Relatório de Sistemas Operacionais: Laboratório de Memória Virtual

Bruno Costa Souto[1], Felipe Augusto Araújo da Cunha[1]

[1]Universidade Federal do Rio Grande do Norte (UFRN)

[2]Centro de Ensino Superior do Seridó (CERES).

Professor João Batista Borges Neto

Resumo. *Este trabalho tem como objetivo reforçar os conceitos teóricos de gerência de memória em sistemas operacionais, com foco nos mecanismos de memória virtual, especialmente nos eventos de page fault e page written. Foram implementados e analisados seis algoritmos de substituição de páginas: FIFO, LRU, Second Chance (SC), Random, LFU e MFU. Os testes foram realizados por meio da simulação da execução de quatro arquivos de trace, variando a quantidade de frames disponíveis (2 a 64). Os dados coletados permitiram uma análise comparativa entre os algoritmos quanto à eficiência na redução de page faults e pages written, possibilitando uma avaliação prática do comportamento de cada estratégia sob diferentes cenários.*

Introdução

Esse trabalho tem como objetivo reforçar os conhecimentos teóricos abordados na disciplina de Sistemas Operacionais sobre à gerência de memória em sistemas modernos. Especificamente, sobre o funcionamento da memória virtual, com destaque para a aplicação de diferentes algoritmos de substituição de páginas, para observar quantidade de *page faults* e *page written*.

Para isso, utilizou-se como base um simulador de memória virtual fornecido pelo professor, no qual foi possível implementar os seguintes algoritmos de substituição: FIFO (*First-In First-Out*), LRU *(Least Recently Used*), *Second Chance* (SC), LFU (*Least Frequently Used*) e MFU (*Most Frequently Used*). O simulador já possuía um algoritmo aleatório *(random*) como referência.

Os testes foram realizados com os arquivos de traços (traces) fornecidos no repositório, que contém sequências de acessos à memória, nas formas de leitura (R) e escrita (W). Cada algoritmo foi executado com diferentes quantidades de frames disponíveis (2, 4, 8, 16, 32 e 64), para assim, avaliar o impacto da alocação de memória no desempenho dos algoritmos.

Nesta análise, dois principais indicadores foram utilizados: o número de *page faults*, que indica quantas vezes o sistema precisou buscar páginas ausentes na RAM, e o número de *pages written*, que mostra quantas páginas modificadas precisaram ser gravadas de volta no disco. A seguir, apresentamos os resultados obtidos com base nesses critérios.

