Fundamentos de Sistemas de Operação

Unix Windows NT Netware MacOS DOS/VS Vax/VMS Linux Solaris HP/UX AIX Mach Chorus

Gestão de Memória:

MV por Paginação a pedido

(revisões/continuação)

Memória Virtual: RAM "cheia" (1)

- Que fazer quando não há nenhuma frame livre?
  - Gestão de Memória: libertar uma frame "à força"
    - Em primeiro lugar, tentar libertar uma frame ocupada por uma página não modificada ➔ não é preciso copiá-la para disco;
    - Se não for possível, escolher uma "vítima", de acordo com uma dada política,
      - Copiá-la para disco, para a área de paging (por razões históricas alguns sistemas, e.g., Unix, designam-na de swapping).
    - Marcar a frame como livre na estrutura de dados de gestão das frames,
    - Marcar a página como "não presente" no mapa de páginas do processo, e associá-la ao seu endereço em disco.

Memória Virtual: RAM "cheia" (2)

- Que política para libertar frames ocupadas?
  - Objectivo: Minimizar o número de faltas de página.
  - Mas como?
    - Com um bom algoritmo de substituição de páginas…
    - FIFO: a vítima é a página mais antiga em memória
      - Mas a página mais antiga pode ser muito acedida! ☹
    - Um algoritmo que escolha a página **menos** acedida!
      - Mas uma página pouco acedida no passado, pode vir a ser muito acedida no futuro... mas não podemos saber o futuro ☺
      - E como saber se uma página é a menos acedida?
        - Associar um timestamp a cada página,
        - Procurar a página de timestamp mais antigo…

Memória Virtual: RAM "cheia" (3)

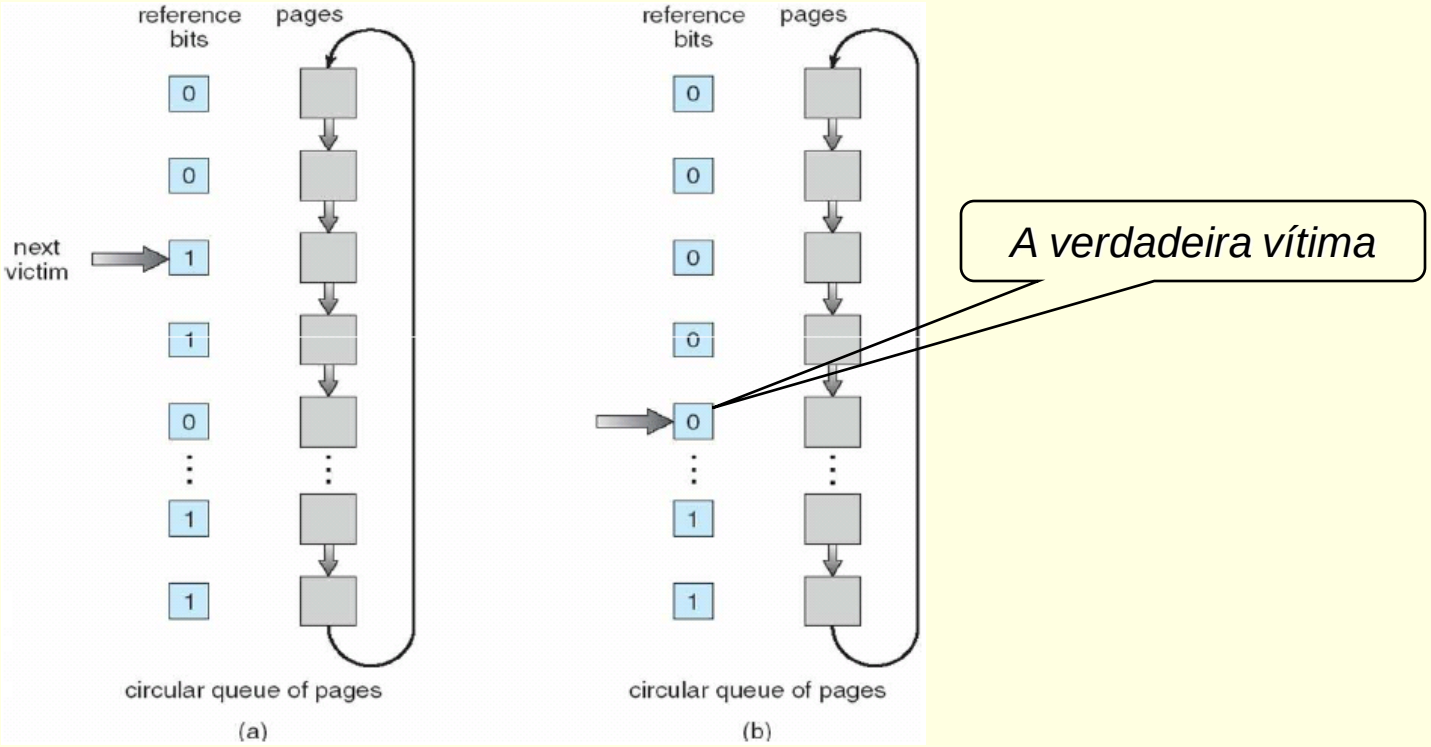
- Podemos realisticamente procurar a página mais antiga?
  - Percorrer 1000000 de timestamps num sistema com 4GB de RAM e páginas de 4KB não é viável, demora muiiiiito!
  - Solução?
- Aproximação: página recentemente menos usada (LRU)
  - Apenas 1 bit, inicializado a zero!
    - Página acedida, bit a 1
    - Escolher uma vítima com o bit a zero
      - E se não houver?
      - É preciso arranjar maneira de pôr os bits de novo a zero, senão não dá ☺

