Aula 15

Memória Virtual - Desempenho

1.1 Contexto
1.2 Políticas de Reposição de Páginas
1.3 *Working-Set*
1.4 *Trashing*

E se encher a memória?

Usamos a memória toda, mas acessos às página que não estão na memória comtinuam:

- Escolha uma página vítima na memória principal;

- Escreva-a no disco

- Usa esta página para guardar a página referenciada recentemente

Então, é necessária uma **política de reposição de páginas**

Uma otimização

Mas, e se a página não tiver sido modificada?

- Implementa-se um bit de sujo/limpo, inicialmente limpo

- Escritas à página tornam-na suja

- A página só é escrita no disco se estiver suja

Análise da políticas de reposição

Nomenclatura: *page* – página virtual
frame – página física

Dada uma sequência de acessos às **páginas**, como o algoritmo se comporta?

A sequência pode ser calculada:

- Aleatoriamente;
- Através da monitoração da execução de programas reais

Exemplo:

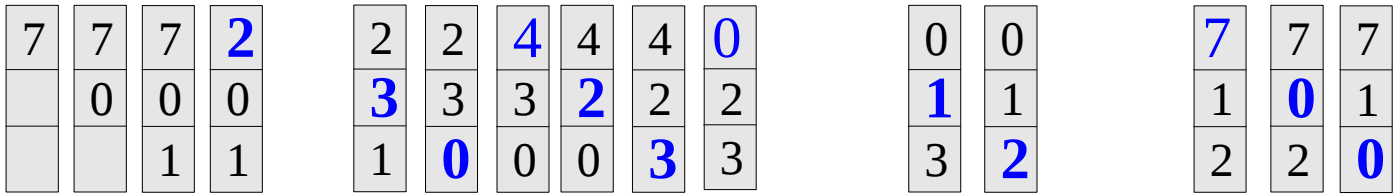
7,0,1,2,0,3,0,4,2,3,0,3,2,1,2,0,1,7,0,1

Política FIFO – *First In/First Out*

Retira-se a página que está há mais tempo na memória
- Simples
- Não muito eficiente

Exemplo:

7,0,1,2,0,3,0,4,2,3,0,3,2,1,2,0,1,7,0,1



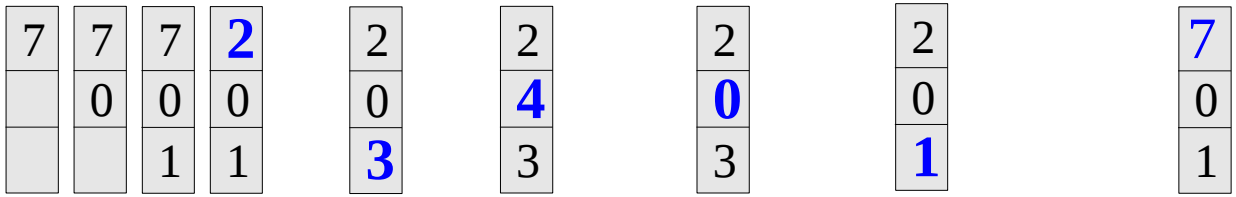
Problema: sofre da anomalia de **Belady**:
- Às vezes, com mais memória, **aumenta-se** o número de **page faults**!

Política Ótima

Retira-se a página que não vai ser usada por mais tempo
- Não é factivel!

Exemplo:

7,0,1,2,0,3,0,4,2,3,0,3,2,1,2,0,1,7,0,1



Mas, assim como SJF, pode ser aproximada...

Política *Least Recently Used – LRU - 1*

A política ótima olha para o futuro

De modo similar, podemos olhar para o passado e escolher para remoção aquela página que foi usada há mais tempo

Exemplo:

7,0,1,2,0,3,0,4,2,3,0,3,2,1,2,0,1,7,0,1

7	7	7	**2**	2	**4**	4	4	**0**	**1**	1	1	**1**							
	0	0	0	0	0	0	**3**	3	3	**0**	1								
		1	1	**3**	3	**2**	2	2	2	2	**7**								

Basta guardar a hora de acesso a cada página e escolher a mais antiga

“Basta”? É fácil, assim?

Política *Least Recently Used – LRU - 2*

Implementação complicada:

- Associa-se o tempo de acesso a cada item na tabela de páginas
  - A cada acesso, o tempo é incrementado
  - Quando cada página for removida, efetua-se uma busca na tabela de páginas

Caro!

