Instituto de Informática - UFRGS

Sistemas Operacionais

Políticas de substituição de páginas

Aula 17

Introdução

- Paginação por demanda
  - Na ocorrência de *page-fault* é necessário realizar *page-in*
  - Duas situações:
    - Com memória disponível (quadro): simplesmente executa page-in
    - Sem memória disponível (quadro): necessário liberar um quadro e executa page-in.
  - Manter uma ocupação eficiente da memória
- Mecanismo de memória virtual realiza duas decisões
  - Página a ser substituída
  - Quantidade de memória (quadros) alocado para cada processo

Instituto de Informática - UFRGS
A. Carissimi -8-mai-12

Política de substituição

- Objetivo principal
  - Substituir a página que provavelmente não será referenciada em um futuro próximo.
- Critérios gerais de escolha
  - Determinar a página menos necessária
  - Otimizações: nem toda página necessita sofrer *page-out*
    - Páginas não modificadas
    - Páginas *read-only* (código)
  - Páginas que nunca deve ser substituídas (*frame locking* ou *pinning*)
    - e.g.: código e estruturas de dados do sistema operacional, buffers de E/S

Instituto de Informática - UFRGS
A. Carissimi -8-mai-12

Algoritmos de substituição de páginas

- Implementam uma política de substituição
- Algoritmos básicos
  - MIN ou OPT: algoritmo ótimo (referencial)
  - FIFO (First-in, first-out): critério de “antiguidade”
  - LRU (*least recently used*): critério de utilização
  - VMIN ou VOPT: variação do algoritmo ótimo
  - *Working set*: critério de conjunto de páginas em uso

Instituto de Informática - UFRGS
A. Carissimi -8-mai-12

Avaliação dos algoritmos de substituição

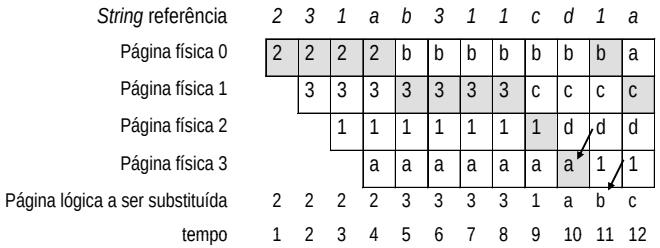
- Baseado em *string* de referência
  - Modelo que representa o traço de execução de um programa
  - $1 \leq m \leq n$
    - m: número de página alocada em memória por um processo
    - n: número de páginas distintas no string de referência
  - Simula uma sequência de acessos a páginas
    - e.g.: string de referência: 6 4 7 2 3 1 5 6 7 8 9 1 2 9 3 5 6
- Critérios de avaliação
  - Número de falta de páginas
  - Quantidade total de páginas carregadas na memória

Algoritmo ótimo

- Seleciona para substituição uma página que não é mais necessária ou que será referenciada dentro do maior intervalo de tempo
  - Impossível de se ter conhecimento do "futuro"
  - Emprego de aproximações
  - Serve como um ponto de referência para comparar aproximações

First-In, First-Out (FIFO)

- Quadros na memória principal são alocados a páginas na forma de um *buffer* circular
  - Simples implementação
  - Substitui a página que está a mais tempo na memória
- Exemplo:



Problemas com FIFO

- Não considera o uso da página
  - Página substituída pode ser necessária logo em seguida
- Anomalia de Belady
  - Taxa de falha pode aumentar enquanto que a quantidade de quadros alocados aumenta

*Least Recently Used** (LRU)

- Página a ser substituída é a página que foi referenciada a mais tempo
  - Pelo princípio da localidade, esta página deve ser a de menor probabilidade de ser referenciada em um futuro próximo
  - Cada página deve possuir a "data" da última referência
- Exemplo:

String referência	2	3	1	a	b	3	1	1	c	d	1	a
Página física 0	2	2	2	2	b	b	b	b	b	d	d	d
Página física 1		3	3	3	3	3	3	3	3	3	3	a
Página física 2			1	1	1	1	1	1	1	1	1	1
Página física 3				a	a	a	a	a	c	c	c	c
Página lógica a ser substituída	2	2	2	2	3	1	a	a	b	3	3	c
tempo	1	2	3	4	5	6	7	8	9	10	11	12

*Menos recentemente usada

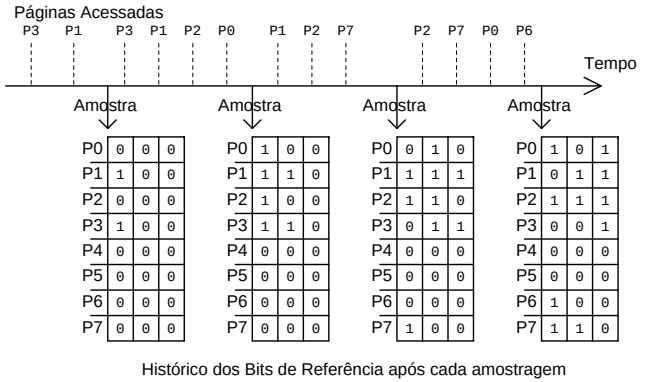
Implementação de *Least Recently Used* (LRU)