Memória Virtual: RAM "cheia" (4)

- Aproximação LRU: como pôr os bits a zero, "de vez em quando"?
  - Força bruta: de vez em quando, "zerar" tudo
  - Algoritmo da "2ª oportunidade" (second chance)
    - As frames são tratadas como se pertencessem a uma lista circular (ver próxima pag.)
      - Num dado momento o apontador aponta uma vítima potencial
      - Se a vítima tem o bit a um, ele é posto a zero, deixa-se a página em memória, e considera-se a vítima seguinte…
      - Quando aparecer uma vítima com o bit a zero, é escolhida!

Memória Virtual: RAM "cheia" (5)

- Aproximação LRU "2ª oportunidade": um "apontador" regista a próxima vítima potencial…



Memória Virtual: RAM "cheia" (6)

- Faltas globais vs. locais
  - Locais: vítimas, só do próprio processo
  - Globais: a vítima pode ser doutro processo
- Proactividade para fugir à "memória cheia" (Linux)
  - Em vez de só ejectar páginas quando um processo precisa de uma frame e não as há livres,
  - Há 2 "constantes" configuradas no SO: **high** e **low watermark -** linhas de água de (maré)"alta" e "baixa"
  - Se o SO detecta que a ocupação de memória excede high, é lançado um processo-kernel (daemon) que "desata" a ejectar páginas dos processos, libertando memória…

Memória Virtual: RAM "cheia" (7)

- … e muito mais havia para dizer… ☺

Memória Virtual: Notas soltas (1)

- Definição: Working Set (de um processo)
  - Conjunto de páginas do EE (virtual, naturalmente) do processo que num dado instante residem em memória.
- Para que um processo "corra" de forma eficiente há um número mínimo de páginas que tem de estar em memória[1] (minimum working set size)…
  - Senão há PF atrás de PF
- Em alguns SOs é possível garantir que um processo tem sempre direito a um número mínimo de páginas em memória (AOS/VS)

[1] Não chega… se o processo não exibir localidade de referência, também haverá PF atrás de PF…

Memória Virtual: Notas soltas (2)

- Optimização das PFs
  - Numa falta de página, carrega-se não a página mas também umas “quantas” vizinhas – um cluster de páginas (usado no Windows, AOS/VS, VAX/VMS)

E quando… o SO está "lixado"? (1)

- **Thrashing** - Considere a seguinte situação:
  - P_x corre, mas sofre logo um page fault (PF)
    - Não há RAM livre, logo uma vítima é escolhida; mas essa vítima é uma página de P_y
  - Agora corre P_y, mas sofre logo um PF
    - Não há RAM livre, logo uma vítima é escolhida; mas essa vítima é uma página de P_z
  - Agora corre P_z, mas sofre logo um PF
- Diz-se "o sistema está em **thrashing**" (lixado)

E quando… o SO está "lixado"? (2)

- **Thrashing**
  - O SO gasta tanto tempo na sua própria gestão que não sobra tempo para executar os processos aplicacionais
    - Não há RAM livre, logo uma vítima é escolhida; mas essa vítima é uma página de P_y
- Soluções
  - Controle de admissão: recusar lançar mais processos. Usado por alguns SOs mais antigos (AOS/VS)…
  - Linux: OOM (Out-Of-Memory) Killer – processo do kernel que mata processos de utilizador se detecta que há uma escassez de memória (escolhe um que esteja a gastar muita memória)