Instituto de Informática - UFRGS

Sistemas Operacionais

Paginação por demanda

Aula 16

Introdução

- Paginação por demanda consiste em alocar memória (quadros) a medida que páginas são referenciadas (e continuam a ser...)
- Consiste em:
  - Espaço de swap
  - Procedimentos de copiar (movimentar) páginas entre disco e memória
    - Procedimento de pagging (page daemon)
    - *page-in* e *page-out*
  - Mecanismo de substituição de páginas
    - Algoritmos de substituição

Instituto de Informática - UFRGS
A. Carissimi -26-avr.-12

Implementação da paginação por demanda

- Uma referência está sempre associada a uma página
  - Página pertence ao espaço de endereçamento do processo
  - Página está carregada na memória (página válida)
  - Página está na área de swap (página inválida)
  - Página não está na memória, nem na área de swap (nova alocação)
    - Comportamento típico da áreas dinâmicas (pilha e heap)
- Informações mantidas na entrada da tabela de páginas associada a uma página específica
  - Bits de controle

Instituto de Informática - UFRGS
A. Carissimi -26-avr.-12

Entrada da tabela de páginas

- Informações para:
  - Tradução de endereços lógicos em físico
  - Proteção e compartilhamento: páginas read/write/execute e compartilhada
  - Carga por demanda: bit de validade (está ou não carregada na memória)
    - Nova semântica
  - Substituição de páginas:
    - Modificação (*dirty bit*)
    - Referência a página
    - Endereço do bloco do disco no *swap* onde página está mantida

Novo

Instituto de Informática - UFRGS
A. Carissimi -26-avr.-12

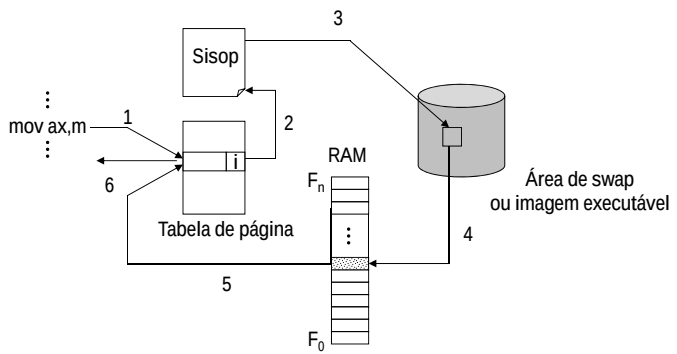
Falta de página e carga por demanda

- Na tradução de endereços consulta o bit de válido, se:
  - Presente: realiza a tradução
  - Ausente: gera interrupção de falta de páginas (*page fault*)
- Tratador de falta de páginas
  - Aciona a gerência de memória virtual
  - Carrega a página que provocou a falta para memória
    - Considera que tem quadro livre na RAM para receber a página
    - Entrada da tabela de páginas fornece o bloco do *swap* que contém a página que falta
  - Atualiza tabela de páginas
  - Executa uma política para substituição de página
- Processo em falta de página passa para o estado "bloqueado"

Page-in, page-out

- *Page-in* ocorre quando uma página a ser acessada não está na memória
  - É trivial quando se tem quadro livre na RAM
- Sem quadro livre na RAM:
  - Necessário executar uma substituição de página
    - Seleção de uma página "vítima"
    - Marcar na tabela de páginas a página "vítima" como inválida
    - Realizar *page-out* e *page-in*
    - Atualizar a entrada da nova página como válida
    - Voltar a execução da instrução que provocou a falta de página
- *Page-out*
  - Necessário fazer apenas se a página "vítima" foi modificada na memória

Esquema da paginação por demanda

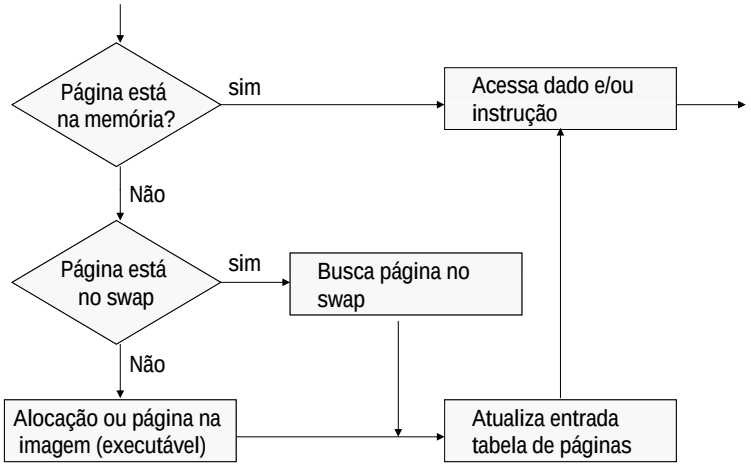


Tratamento de falta de página (*page-fault*)