Sumário

1. Análise Comparativa

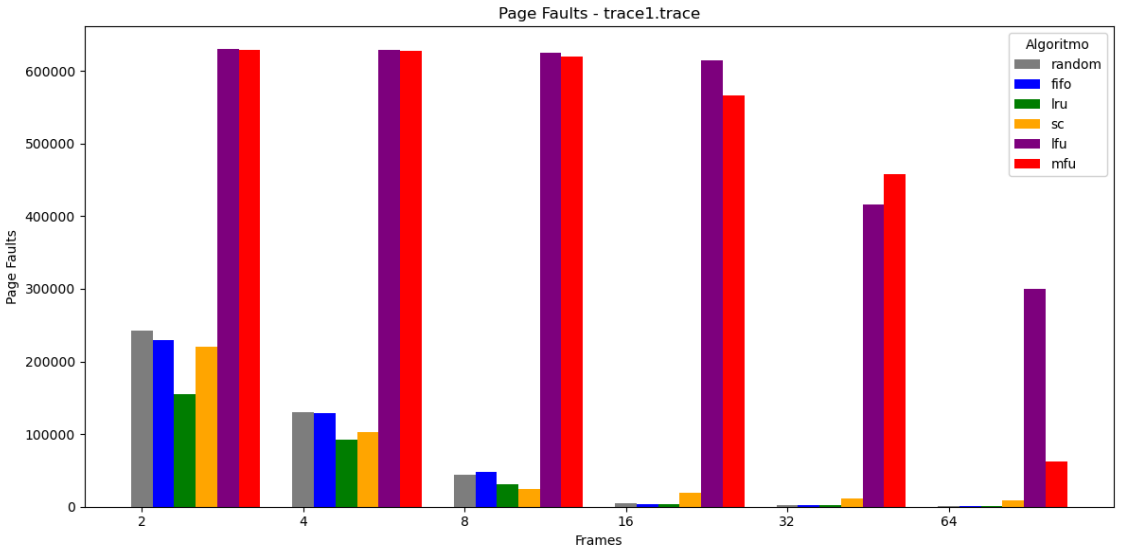
1.1. Page Faults

A análise dos gráficos a seguir revela o comportamento de cada algoritmo de substituição em relação ao número de faltas de página (page faults) para os diferentes arquivos de trace e variações no número de frames. Como esperado, à medida que o número de frames aumenta, o número de faltas de página diminui, mas a taxa de melhoria varia bastante entre os algoritmos.

1.1.1. Trace 1

No primeiro trace, observa-se que os algoritmos LFU e MFU apresentaram desempenho significativamente inferior aos demais, com um número de *page faults* extremamente alto, mesmo com o aumento do número de frames. O MFU, em particular, manteve uma taxa de faltas muito elevada em todos os cenários, junto com o LFU, embora tenha obtido um desempenho relativamente melhor que o LFU ao utilizar 64 frames, o que indica que o trace 1 favorece seu uso apenas nesse contexto mais amplo de alocação de memória.

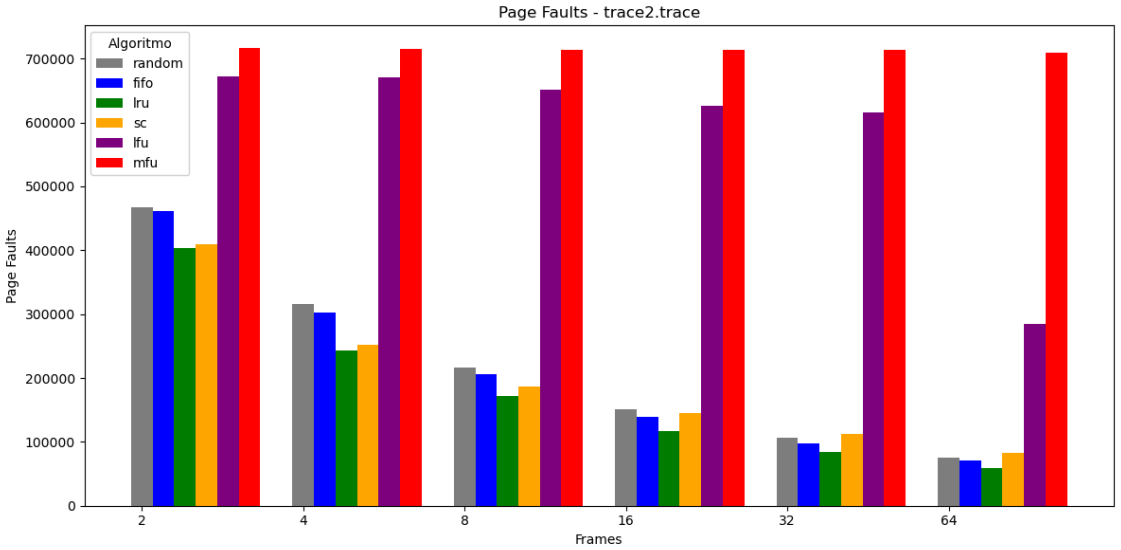
Por outro lado, algoritmos como LRU, SC e FIFO se mostraram mais eficientes. O LRU apresentou o melhor desempenho geral neste trace. Já o *Random*, devido à sua natureza não determinística, teve desempenho inferior ao LRU, SC e FIFO na maioria dos casos, o que era esperado dado que não utilizava nenhuma política de escolha de páginas, mas que ainda sim teve resultados até melhores que o FIFO em alguns casos.



