EP3 MAC0422 2017-2

Guilherme Costa Vieira
Nº USP: 9790930
Victor Chiaradia Gramuglia Araujo
Nº USP: 9793756

Simulação.

1. No começo da simulação todos os dados do arquivo são mandados para uma lista com os processos que ainda não chegaram e uma lista com os tempos em que as compactações irão ocorrer.
    a. A simulação roda até as duas listas estarem vazias:
        i. Quando um processo tem seu t0 igual ao tempo da simulação (time), ele será passado da lista to_arrive para a l_procs (algoritmos de fit são usados aqui).
        ii. Acessamos a memória física (algoritmos de paginação).
        iii. Removemos processos da l_procs quando tf == time.
    b. E por fim, compactamos a memória.

REPRESENTAÇÃO INTERNA DA MEMÓRIA

1. V_mem : É uma lista de listas que possui o PID do processo (-1 se não houver processo), a primeira página ocupada por ele e a última página (não inclusiva).
   a. Ex : [[4, 0, 3], [-1, 3, 5], [8, 5, 8]]
2. P_mem : É uma lista que representa a memória física e está dividida em páginas. Cada índice contém a página da memória virtual que está carregada (-1 se está vazia).
   a. Ex : [-1, 2, 7, 4]
3. Com nossa função glue_mem(), garantimos que em na v_mem não há dois itens consecutivos com PID -1.

Algoritmos de fit.

Best e worst fit :

1. Procuram na v_mem uma posição vazia que possa comportar o novo processo e seja uma posição “melhor” (de acordo com o propósito do algoritmo) que a anterior.
2. Separamos a memória se necessário (buraco contém mais páginas do que o necessário).
    a. Juntamos a nova posição vazia com uma vizinha se ela existir (glue_mem()).

Algoritmos de fit.

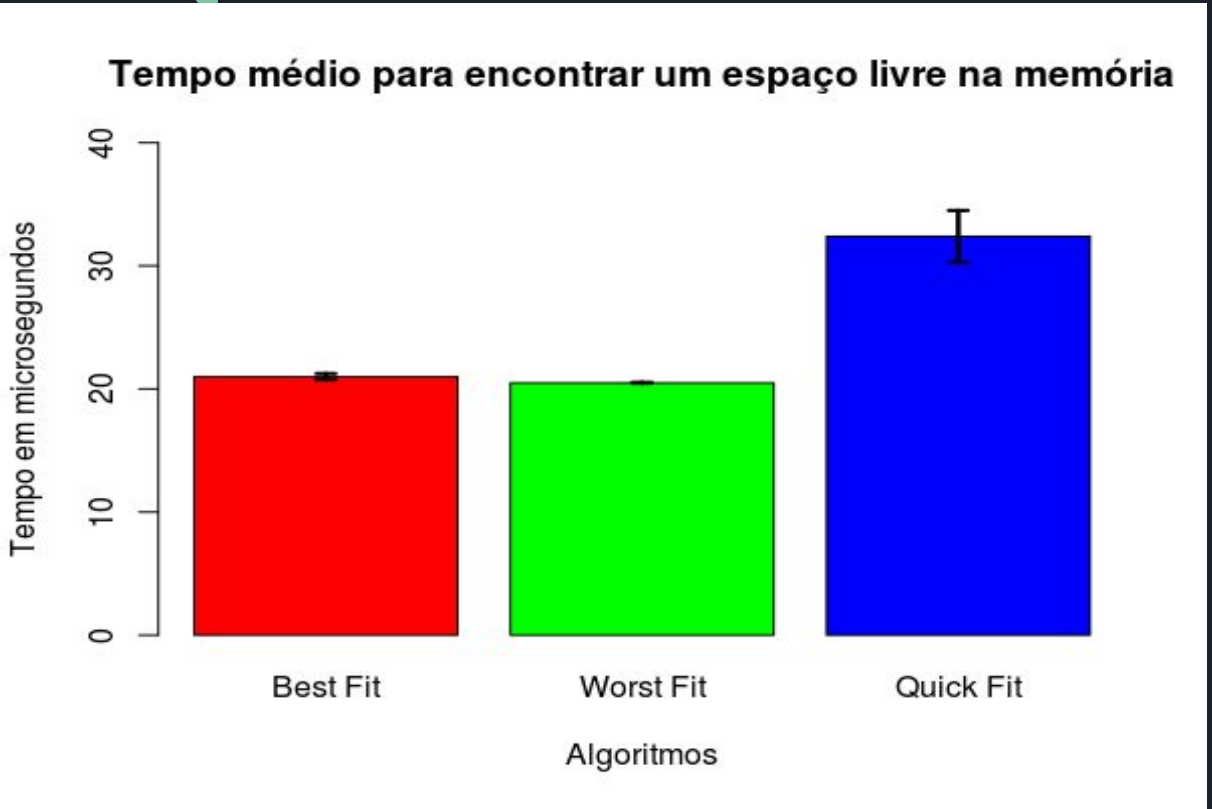
Quick_fit :

1. Temos uma lista com todas as páginas que serão requisitadas pelo menos 10% das vezes.
2. Uma lista de listas que possui todas a posições da v_mem que comportam aquele tamanho de página.
3. Procura-se um espaço para o processo se seu tamanho for muito requisitado.
   a. A lista com as posições livres é atualizada.
4. Se a página requisitada não estiver na lista, o algoritmo best_fit é utilizado.