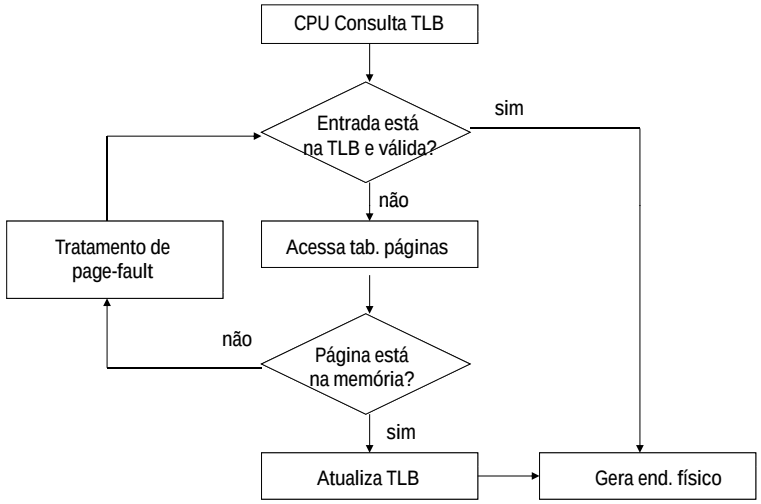
- Passos genéricos a serem realizados (visão do sistema)
  1. *Trap* para sistema operacional
  2. Salvamento de contexto
  3. Detecção de interrupção é por *page-fault*
  4. Verifica se referência é válida e determina localização da página no disco
  5. Solicita operação de leitura do disco para o frame
  6. Passa processo para um estado bloqueado e escalona outro processo
  7. Interrupção do disco (final de transferência da página)
  8. Atualiza tabela de páginas
  9. Passa processo do estado bloqueado para estado apto

Páginas associadas ao núcleo do sistema operacional estão "pinadas" na memória, isso é, NUNCA vão provocar falta de página.

Busca de páginas no disco



Interação entre sistema de paginação e TLB



Tempo efetivo de acesso a memória

- Desempenho depende da taxa de falta de páginas do sistema
  - Taxa de falta de páginas = $0 \leq p \leq 1.0$
    - p = 0; qualquer referência NÃO provoca falta de página
    - p = 1, qualquer referência provoca uma falta de página
- Tempo efetivo de acesso:
  - Tempo médio de acesso a memória "percebido" por um processo
    - Tempo consumido pela MMU para a tradução (inclui o *hit/miss ratio* na TLB e eventuais tempo de acesso tabela de página em memória)
    - Tempo consumido pela gerência de memória virtual (considera as operações *page-in, page-out* , substituição de página e tempo de bloqueio)

Desempenho da paginação por demanda

$$t_{efetivo} = (1-p) \times (t_{TP} + t_{MEM}) + p \times (t_{TP} + t_{TFP} + t_{TP} + t_{MEM})$$

- onde:
  - t_{TP} : tempo acesso a tabela de páginas (TLB ou memória)
  - t_{TFP} : tempo de tratamento da falta de páginas
  - t_{MEM} : tempo de acesso a uma posição de memória
- Pode ser aproximado por:

$$t_{efetivo} = (1-p) \times t_{acesso} + p \times t_{page\_fault}$$

- onde:
  - t_{acesso} é tempo para acessar página na memória
  - t_{page_fault} é o tempo para o tratamento da falta de páginas e acessar a posição de memória que a provocou

Exemplo: desempenho da paginação por demanda

- Exemplo 1:

  tempo acesso: 100 ns

  tempo page_fault: 25 ms

  t_e =(1-p) x 100ns + p x 25000000ns

  t_e = (100 + 24999900 x p) ns $\rightarrow$ p=1/1000

  $t_e \cong$ 25 us (250 x t_{acesso})

- Exemplo 2: Objetivo=> aumentar no máximo em 10% o tempo de acesso

  $110 \cong 100 + p \times 25000000$

  $10 \cong 25000000p$

  p = 0.0000004 (1 *page fault* a cada 250000 acessos)

Tempo efetivo de acesso considerando TLB

- Mais precisamente:

$$t_{efetivo} = pr_2 \times (t_{TLB} + t_{MEM}) + (pr_1 - pr_2) \times (t_{TLB} + 2 \times t_{MEM}) + \\ + (1 - pr_1) \times (t_{TLB} + t_{MEM} + t_{TFP} + t_{TLB} + 2 \times t_{MEM})$$

- Onde:
  - pr_1: probabilidade da página estar em memória
  - pr_2: probabilidade da entrada da página estar na TLB (*hit ratio*)
  - t_{mem}: tempo de acesso a memória
  - t_{TLB}: tempo de acesso a TLB
  - t_{TPF}: tempo de tratamento de *page fault*

Como melhorar o desempenho da paginação por demanda?