- Contadores por entrada da tabela de páginas
  - Tempo da última referência (*time stamp*)
  - Desvantagem é o *overhead*
    - Atualização da entrada na tabela de páginas
    - Seleção da página vítima
- Uma aproximação é empregar uma pilha
  - Página referenciada é inserida no topo da pilha
  - Topo da pilha está a página referenciada mais recentemente
  - Base da pilha está a página referenciada menos recentemente
  - Na realidade se trata de um lista duplamente encadeada
- Desvantagem: custo de processamento (*software*)

Aproximações de LRU via suporte de hardware

- Hardware (MMU) deve suportar bit(s) de referência
- Bit(s) de referência podem ser utilizados para aproximar LRU
  - Bit de referência = 1 (página acessada); 0 (página não acessada)
  - Substitui página cujo bit de referência é igual a zero
    - Desvantagem: não se sabe o "uso passado" e a ordem
- Melhoria: Algoritmo de histórico de bits de referência
  - Incluir vários bits de referência adicionais (registrador)
  - A cada Δt consulta o bit de referência e atualiza registrador (*shift register*)

Exemplo: construção de histórico de bits de referência



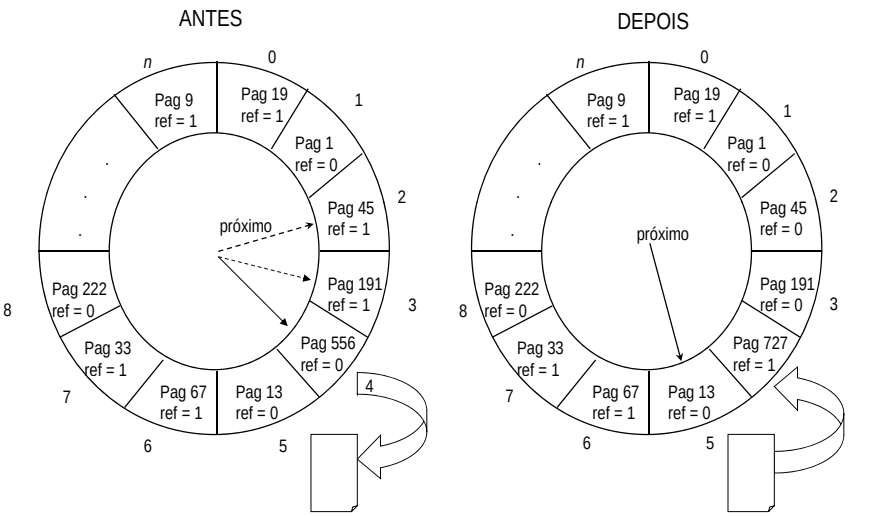
Histórico dos Bits de Referência após cada amostragem

Desvantagem: custo de implementação em hardware (área) e em software (processamento)

Algoritmo de segunda chance (relógio)

- Aproxima o LRU empregando um único bit de referência (bit r)
  - Também denominado de bit de uso ou bit de acesso
- Mantém uma lista circular com todas as páginas residentes na memória e um ponteiro para a página “próxima vítima”
  - Ordenamento FIFO
  - “Segunda chance” vem do fato que a página com bit r=1, recebe a chance de ser referenciado antes da próxima seleção de vítima
- Funcionamento: se o ponteiro aponta para uma página com:
  - Bit r = 0: a página é substituída e o ponteiro avança de uma unidade
  - Bit r = 1 o bit é zerado e o ponteiro é incrementado de uma unidade

Esquematização do algoritmo de segunda chance



Algoritmo de segunda chance melhorado

- Evitar que página a ser substituída seja escrita no disco (*page-out*)
  - Páginas não modificadas na memória podem ser sobre-escritas
  - Hardware deve prover bit sujo (*dirty bit*); indicação se página foi modificada
- Seleção da página vítima é feita com base no bit *r* e no bit sujo (s)
  - Bit r=1; s=1: zera bit r, incrementa ponteiro de uma unidade
    - Usada recentemente e modificada
  - Bit r=1; s=0: zera bit r, incrementa ponteiro de uma unidade
    - Usada recentemente e não modificada
  - Bit r=0; s=1: zera bit s, incrementa ponteiro de uma unidade
    - Não usada recentemente, mas modificada
  - Bit r=0; s=0: página a ser substituída, incrementa ponteiro de uma unidade
    - Não usada recentemente, nem modificada

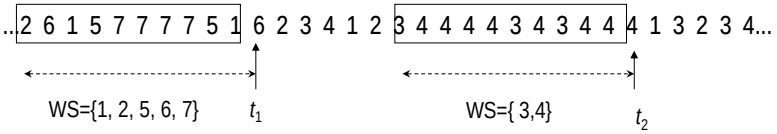
Variação do Algoritmo ótimo (VMIN ou VOPT)