Política *Least Recently Used – LRU - 3*

Outra implementação usa pilha de páginas:

- Cada acesso move a página para o topo da pilha
- A página no fim é retirada
- Manipulação da pilha é mais complexa

Mas... *LRU* não sofre da anomalia de Belady

Aproximações da *LRU*

Muitos processadores fornecem um auxílio para a implementação da LRU:

- Um bit de referência é ligado quando a página é acessada

Usando os recursos do hardware podemos implementar a LRU de forma muito mais eficiente:

- Basta inicializar o bit de referência com 0 e deixar o hardware colocá-lo em 1 quando a página for acessada
- Quando formos repor uma página, escolher uma com o bit de referência em 0

Eficiente, mas e quando todas as páginas tiverem sido referenciadas?

Mais Bits de Referência - 1

Podemos melhorar a precisão do método anterior:
- Guarde um byte para cada página de memória;
- Periodicamente (ex: a cada interrupção do relógio), SHR o byte de cada página e & o bit de referência mais significativo:

Byte por página	Bit de referência	Agregado	SHR
11000100	0	011000100	01100010
01110110	1	101110110	10111011

Mais Bits de Referência - 2

- Desta forma, uma página com byte de referência 00000000 não foi acessada nos últimos 8 ciclos;
- Uma página com byte 11111111 foi acessada em todos os últimos 0 ciclos

A página com o menor valor do byte de referência deve ser removida

O número de bits de história pode variar

Algoritmo da Segunda Chance

- FIFO modificado
- Escolha uma página a ser removida usando FIFO
- Mas, antes de remove-la, veja seu bit de referência:
  - Se for 0 (página não acessada), remova-a
  - Se for 1, zere-o e dê uma segunda chance à pobre página

Fácil de implementar, melhor que FIFO, mas...

- Pode degenerar em FIFO se todos os bits de referência estiverem setados

Segunda Chance ao Segunda Chance

Um algoritmo segunda chance mais sofisticado leva em conta também o bit limpo/sujo da página:

Páginas podem ser:

1. (sem ref., limpa): Não foi usada nem modificada, ótima para remover
2. (sem ref., suja): Não foi usada, mas sim modificada, Não tão boa quanto a anterior, pois tem que gravar
3. (ref., limpa): Foi usada, mas pelo menos não tem que ser gravada
4. (ref., suja): Usada e tem que gravar. Deixa por último

Escolhe-se a página a remover na ordem 1 a 4

Usado no Macintosh (oh, oh... mau sinal...)

Outros Algoritmos

- LFU: Least Frequently Used

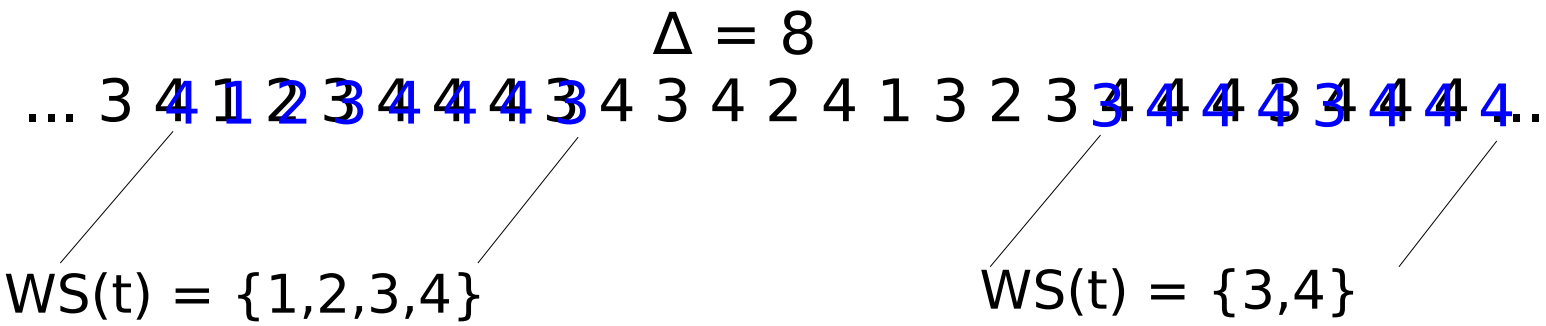
- MFU: Most Frequently Used

Implementações caras (muito embora aproximações sejam possíveis), e não aproximam a política ótima muito bem

Modelo de Working-Set - 1

A **janela de trabalho** é uma medida de localidade:

