Gerenciamento de memória

Políticas de trocas de páginas

4 de setembro de 2021

autor: Genilton Barbosa - 5153

VISÃO GERAL

Foi dado como objetivo a implementação dos algoritmos de troca de página, e a análise de desempenho entre eles.

INTRODUÇÃO

No contexto do gerenciamento de memória, a paginação da memória tem como objetivo uma utilização mais eficiente da mesma. Esse processo é feito através da virtualização da memória, que significa subdividir a memória física em partições chamadas de molduras(frames), essas partições serão ocupadas pelas páginas, que serão suas correspondentes virtuais, entretanto o número de páginas e por consequência a memória virtual pode exceder o tamanho da memória física, portanto é necessário realizar a paginação, que é a troca de páginas entre a memória principal e secundária.

ALGORITMOS

Os algoritmos de substituição de páginas devem exercer sua função da forma mais otimizada possível, para minimizar o número de pages faults (Quando é buscado uma página que não está na memória principal).

- FIFO: Substitui a página que está na memória há mais tempo. Para a implementação desse algoritmo foi mantida a referência a primeira página inserida na memória, quando solicitado uma troca de página o algoritmo retorna essa página, e incrementa a referência para segunda página que entrou na memória.

- FIFO + Bit R (Segunda Chance): Esse usa como base o FIFO, com aditivo de uma verificação no bit de referência, quando uma página é escolhida para troca através do FIFO, e verificado se ela foi referenciada recentemente, caso tenha sido, ela vai para o final da fila e tem seu bit de referência alterado para 0. e então é feito a verificação na segunda página da fila, e assim por diante.

- NRU: O algoritmo NRU (Not Recently Used) procura por páginas que não foram referenciadas nos últimos acessos para serem substituídas. Tal informação é mantida através dos bits de referência e de modificação. As substituições das páginas seguem a seguinte prioridade: páginas não referenciadas e não modificadas, páginas não referenciadas, páginas não modificadas e páginas referenciadas e modificadas. Para a implementação o algoritmo percorre as páginas procurando uma que se encaixe na primeira prioridade, caso não encontre, ele procura pela segunda e assim por diante.

- Aging(Envelhecimento): Retorna a página mais velha, para implementação cada página mantém um contador que é incrementado quando referenciado e decrementado com o passar do tempo, quando uma troca é solicitada ele retorna a página com menor contador.

- LRU: O algoritmo LRU substitui a página que não foi referenciada pelo maior período de tempo. A implementação desse algoritmo foi feita utilizando uma lista duplamente encadeada e um array acessado através de uma função hash. Quando necessário remover uma página e selecionando a página do Head, que é a mais velha na lista, quando a página é adicionada ou referenciada, ela vai para o Tail. Cada nó da lista é referenciado por uma célula do array hash, com intuito de acelerar o acesso, pois acessar o array utilizando a função hash e O(1), enquanto o acesso diretamente na lista e O(N).

- Random (já implementado): Retorna aleatoriamente a página.

METODOLOGIA DE TESTES

Para os testes foram utilizados strings de referência, que são uma forma de simular as requisições de páginas em um ambiente de teste.

Existem três possíveis fontes de strings de referência. Uma e gerar as requisições com um gerador de números aleatórios para gerar referência sintética. Outra possível forma é utilizando os princípios de localidade temporal e local, para que assim tenha uma representação sintética mais fiel com a realidade.

Também existe o modelo em que as strings de referência reais são adquiridas do rastreamento de processos de tempo virtual com o método de rastreamento de hardware.

Nos testes a seguir foram utilizados strings de referência gerada por um algoritmo aleatório, que tenta simular o princípio da localidade e da temporariedade. Para isso as páginas iniciam com a mesma probabilidade de serem referenciadas, essa probabilidade é incrementada em +4 quando é referenciada diretamente e em +2 quando um dos seus vizinhos é referenciado. Existe uma taxa de esquecimento que diminui em -1 a probabilidade de todas as páginas a cada referência, exceto se já estiver no valor mínimo que é 1. A distância que define quantos vizinhos de uma determinada página terão sua probabilidade aumentada e definida previamente no argumento range. Para executar o algoritmo use:

- ./program <pages> <requests> <range>

Passando assim o número de páginas <pages>, acessos a memória <requests>, e o alcance das alterações <range>. Por padrão nestes testes o range foi definido como 10% do número de páginas. Os tipos de acesso foram gerados aleatoriamente.