- Ajusta o número de páginas em memória de acordo com as futuras referências
- Princípio básico:
  - Página P é acessada no instante de tempo t
  - Se página P não será acessada nas próximas τ referências, ela pode ser removida da memória
    - Define um janela de tempo (t; t + τ)
      - Tamanho da janela é τ + 1 acessos
- Fator τ é uma constante de projeto de sistema
- Não realizável devido a necessidade de conhecimento do futuro

Modelo *working-set*

- Aproximação do algoritmo VMIN que estima o futuro com base no passado
- Baseado no princípio da localidade
- Idéia é examinar as τ referências mais recentes em um instante *t*
  - Conjunto de páginas acessadas por um processo no intervalo (t - τ ; t)
    - Forma o *working set* do processo
  - τ é constante do projeto do sistema

τ = 10 unidades

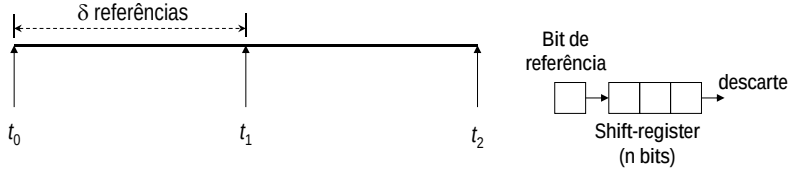


Problemas com *working set*

- Estimar o valor de τ
  - τ = valor pequeno: não abrange toda a localidade do processo
  - τ = valor grande: abrange várias localidades
    - $\tau = \infty$: abrange todo o programa
- Custo da implementação em tempo de processamento
  - *Working set* é recalculado a cada referência
  - Para reduzir o custo emprega-se aproximações baseadas no bit de referência em data de acesso (*time stamp*)

Definição e manutenção do working set (aproximação 1)

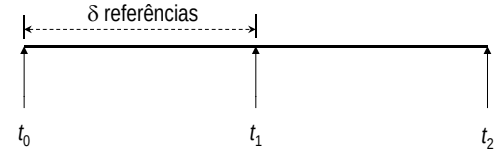
- Aproximação baseada na consulta periódica do bit de referência
  - Aumento da frequência de interrupção e do número de bits de referência melhora a precisão da aproximação
- Tamanho da janela é δ x n
- Página é eliminada do *working set* quando chega a zero



Definição e manutenção do *working set* (aproximação 2)

- Baseada no bit de referência e no tempo de acesso
- A cada δ referências:

```
se (bit_r == 1) então {
        bit_r = 0;
        tempo_acesso = tempo_atual;
}
senão {
        toff = tempo_atual - tempo_acesso;
        se (toff > tmax ) Remove página do WS;
}
```



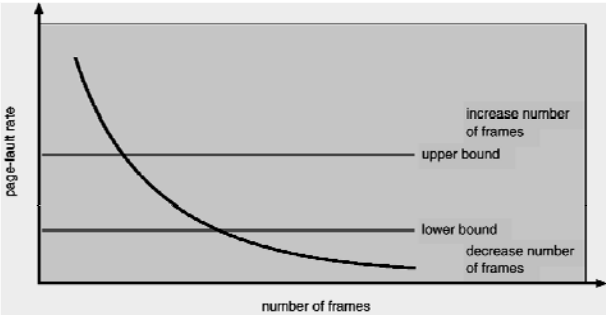
Usando *working set* para controlar *thrashing*

- Define-se WSS_i como o *working set size* do processo P_i
- Σ WSS_i é a quantidade total de quadros necessários no sistema em um dado instante
- *Thrashing* ocorre quando Σ $WSS_i > m$ (quantidade de quadros)
  - Suspende um ou mais processos para evitar essa situação
    - Realização de *swap* (processo "vítima" inteiro para o disco)

Método de frequência de falta de páginas (PFF*)

- Forma de gerenciar a alocação de memória de um processo
  - Não informa QUAIS páginas devem ser substituídas, mas sim QUANTAS páginas podem ser substituídas
- Indica quando se deve aumentar o diminuir a quantidade de memória alocada a um processo
  - Controla o tamanho do conjunto de alocação com base na taxa de falta:
    - Taxa alta: processo necessita de mais quadros
    - Taxa baixa: processo pode liberar quadros
  - Objetivo é manter a taxa de faltas de página dentro de um limite razoável

Método da frequência de falta de páginas



Leituras complementares

- A. Tanenbaum. *Sistemas Operacionais Modernos* (3ª edição), Pearson Brasil, 2010.
  - Capítulo 3: seção 3.4
- A. Silberchatz, P. Galvin; *Sistemas Operacionais*. (7ª edição). Campus, 2008.
  - Capítulo 9: seção 9.4 e 9.5
- R. Oliveira, A. Carissimi, S. Toscani; *Sistemas Operacionais*. Editora Bookman 4ª edição, 2010
  - Capítulo 7 (seções 7.2 e 7.3)