1.1.2. Trace 2

O segundo trace reforça a tendência de má performance dos algoritmos MFU e LFU, que novamente se destacam como os piores em termos de número de *page faults*, mantendo as maiores taxas em todos os valores de frames. Desta vez, o MFU apresentou um desempenho consistentemente ruim, com números extremamente altos em todos os cenários. O LFU também manteve um desempenho insatisfatório, embora apresente uma leve melhora com o aumento da quantidade de frames.

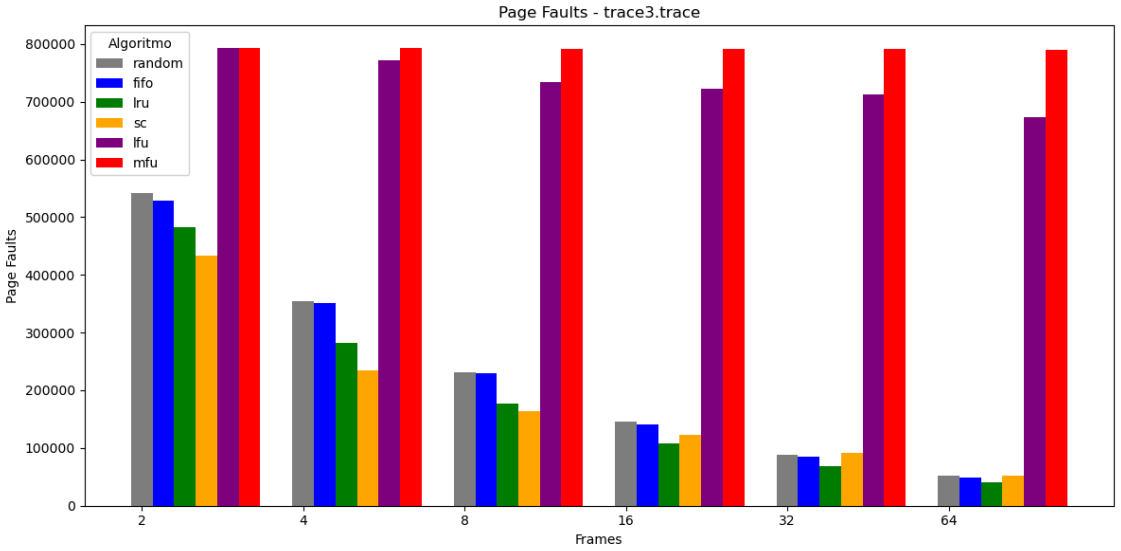
Os algoritmos LRU, SC e FIFO continuaram demonstrando desempenho estável, com reduções progressivas no número de *page faults* à medida que o número de frames aumenta, assim como observado no Trace 1. No entanto, os valores em geral foram um pouco mais altos neste trace. Já o algoritmo Random novamente ficou atrás de LRU, SC e FIFO, apresentando um comportamento intermediário, mas não ruim pois se assemelhou bastante ao FIFO.



1.1.3. Trace 3

Este trace apresenta características muito semelhantes às do Trace 2. Nessa situação, os algoritmos LRU, SC e FIFO continuam com bom desempenho, mantendo-se entre os que apresentaram menor número de *page faults*. Destaque para o SC, que chegou a superar o LRU em alguns casos, demonstrando sua eficácia em certos padrões de acesso.