Algoritmos de paginação.

1. Todo algoritmo de paginação possuem uma estrutura extra para ajudar a decidir qual página (índice da p_mem) será substituída.
2. Em todos os algoritmos se houver uma posição vazia, ela é escolhida.
3. A p_mem é percorrida para se saber se houve page_fault.
4. LRU4: Usamos uma matriz para implementar os contadores, onde a primeira coluna corresponde ao bit R.
5. OPTIMAL: Lista que contém o processo, a posição que será acessada e o instante de tempo do acesso.
    a. O OPTIMAL é o único algoritmo em que sua estrutura é criada enquanto o arquivo de trace é lido.

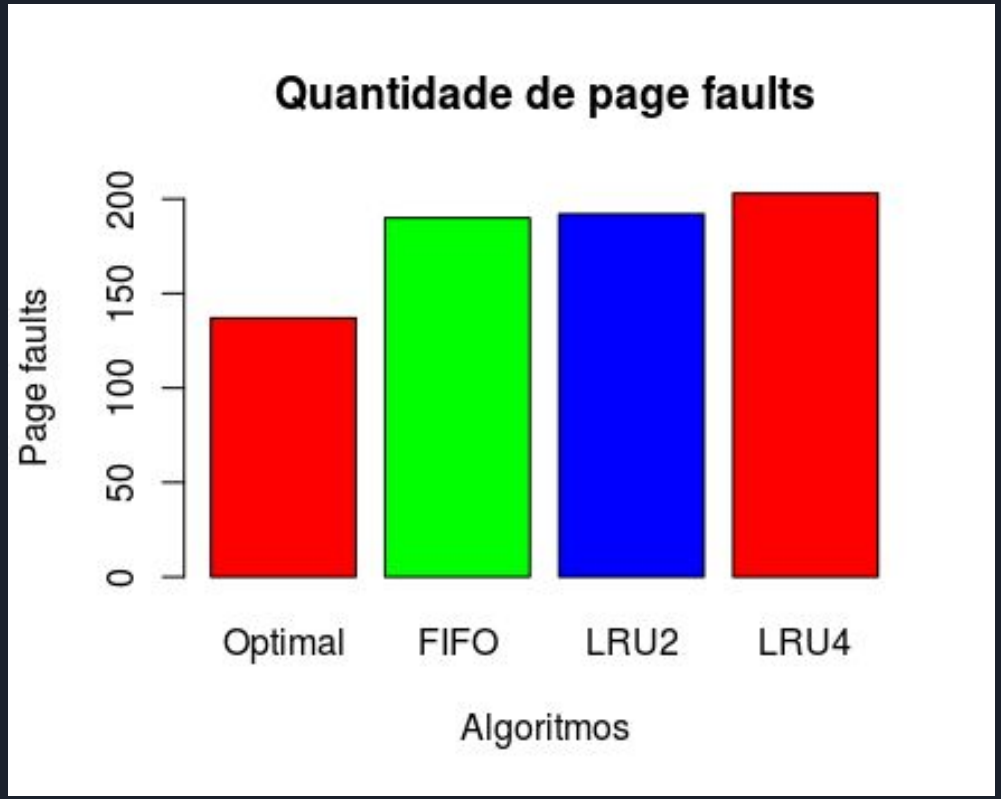
Gráficos dos algoritmos de gerência de espaço livre



Confiança de 95%

- Média Best Fit = 21µs
  - IC = (20.75µs, 21.25µs)
- Média Worst Fit = 20.5µs
  - IC = (20.464µs, 20.536µs)
- Média Quick Fit = 32.4µs
  - IC = (30.3µs, 34.5µs)

Gráficos dos algoritmos de páginação



- Optimal = 137
- FIFO = 191
- LRU2 = 192
- LRU4 = 203

Conclusão:

Fit :

Usamos o tempo em que o programa usa a função de fit, por isso no Quick Fit também contamos o tempo necessário para “consertar” a lista com posições livres, que se mostrou uma operação muito custosa em tempo.

Em nossos testes o tempo do quick_fit aumentava conforme os testes acabavam.

O Best Fit e o Worst Fit tiveram resultados muito semelhantes com leve vantagem para o Worst Fit devido sua média e variância menores.

Contudo, todos os algoritmos tiveram resultados na mesma ordem de grandeza de tempo.

Conclusão:

Paginação :

Não há um intervalo de confiança pois todos os resultados de cada algoritmo é igual para cada medição para um determinado trace .

Mudar o tempo em que o bit R é atualizado muda os resultados do LRU4.

Os algoritmos testados (FIFO, LRU2, LRU4) tiveram resultados semelhantes, com leve vantagem para o FIFO.

O Optimal foi cerca de 40% melhor que os outros algoritmos.

Trace :

64 processos, 6 compactações, memória física de tamanho 96 bytes, virtual de tamanho 612.

S = 4, P = 12.