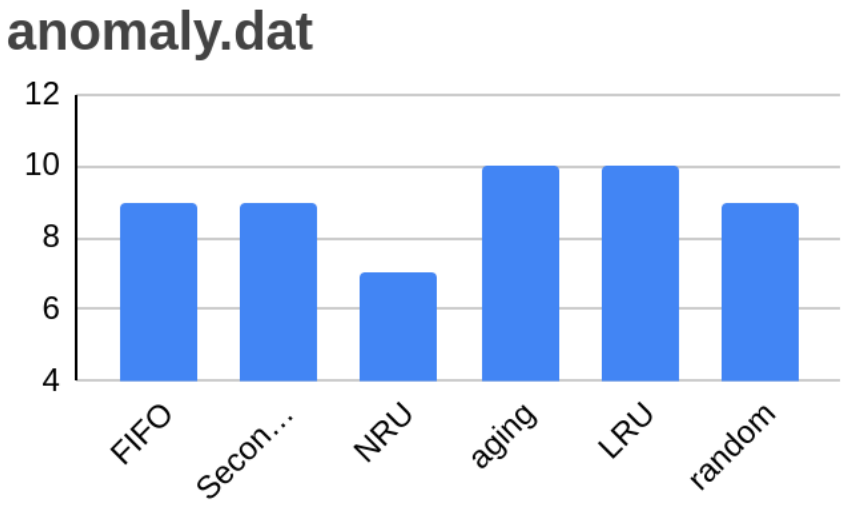
RESULTADOS

Os testes a seguir foram executados utilizando clock padrão de 10. O comando utilizado para executar a simulação:

- program <algorithm> <clock_freq>

Resultados:

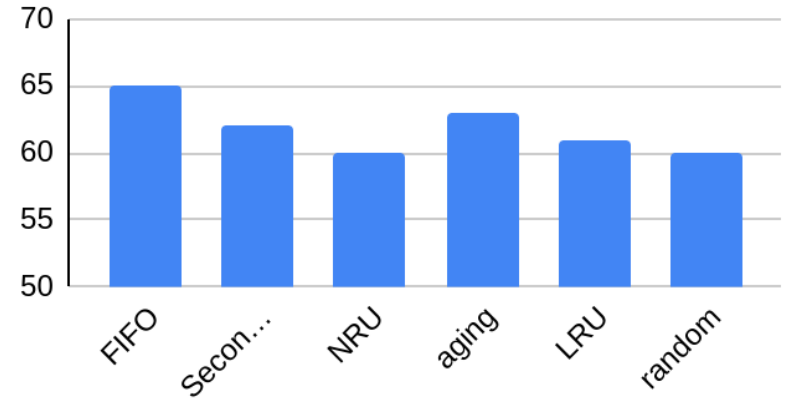
	anomaly	test1	test2	test3	test4	test5	test6	Total
FIFO	9	65	50	144	121	246	189	824
Second Chance	9	62	48	143	124	246	195	827
NRU	7	60	51	144	117	243	193	815
aging	10	63	50	140	125	241	213	842
LRU	10	61	52	144	123	244	190	824
random	9	60	56	147	130	245	275	922