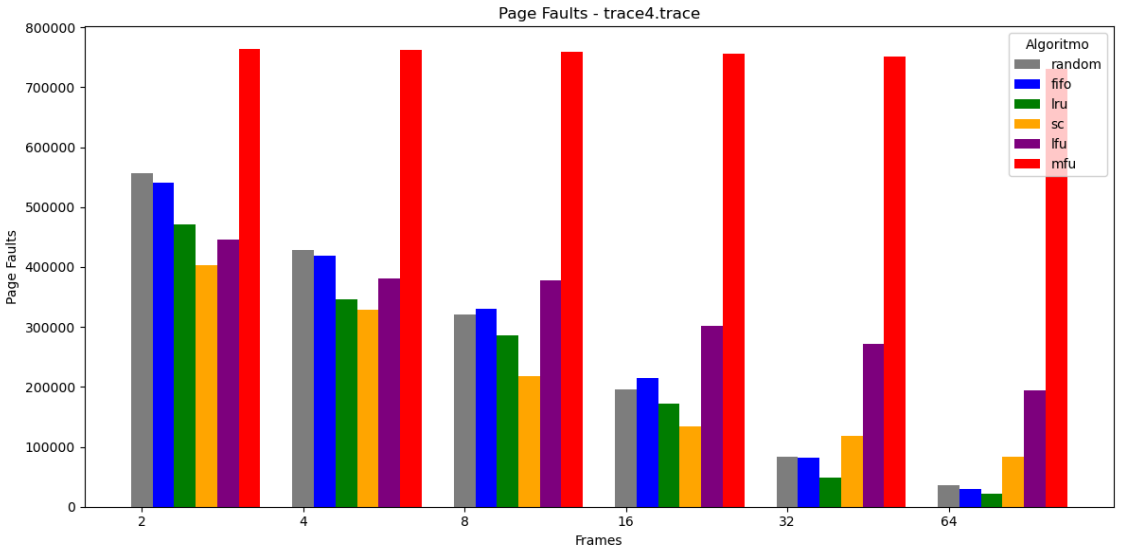
Por outro lado, os algoritmos LFU e MFU seguem com altos números, mesmo com o aumento da quantidade de frames. Esse comportamento reforça a limitação dos algoritmos baseados exclusivamente em frequência de acesso. E o Random seguiu com resultados parecidos ao do FIFO.



1.1.4 Trace 4

O quarto trace revela um dado bastante relevante em comparação com os demais traces analisados até agora. Os algoritmos LRU, SC e FIFO mantêm um desempenho consistente, com baixos índices de *page faults*, assim como nos traces anteriores. O MFU, por sua vez, continua apresentando desempenho significativamente inferior.

No entanto, chama atenção o comportamento do LFU, que se mostrou eficaz em alguns cenários, especialmente nas configurações com 2 e 4 frames, chegando a superar os algoritmos LRU, FIFO e Random em número de *page faults*. Esse resultado indica que, dependendo do trace, os resultados obtidos podem ser bastante diferentes.



1.2. Page Written

Outro ponto importante na avaliação do desempenho dos algoritmos de substituição de páginas é a quantidade de páginas escritas em disco, *pages written*. Esse indicador mostra quantas páginas modificadas (dirty pages) precisaram ser gravadas de volta no armazenamento durante o processo de substituição. Um alto número de *pages written* pode indicar não apenas maior uso do sistema de armazenamento.

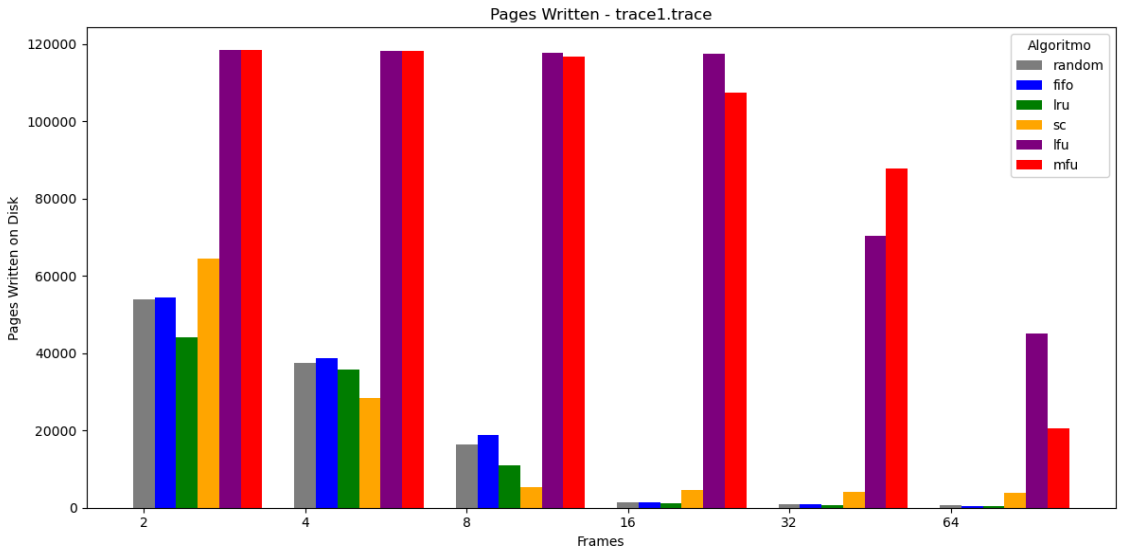
A eficiência de um algoritmo não está apenas em reduzir *page faults*, mas também em evitar substituições frequentes de páginas que sofreram alterações. Assim, algoritmos que mantêm páginas sujas por mais tempo antes de substituí-las tendem a apresentar melhor desempenho nesse aspecto.