- Otimizar o procedimento de busca de páginas no disco
  - Arquivo de swap versus partição específica para swap
- Manter a taxa de falta de páginas (*page fault*) pequena
  - Realizar pré-paginação especulativa (trazer página p_i e suas redondezas)
    - Noção de região de localidade
- Dimensionamento apropriado do tamanho da página
  - Nem sempre é possível devido ao suporte de hardware disponível
  - Compromisso custo x benefício
    - Tamanho da tabela de páginas, fragmentação interna, regiões de localidades, uso eficiente da memória

Alocação de memória a um processo

- Taxa de falta de páginas varia com a quantidade de memória alocada
- Três zonas:
  - Pouca alocação, alta taxa de falta de páginas (subdimensionamento)
  - Média alocação, média taxa de de falta páginas (zona de "conforto")
  - Alta alocação, baixa taxa de falta de páginas (superdimensionamento)
- Não é trivial achar a zona de conforto
  - Varia de processo a processo
  - Um mesmo processo tem padrões diferentes de comportamento durante a sua execução.

Ultrapaginação (*thrashing*)

- Zona de subdimensionamento
  - Processos vão para o estado de bloqueado (tratamento page-fault)
  - CPU realiza trocas de contexto e gerencia o page-fault
- Situação *thrashing*
  - Alto tráfego de páginas (page-in, page-out) e baixo uso eficiente da CPU
  - Apenas baixa eficiência não é sinônimo de thrashing
    - Pode ser apenas que há poucos processos e, por coincidência, muitos estão no estado de bloqueado aguardando E/S ou evento sincronização
- Situação "perversa"
  - CPU pode ficar ociosa (E/S é demorado) então sistema operacional aceita criação de mais processos

Solução: Aumentar a memória disponível para processos, retirando alguns deles da memória (reduz grau de multiprogramação)

Determinação do tamanho de página

- Balanceamento de fatores conflitantes
  - Página pequena
    - Redução da fragmentação interna
    - Região de localidade tendem a ser pequenas
    - Aumento da tabela de página
    - Diminui a taxa de transferência disco-memória
  - Página grande
    - Aumenta a fragmentação interna
    - A página possui mais código que o necessário (> região de localidade)
    - Diminui a tabela de página
    - Aumenta a taxa de transferência disco-memória

Análise matemática

- Processo de *s* bytes, páginas de *p* bytes e cada entrada da tabela de páginas com *e* bytes, se tem:

$$custo = \frac{s.e}{p} + \frac{p}{2}$$

$$-\frac{se}{p^2} + \frac{1}{2} = 0$$ Derivada primeira em relação a p

$$p = \sqrt{2se}$$ Tamanho ótimo para *p*

- Tendência: aumentar o tamanho das páginas
  - Hoje é comum ainda ter-se páginas de 4 Kbytes

TLB, cache e desempenho

- Novo problema: TLB ficaram pequenas
  - Aumento da memória, aumenta número de páginas, diminui o hit ratio, aumenta o tempo médio
  - Alcance_TLB = *nro_entradas* x *tam_página*
- Outro problema: tamanho das caches
  - Caches são maiores que o alcance_TLB
  - Falta na TLB obriga acesso a RAM, prejudicando o sistema de cache
    - Solução: cache usar endereços lógicos e ficar "na frente" da TLB
    - Continua a prejudicar o desempenho da memória virtual
- Solução trivial: páginas maiores, mas não é bom
- Introdução das superpáginas

Superpáginas

- Recurso de hardware de alguns processadores (TLB)
- Superpágina:
  - Tamanho é uma potência de 2 do tamanho da página
    - Informação do tamanho é dado na entrada da TLB
  - Endereços lógicos e físicos são alinhados em multiplos do tamanho da superpágina
- Endereço lógico passa a ser composto por páginas e superpáginas
  - Paginas sucessivas viram um superpágina (*promotion*)
  - Superpáginas podem virar páginas simples (*demotion*)

Leituras complementares

- A. Tanenbaum. *Sistemas Operacionais Modernos* (3ª edição), Pearson Brasil, 2010.
  - Capítulo 3: seção 3.7
- A. Silberchatz, P. Galvin; *Sistemas Operacionais*. (7ª edição). Campus, 2008.
  - Capítulo 8 (seções 8.6 e 8.7)
- R. Oliveira, A. Carissimi, S. Toscani; *Sistemas Operacionais*. Editora Bookman 4ª edição, 2010
  - Capítulo 6 e capítulo 7 (seção 7.5)