Páginas: 10

Frames: 3

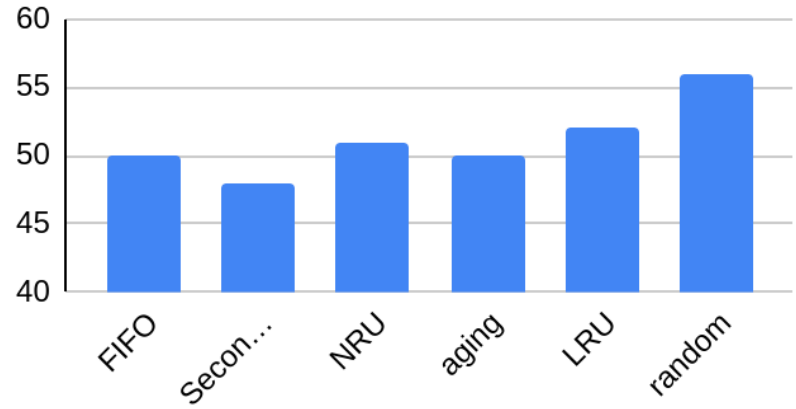
test1.dat



Páginas: 20

Frames: 6

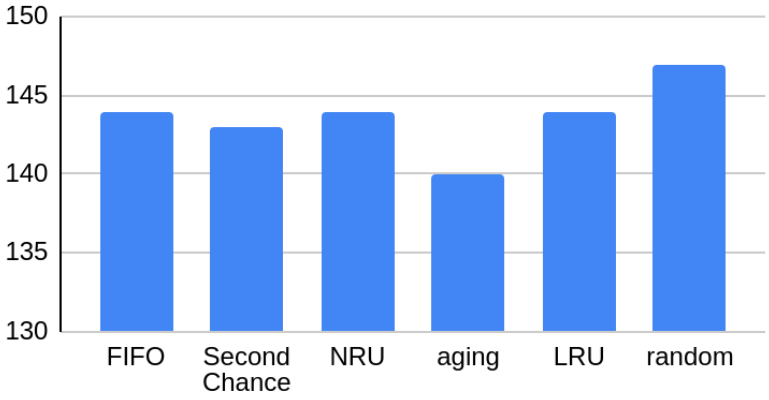
test2.dat



Páginas: 20

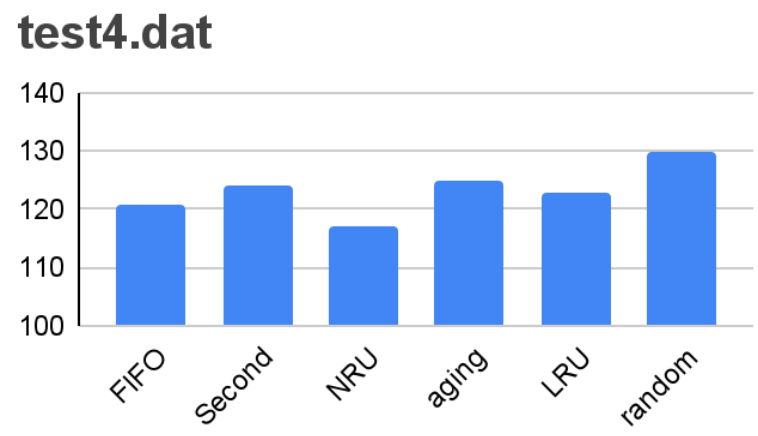
Frames: 9

test3.dat



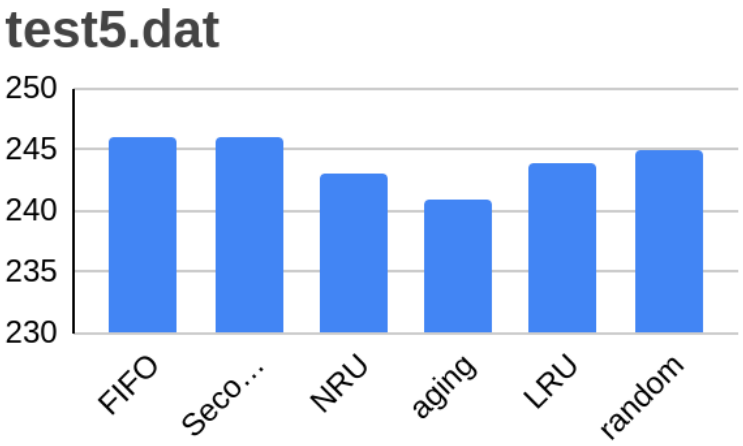
Páginas: 40

Frames: 10



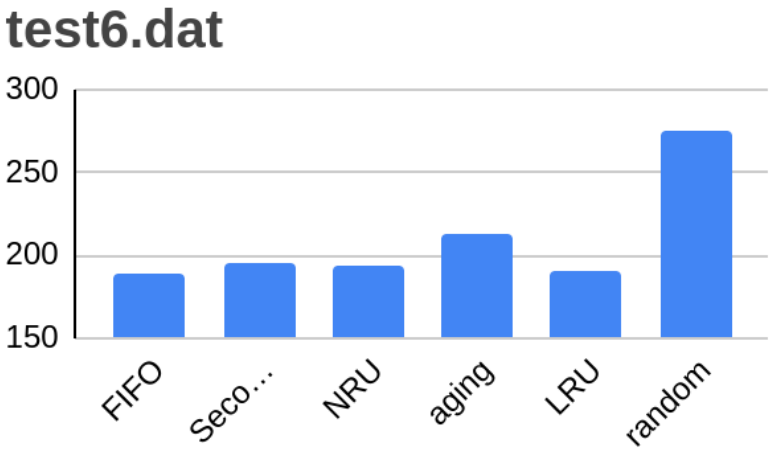
Páginas: 40

Frames: 15



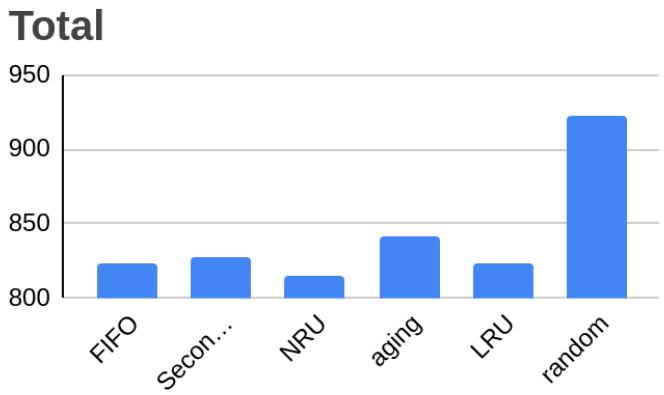
Páginas: 100

Frames: 20



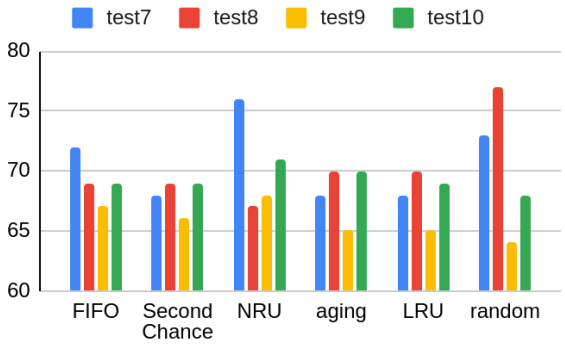
Páginas: 100

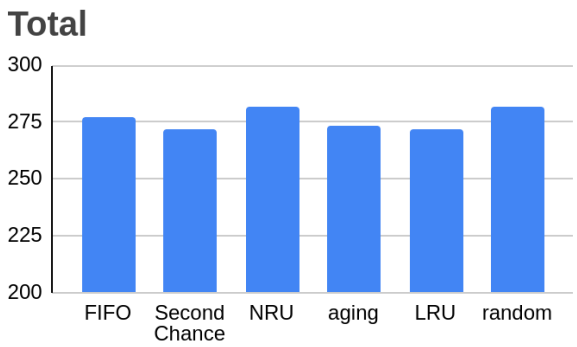
Frames: 35



Os próximos resultados são de strings de referências geradas aleatoriamente:

	test7	test8	test9	test10	Total
FIFO	72	69	67	69	277
Second Chance	68	69	66	69	272
NRU	76	67	68	71	282
aging	68	70	65	70	273
LRU	68	70	65	69	272
random	73	77	64	68	282





Esses 4 últimos testes tiveram Páginas: 20, Frame: 6.

CONCLUSÃO

Com os resultados obtidos, podemos observar que o algoritmo NRU, obteve os melhores resultados, pois obteve menor número de pages faults, seguido do FIFO, e do Second Chance.

Se considerarmos o nível de complexidade O(n) do algoritmo NRU, que apesar de não ser tão custoso quanto o aging, e o LRU, ainda assim é um algoritmo que consumiria mais recursos que o Second Chance e o FIFO, portanto em vista da pequena margem de vantagem do NRU sobre o FIFO e o Second Chance, podemos tomar que esse dois algoritmos apresentam o melhor custo benefício, por serem mais simple, e menos custosos.

Já nos 4 últimos testes que foram gerados totalmente de forma aleatória, o algoritmo NRU não obteve bom desempenho, dessa vez o Second Chance e o LRU foram melhores. Isso mostra que de fato o Second Chance, é o melhor custo benefício.

É importante salientar que esses testes foram feitos com strings de referências sintéticas, geradas aleatoriamente, e apesar das técnicas de localidade local e temporal empregadas para tentar simular um cenário real, ainda assim essas strings não necessariamente condizem com a realidade. Portanto, a única conclusão que podemos tomar, é sobre esse cenário fictício proposto.

REFERÊNCIAS

- W. James Wittneben. Design and evaluation of a reference string sampling

- method. Iowa State University, 1979, Disponível: https://lib.dr.iastate.edu/cgi/viewcontent.cgi?referer=https://www.google.com/&httpsredir=1&article=8315&context=rtd

- Buron filter and lru cache, Programmer All, Disponivel: https://www.programmerall.com/article/79742215934/

- LRU Cache & Bloom Filter, Programmer All, Disponivel: https://www.programmerall.com/article/68161527510/