60.00% Page Hits: 8 Page Hit Rate: 40.00% Replacements: 8 Final set: frame_ids: 0 1 2 3 page_ids: 2 1 7 3</pre>
<pre>== Simulation Results == Algorithm: NRU Frames: 4 References: 20 Page Faults: 9 Page Fault Rate: 45.00% Page Hits: 11 Page Hit Rate: 55.00% Replacements: 5 Final set: frame_ids: 0 1 2 3 page_ids: 7 0 2 1</pre>	<pre>== Simulation Results == Algorithm: NRU Frames: 3 References: 20 Page Faults: 12 Page Fault Rate: 60.00% Page Hits: 8 Page Hit Rate: 40.00% Replacements: 9 Final set: frame_ids: 0 1 2 page_ids: 7 0 1</pre>

Tabela comparativa com número de Page Faults e Page Fault Rate:

Molduras	FIFO	LRU	Ótimo	Segunda Chance/Clock	NRU*	LFU	MFU
3 Frames	15 (75%)	12 (60%)	9 (45%)	13 (65%)	12 (60%)	14(70%)	13(65 %)
4 Frames	10 (50%)	8 (40%)	8 (40%)	16 (80%)	9 (45%)	8(40%)	12 (60%)

*Considerando o teste mais recente de NRU

Como o NRU depende dos bits **R** (Referência) e **M** (Modificação), o simulador decide aleatoriamente se houve escrita. Isso altera a categoria da página e, consequentemente, a prioridade de substituição a cada execução.

5. Conclusão da Validação

A implementação do algoritmo **Ótimo** forneceu o parâmetro ideal de desempenho, enquanto o **LRU** provou ser uma aproximação prática superior ao **FIFO** em cenários de baixa memória.

Taxa de Faltas: O efeito de localidade é o que permite que a taxa de faltas baixe conforme aumentamos as molduras (frames).

Evições: Algoritmos que respeitam a localidade minimizam o número de evições, pois mantêm as páginas ativas na memória física por mais tempo.

Validação da Localidade de Referência: O algoritmo **LRU** consolidou-se como a melhor política prática, alcançando com 4 frames a mesma eficiência do **Ótimo (8 faltas)**. Isto prova que a recência de uso é um previsor altamente fiável para o comportamento do processo.

Superioridade sobre o FIFO: O **FIFO** apresentou o pior desempenho em todos os cenários (até 75% de taxa de falta), evidenciando que ignorar a frequência e a recência de acesso torna a gestão de memória inefficiente.

Viabilidade do Custo-Benefício: Algoritmos baseados em bits de controle, como **Clock** e **NRU**, entregaram resultados superiores ao FIFO com um custo computacional significativamente menor que o LRU, posicionando-se como as soluções mais equilibradas para implementação em sistemas reais.

Impacto Direto da Memória: O aumento de frames reduziu drasticamente as faltas de página em todos os algoritmos, confirmando que a expansão da memória física é o fator de maior impacto na mitigação do *overhead* de paginação e na prevenção do *thrashing*.

5.1 Discussão: Anomalia de Belady

A Anomalia de Belady é um fenômeno onde, ao aumentar o número de molduras de página (frames), o número de faltas de página também aumenta. Este comportamento é exclusivo de algoritmos que não são do tipo "pilha", como o FIFO.

Com base nos testes realizados:

- **Observação:** No traço de Silberschatz, ao passar de **3 para 4 molduras**, o FIFO reduziu as faltas de **15 para 10**.
- **Conclusão Técnica:** Portanto, nesta sequência de referências específica, a Anomalia de Belady **não ocorreu**. O simulador demonstrou que o sistema se comportou de forma convencional, onde mais memória física resultou em melhor desempenho.