- Define-se como Δ seu tamanho
- A janela de trabalho *WS(t)* consiste das últimas Δ referências a páginas da memória



- O tamanho médio do WS de um programa, dado Δ, define o número de frames a serem alocados ao processo

Modelo de Working-Set - 2

Como definir Δ?
- Através da monitoração do comportamento de um conjunto de programas

Como alocar frames a processos?
- Informalmente: assume-se que o processo vai usar *n* frames, mas não se garante
- Formalmente, não permite a utilização de mais que *n* frames

Informalmente, o working set de um processo define o número médio de frames alocados a cada instante de sua execução:
- Ou seja, o processo vai usar, na média, n/*T*% da memória principal

Quantos processos rodar?

O S.O. deve permitir a execução de até *m* processos de tal forma que a soma de seis working sets não seja maior que o número de frames disponível

- Desta forma, todos os processos estarão executando de forma “confortável”

E se houverem mais processos?

Thrashing! (1)

thrashing: derrota, fiasco, colapso rápido e violento

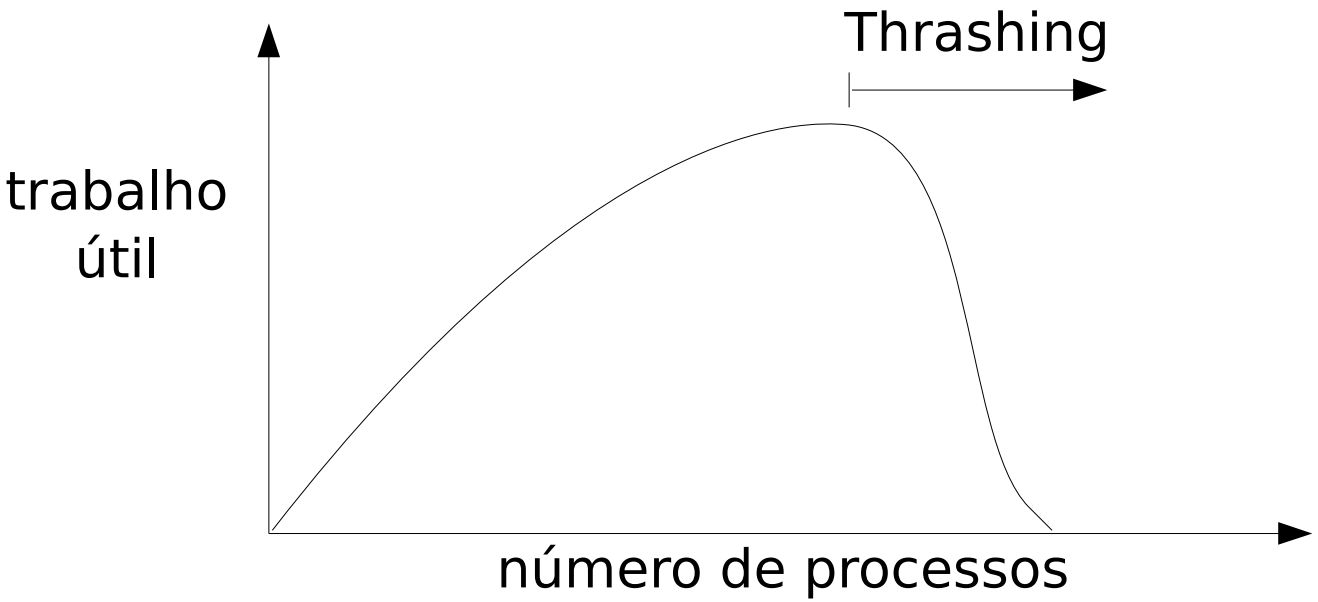
Neste caso, cada processo terá disponível para si um número de frames menor que seu working set:

- Ou seja, haverão referências frequentes a páginas ausentes:

localidade > disponibilidade

- Com isto o número de page faults aumenta significativamente, e o desempenho diminui

Thrashing! (2)



Ou seja, gasta-se mais tempo decidindo o que fazer, do que fazendo alguma coisa...
... soa familiar?

Conclusão

- O que fazer quando a memória “enche”?
  - Políticas de gerência de memória

- Políticas:
  - Política ótima: apenas como alvo de comparações
  - LRU e aproximações
  - FIFO e Segunda Chance

- Working set:
  - Quandos frames um processo precisa?
  - Thrashing: quando o S.O. trabalha mas não faz nada