Nos tópicos a seguir, são analisados os resultados de *pages written* para cada um dos quatro traces utilizados, considerando as diferentes quantidades de frames.

1.2.1. Trace 1

No primeiro trace, é possível observar uma diferença significativa no comportamento dos algoritmos quanto à quantidade de páginas escritas em disco. Os algoritmos LFU e MFU novamente se destacam negativamente, assim como já havia sido observado nos valores de *page faults*, mantendo valores extremamente altos de *pages written* em praticamente todas as configurações de frames.

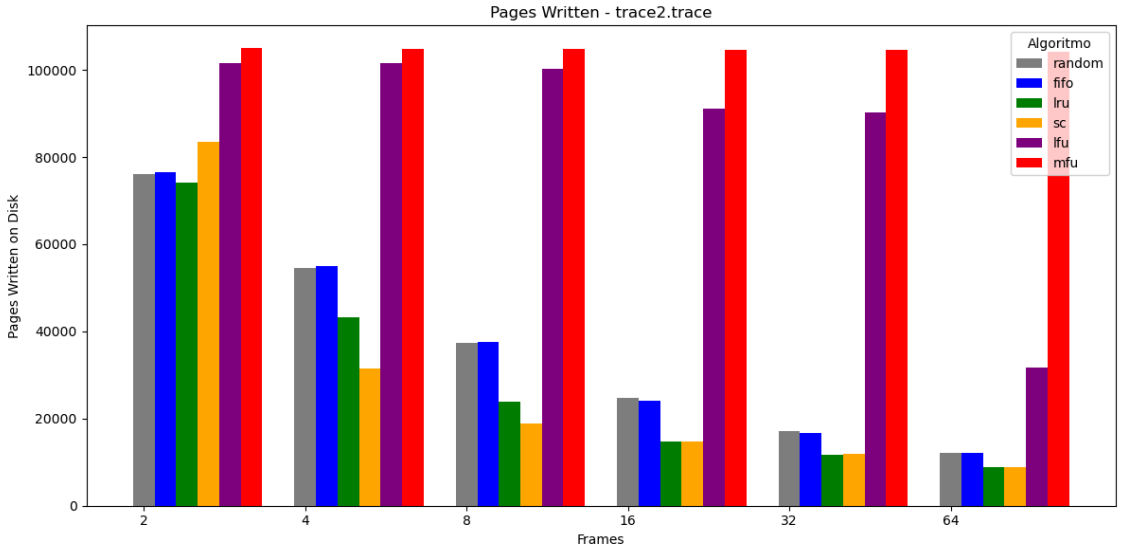
Em contrapartida, algoritmos como LRU, SC e FIFO demonstram maior eficiência, com reduções expressivas no número de escritas à medida que o número de frames aumenta. O algoritmo Random apresenta um desempenho intermediário, com resultados variados, mas ainda superando SC e FIFO em alguns cenários.



1.2.2. Trace 2

No segundo trace, o comportamento geral se repete: MFU e LFU continuam sendo os algoritmos com os maiores números de *pages written*, independentemente da quantidade de frames. O MFU, em particular, atingiu valores elevados mesmo com 64 frames disponíveis, sem apresentar melhora significativa em nenhum dos cenários.

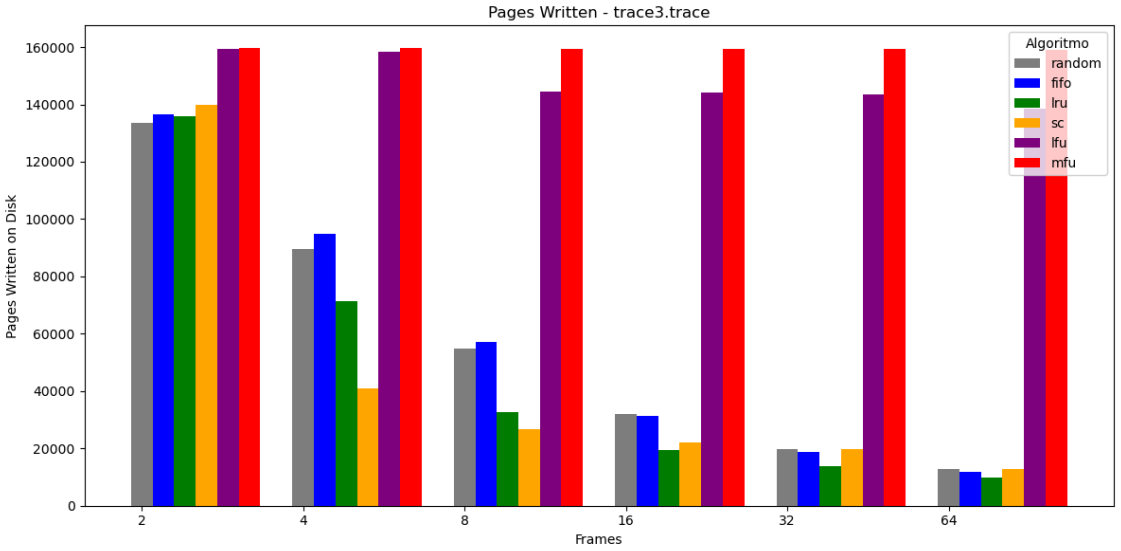
Por outro lado, os algoritmos LRU, SC e FIFO voltam a demonstrar maior eficiência, com quedas consistentes na quantidade de páginas escritas à medida que o número de frames aumenta. O algoritmo Random manteve-se em um nível intermediário, com desempenho próximo ao do FIFO em praticamente todas as configurações.



1.2.3. Trace 3

No trace 3, foram observados alguns resultados distintos em relação aos anteriores. Houve um número de *pages written* bastante elevado para todos os algoritmos quando apenas 2 frames estavam disponíveis. Nos demais tamanhos de frame, os valores seguiram uma tendência semelhante à dos traces previamente analisados.

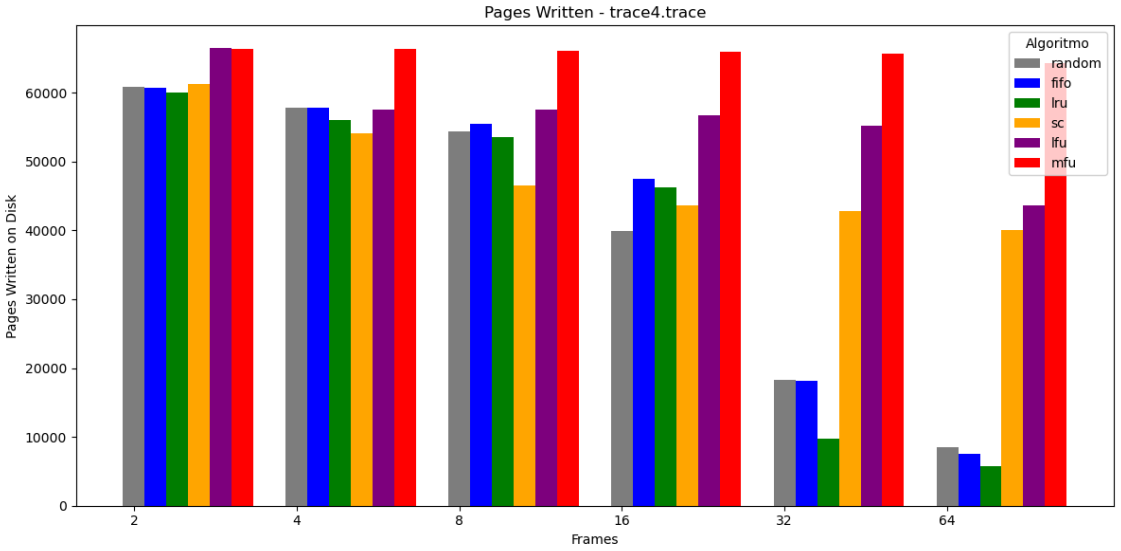
Os algoritmos LFU e MFU mantiveram-se ineficientes, apresentando valores elevados de escrita em disco independentemente da quantidade de frames. Já os algoritmos LRU, SC e FIFO, com exceção do cenário com 2 frames, mostraram-se mais eficientes, com redução gradual nas escritas à medida que o número de frames aumentava. Vale destacar o desempenho do SC no cenário com 4 frames, onde superou significativamente os demais algoritmos. O algoritmo Random seguiu semelhante ao FIFO



1.2.4 Trace 4

O trace 4 apresentou resultados significativamente diferentes dos demais analisados até o momento. Nota-se que todos os algoritmos tiveram desempenho inferior em grande parte das configurações, especialmente com 2, 4, 8 e 16 frames, cenário em que os algoritmos que anteriormente se destacavam, como LRU, FIFO e SC passaram a ter desempenhos semelhantes aos dos sempre problemáticos LFU e MFU.

LRU e FIFO só voltaram a apresentar bons resultados a partir de 32 frames, recuperando parcialmente sua eficiência. Vale destacar o comportamento do SC, que nos traces anteriores se mantinha entre os melhores algoritmos, mas que neste trace teve desempenho consistentemente inferior, ficando próximo do LFU em praticamente todos os cenários. Já os algoritmos LFU e MFU continuaram com os maiores números de *pages written*, mantendo o padrão observado nos outros testes. O algoritmo Random seguiu semelhante ao FIFO.



2. Conclusão

Com base em todos os testes realizados, foi possível observar que algoritmos como LRU, FIFO e Second Chance (SC) apresentaram, de modo geral, os melhores desempenhos nos cenários testados, especialmente quando a quantidade de frames era maior. O LRU destacou-se por apresentar os menores índices de *page faults* na maioria dos *traces*. No entanto, em alguns casos, como no *trace* 4, foi superado pelo SC, indicando que ambos possuem desempenho médio bastante semelhante. O FIFO apresentou um médio desempenho, pois, em diversos cenários, acabou se assemelhando ao Random, que não utilizou nenhum critério sofisticado na escolha da página substituída, mas, ainda sim, o FIFO obteve resultados pouco inferiores aos resultados do LRU e SC.

Por outro lado, os algoritmos LFU e MFU, baseados em frequência de uso, mostraram-se pouco eficazes. Em praticamente todos os testes, mantiveram altos índices de *page faults* e *pages written*, demonstrando que a frequência não é um bom critério para substituição de páginas. Os resultados também evidenciaram que o algoritmo Random, embora simples, apresentou desempenho relativamente estável, ficando consistentemente próximo do FIFO em quase todos os testes.

Por fim, a análise dos *pages written* reforçou os resultados obtidos em *page faults*, com LFU e MFU gerando mais escritas em disco e, consequentemente, aumentando o custo de operações de E/S. Já os algoritmos LRU, FIFO e SC mostraram-se mais eficientes nesse aspecto, reduzindo a necessidade de escrita.

Dessa forma, conclui-se que a escolha do algoritmo de substituição de páginas impacta significativamente o desempenho do sistema. Algoritmos baseados em tempo de uso recente tendem a oferecer melhores resultados na maioria dos cenários. Entretanto, entre os algoritmos que não utilizam frequência como critério, o desempenho pode variar dependendo do padrão de acesso da aplicação, havendo situações em que um algoritmo se destaca mais que os outros conforme o contexto de uso.

References

Neto, João. Sistemas Operacionais. [Slides]. Universidade Federal do Rio Grande do Norte, 2025. Acesso em: 11 jul. 2025.